e-Informatica

software engineering journal

2026 volume 20 issue 1



e-Informatica

e-Informatica

software engineering journal

2026 | volume 20 | issue 1



e-Informatica



Wrocław University
of Science and Technology

Editor-in-Chief
Lech Madeyski (Lech.Madeyski@pwr.edu.pl, https://madeyski.e-informatyka.pl)
Associate Editors
Mirosław Ochodek (Miroslaw.Ochodek@put.poznan.pl)
Adam Przybyłek (Adam.Przybylek@universityofgalway.ie)
Editor-in-Chief Emeritus
Zbigniew Huzar (Zbigniew.Huzar@pwr.edu.pl)

Faculty of Information and Communication Technology, Department of Artificial Intelligence
Wrocław University of Science and Technology,
50-370 Wrocław, Wybrzeże Wyspiańskiego 27, Poland

e-Informatica Software Engineering Journal
www.e-informatyka.pl, DOI: 10.37190/e-inf

Editorial Office Manager: Wojciech Thomas

Typeset by Wojciech Myszka with the LaTeX Documentation Preparation System

All articles in the journal are shared under a Creative Commons-ShareAlike 4.0 International license (CC BY-SA 4.0)

OFICYNA WYDAWNICZA POLITECHNIKI WROCŁAWSKIEJ
Wybrzeże Wyspiańskiego 27, 50-370 Wrocław
www.oficyna.pwr.edu.pl;
e-mail: oficwyd@pwr.edu.pl; zamawianie.ksiazek@pwr.edu.pl

ISSN-L 1897-7979

Editorial Board

Editor-in-Chief

Lech Madeyski (Wrocław University of Science and Technology, Poland)

Associate Editors

Mirosław Ochodek (Poznań University of Technology, Poland)

Adam Przybyłek (University of Galway, Ireland)

Editor-in-Chief Emeritus

Zbigniew Huzar (Wrocław University of Science and Technology, Poland)

Editorial Board Members

Pekka Abrahamsson (NTNU, Norway)
Muhammad Ovais Ahmad (Karlstad University, Sweden)
Nauman bin Ali (Blekinge Institute of Technology, Sweden)
Apostolos Ampatzoglou (University of Macedonia, Thessaloniki, Greece)
Sami Beydeda (ZIVIT, Germany)
Miklós Biró (Software Competence Center Hagenberg, Austria)
Pearl Brereton (Keele University, UK)
Mel Ó Cinnéide (UCD School of Computer Science & Informatics, Ireland)
Steve Counsell (Brunel University, UK)
Maya Daneva (University of Twente, The Netherlands)
Norman Fenton (Queen Mary University of London, UK)
Joaquim Filipe (Polytechnic Institute of Setúbal/INSTICC, Portugal)
Thomas Flohr (University of Hannover, Germany)
Francesca Arcelli Fontana (University of Milano-Bicocca, Italy)
Félix García (University of Castilla-La Mancha, Spain)
Carlo Ghezzi (Politecnico di Milano, Italy)
Janusz Górski (Gdańsk University of Technology, Poland)
Eduardo Guerra (Free University of Bozen-Bolzano, Italy)
Tracy Hall (Lancaster University, UK)
Andreas Jedlitschka (Fraunhofer IESE, Germany)
Kai-Kristian Kemell (Tampere University, Finland)
Barbara Kitchenham (Keele University, UK)
Stanisław Kozielski (Silesian University of Technology, Poland)
Pericles Loucopoulos (The University of Manchester, UK)
Kalle Lyytinen (Case Western Reserve University, USA)
Leszek A. Maciaszek (Wrocław University of Economics, Poland
and Macquarie University Sydney, Australia)
Jan Magott (Wrocław University of Science and Technology, Poland)
Zygmunt Mazur (Wrocław University of Science and Technology, Poland)
Bertrand Meyer (ETH Zurich, Switzerland)
Matthias Müller (IDOS Software AG, Germany)
Jürgen Münch (University of Helsinki, Finland)
Jerzy Nawrocki (Poznan University of Technology, Poland)
Janis Osis (Riga Technical University, Latvia)

Fabio Palomba (University of Salerno, Italy)
Mike Papadakis (Luxembourg University, Luxembourg)
Mirko Perkusich (VIRTUS, Brazil)
Kai Petersen (Hochschule Flensburg, University of Applied Sciences, Germany)
Adam Przybyłek (Gdańsk University of Technology, Poland)
Łukasz Radliński (West Pomeranian University of Technology in Szczecin, Poland)
Guenther Ruhe† (University of Calgary, Canada)
Krzysztof Sacha (Warsaw University of Technology, Poland)
Martin Shepperd (Brunel University London, UK)
Rini van Solingen (Drenthe University, The Netherlands)
Miroslaw Staron (IT University of Göteborg, Sweden)
Tomasz Szmuc (AGH University of Science and Technology Kraków, Poland)
Guilherme Horta Travassos (Federal University of Rio de Janeiro, Brazil)
Adam Trendowicz (Fraunhofer IESE, Germany)
Burak Turhan (University of Oulu, Finland)
Rainer Unland (University of Duisburg-Essen, Germany)
Sira Vegas (Polytechnic University of Madrit, Spain)
Corrado Aaron Visaggio (University of Sannio, Italy)
Bartosz Walter (Poznan University of Technology, Poland)
Dietmar Winkler (Technische Universität Wien, Austria)
Bogdan Wiszniewski (Gdańsk University of Technology, Poland)
Krzysztof Wnuk (Blekinge Institute of Technology, Sweden)
Marco Zanoni (University of Milano-Bicocca, Italy)
Jaroslav Zendulka (Brno University of Technology, The Czech Republic)
Krzysztof Zieliński (AGH University of Science and Technology Kraków, Poland)

Contents

— Disambiguating Software Property Descriptions: A Solution Proposal and Evaluation
— *Efi Papatheocharous, Séverine Sentilles, Kai Petersen, Federico Ciccozzi* 260101
— A Defect Classification Framework for AI-Based Software Systems (AIODC)
— *Mohammed Alannsary* 260102
— Bridging Humans and LLMs: Investigating Human-AI Collaboration in Multi-agent Requirements Analysis for Organizational AI Adoption
— *Malik Abdul Sami, Zheying Zhang, Muhammad Waseem, Kai-Kristian Kemell, Zeeshan Rasheed, Tomas Herda, Pekka Abrahamsson* 260103
— Realising the Role of Generative AI Among Project Management Professionals
— *Kari Sainio, Petri Kettunen, Pekka Abrahamsson* 260104
— Index-Based Type-3 Clone Detection
— *Zdenek Tronicek* 260105
— From Draft to Standard: An Expert-Validated Catalog of API Gateway Software Metrics
— *Eder dos Santos, Sandra Casas* 260106
— Bridging Participatory Co-Design and Model-Driven Engineering for Community-Based Web-3D Framework Development: Method and Validation
— *Ratna Wardani, Masduki Zakaria, Tika Novita Sari, Dian Novian* 260107
— KPSO-Fuzzy: An AI-based Hybrid Approach for Software Requirements Prioritization
— *Rahila Anwar, Muhammad Bilal Bashir* 260108
— A Modular Multi-Agent LLM Architecture for Text-to-Diagram Generation and User-Guided Refinement
— *Hartwig Grabowski* 260109
— Onboarding Newcomers in Large-Scale Agile Projects: A Multiple Case Study
— *Pavithra Herath, Muhammad Ovais Ahmad, Tomas Gustavsson* 260110

e-Informatica Software Engineering Journal
Volume 20, Issue 1, pages: 1–31
DOI: 10.37190/e-Inf260101
BibTeX

Disambiguating Software Property Descriptions: A Solution Proposal and Evaluation

Efi Papatheocharous*, Séverine Sentilles, Kai Petersen, Federico Ciccozzi
*Corresponding author: efi.papatheocharous@ri.se

Article info

Keywords:
extra-functional properties
software engineering
modelling
ontology
software quality
evaluation

Submitted: 18 Sep. 2025
Revised: 11 Nov. 2025
Accepted: 19 Nov. 2025
Available online: 26 Nov. 2025

Abstract

Context: The state-of-the-art and practice on software quality is growing constantly and presents several challenges. The ever-growing body of knowledge on the topic, obfuscates the situation further, the lack of explicit structure makes it difficult to identify which properties exist today and how they can be evaluated, two critical aspects to increase software quality.
Objective: A step to disambiguate software properties descriptions is made via a Property Model Ontology (PMO) and steps towards the evaluation of the structure are carried out together with experts. The objectives of this paper are: 1) present in detail the PMO, 2) describe the research process used to develop and evaluate the PMO, and, 3) exemplify the usage of the PMO through real instantiations obtained from practitioners and researchers.
Method: Expert interviews and qualitative research methods are used to evaluate the PMO and develop a proof of concept.
Results: The PMO consists of concepts describing extra-functional properties (EFPs) and their evaluation methods, i.e., how to measure the properties. The PMO is instantiated in a modelling environment through a metamodel and an online web-content management system.
Conclusions: Consensus on the definition and structure of EFPs is achieved and a common understanding on how they can be reused in practice.

1. Introduction

As defined by IEEE [1], software quality is the degree to which software possesses a desired combination of properties such as reliability, interoperability, performance, maintainability, and usability. In other words, software quality is embodied through software properties. Regardless, the exact list of properties to be prioritised, how they can be defined can be sometimes left open for interpretation. Software properties are commonly and indifferently referred to as extra-functional properties (EFPs), non-functional properties (NFP), quality properties, quality of service (QoS), or simply metrics.

To mitigate fundamental issues with software quality and particularly challenges around the definitions of properties, it is necessary to carry out extensive investigation, clarity and availability activities of subject domain knowledge and this can be performed through

© 2026 The Authors. Published by Wrocław University of Science and Technology Publishing House.
This is an open access article under the CC BY-SA 4.0 license international.

the use of ontologies. Ontologies and their applicability details can enable measurement, comparison and evaluation of a set of critical quality requirements and help reach to improved architectural decisions. Towards this, one needs to specify a systematic (in terms of consistent), understandable and useful way to define *"how to describe"* and *"how to measure"* EFPs as a foundation.

Recent works [2] pinpoint the lack of empirical validations of quality properties and a gap between solutions proposed in academic settings (i.e.,the NFR method [3], KAOS [4], QUPER [5], and i* [6,7]) and the needs of practitioners [8]. In this paper we bridge this gap by following a dual approach, specifying and validating an ontology, namely the Property Model Ontology (PMO), taking the perspective of academic and industrial validation. The PMO is an ontology which aims to represent the subject domain with as much truthfulness, clarity and expressivity as possible and results to increased expressivity of knowledge, common understanding, clarity of negotiations and problem-solving possibilities for practitioners. An evaluation step is detailed in this work, which includes semantic and pragmatic quality assessments, crucial for structuring knowledge on software quality properties and removing obstacles that obfuscate the state-of-the-art.

An initial version of the PMO for the automotive domain has been described in [9] without however making concrete steps towards its evaluation with experts and collecting feedback from industry. The main contribution of this work is detailing the methodology for developing the ontology, as well as, presenting the evaluation steps carried out through real practical examples. This is a significant advancement from the initial version described in previous work, which did not include concrete steps towards evaluation with experts and collecting feedback from industry.

Since its initial articulation (2016), the PMO has seen limited traction beyond our own pilots. We acknowledge this explicitly and attribute it to (i) the upfront cost for practitioners to model properties formally when informal templates suffice in day-to-day work, (ii) fragmentation of terminology across sub-communities (quality models, measurement, safety/security) which reduces the perceived need for yet another schema, (iii) a lack of tight integration with commonly used toolchains (requirements/issue trackers, architecture decision records, modelling tools), and (iv) scarce, end-to-end exemplars that demonstrate immediate value. In this paper, we report the steps we have taken to address these barriers – publicly releasing PROMOpedia, providing ready-to-use templates, instantiating PMO in a metamodeling environment, and evaluating it on two concrete domains (automotive and security) – and outline further actions to grow adoption (see Sections 5 and 6).

The PMO is a core ontology targeted for the software and systems engineering domain and consists of a structured set of properties to describe software component or system properties, as well as, their evaluation methods, and it has been evaluated through two evaluation rounds, involving in total 10 industrial practitioners and 4 academic researchers. The motivation to develop the ontology is to make a step towards achieving a consensus on the definition and structure of EFPs and share a common understanding on how they can be used. This comprehensive dual evaluation including both semantic and pragmatic quality assessment of the PMO is crucial for structuring knowledge on software quality properties and removing obstacles that obfuscate the state-of-the-art. Such thorough validation is often missing in existing works, making this work a valuable contribution to the field. Another important contribution this work addresses is the issue of scattered knowledge across various communities, domains, and forums. By identifying commonalities and differences, it provides a common ground for fostering discussions and integrating isolated islands of knowledge.

Moreover, to enable software and systems quality assessment the following steps need to be carried out:

1. Establish a prioritised list of EFPs that provide guarantees to the targeted system-level properties;
2. Identify the methods that can be used to evaluate the EFPs;
3. Select to apply one (or several) of the identified methods to evaluate EFPs;
4. Based on the obtained results, identify solutions to achieve the system-level properties aimed for.

In this work, we contribute on the following three aspects: 1) present the evaluated details of the PMO, 2) describe the research process carried out to develop and evaluate the PMO, and, 3) exemplify the usage of the PMO through practical instantiations obtained from industry and academia. The results of these real-world instantiations are presented too.

The remainder of the paper is structured as follows: Section 2 presents the background and motivation based on related work. Section 3 describes the method for constructing the ontology. Section 4 describes the current version of the ontology which is also evaluated in Section 5 by experts and industrial practitioners. The meta-model of the PMO is also described in Section 5. Section 6 discusses the results and lessons learned, limitations and implications for researchers and practitioners. Section 7 concludes the paper and describes future work.

2. Background and motivation

2.1. Software quality, properties, property models and quality models

A plethora of research on extra-functional properties (EFPs) exists. The body of knowledge is however scattered across many communities, domains and forums, encompassing areas such as testing, requirements engineering, maintainability, decision-making, object-oriented programming, component-based software engineering, model-driven engineering, web-services, software architecture, and many more. This creates many isolated islands of knowledge with limited opportunities for cross-fertilization. Consequently, there is currently no commonly acknowledged definition of what EFPs precisely are. One simple definition can be given by contrasting extra-functional properties to software functionality. Software functionality defines *"what"* a system does, an EFP describes instead *"how–well"* a system performs [10].

The lack of general definitions coupled with the absence of a comprehensive property list and the scattered research results on the topic have led to the creation of many taxonomies and classifications. Properties are generally described in those classifications through a hierarchy of sub-characteristics, which can be further divided into sub-attributes which are verifiable and measurable. For example, reliability in ISO/IEC 9126 [11], is defined as *"a set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time"*, i.e., reliability consists of maturity, fault tolerance, recoverability and reliability compliance.

Some of these classifications are referred to as quality models. As defined by Albin in [12], a quality model is a taxonomy of quality attributes and their relationships. It serves as a framework for detailed system specification, design evaluation, and system testing. Examples of these models include McCall's [13], Boehm's [14], the ISO/IEC

9126 [11] and its successor the ISO/IEC 25010 SQUARE quality model [15]. Several other proposals for classifying EFPs have been developed: Laprie's dependability classification [16], Saraiva's catalog of maintainability metrics [17] or Crnkovic's classification of property's composability [18] are few examples in a long list. An overview of a timeline of existing classifications of software properties can be found in [19]. In [8] a decline in the popularity, relevance and importance of quality models after 2010s has been observed. The authors in [20] try to bridge the gap between abstract quality models/standards and practical tool implementations without addressing the inconsistencies, as pointed out in several studies such as [21–23]. The main problem with these taxonomies, classifications and quality models is that they are typically non-orthogonal, inconsistent among each other or even among themselves, even after decades of research, and lack a common ground as recent systematic literature reviews state [24]. Deissenboeck et al. [23] list several issues as follows:

1. *Ambiguous*. Software quality models generally do not conform to an explicit metamodel, that is, they lack precise semantics, and the interpretations of the concepts and their relationships are left open to the readers. Implied corollaries is that they have:
   a) *Overlapping concepts*. There are implicit cross-cutting relations between the properties in the quality models which are not captured nor explained. For example, in ISO 25010, security is orthogonal to reliability. However, an unreliable system might negatively affect security, such as when the system is under a Denial-of-Service (DoS) attack.
   b) *No guidelines*. They do not provide satisfying user's guidelines on how to use the quality models or how to tailor them for use in a given context.
2. *Limited*. They only support a given view on software quality such as quality in use vs. quality requirements, or system property vs. implementation property.
3. *Arbitrary*. They lack clearly defined decomposition criteria that determine how complex quality concepts ought to be decomposed and the selections of the core properties often seem arbitrary. For example, in ISO 25010, it is debatable that availability should be a sub-characteristic of reliability.
4. *Limited usability*. The properties are too abstract to be straightforwardly used in the verification and validation of a concrete software system (e.g., maintainability [25], usability) and therefore their usability is limited.
5. *Improper metrics*. The supporting metrics often have:
   a) Unclear motivation and validation [26];
   b) Ignore context factors which might influence the validity of a property, i.e., the applicability in a given context [27];
   c) Fail to disclose the impact of the supporting metrics on quality [28].

Glinz [22] identifies similar problems in the classification of non-functional requirements, namely the usage of divergent concepts in the sub-classifications and ambiguity in the types of requirements being classified.

This leads to loose and ambiguous terminology usage [9, 29, 30], uncontrolled and subjective measurement activities and, more notably, to the following issues which obfuscate the state-of-the-art and practice even further:

1. *Synonymous EFPs*. One property can be found under many different denominations, which are in fact completely equivalent (e.g., *"Class Coupling"*, *"Coupling between Objects"* [31]).
2. *Polysemous EFPs*. Two properties can have the same name but refer to two completely different things. The difference can be in semantics, value representation, usage, etc.

For example, the *"DC metrics"* in maintainability can refer either to the *"Degree of Cohesion"* or *"Descendants Count"* [31].

3. *Paronymous EFPs*. Two properties can have the same etymological root, but different meanings (e.g., "composability" and "compositionality", which both derive from "composition", but imply different composition qualities of a component assembly).
4. *Incomplete and inconsistent specifications for EFPs evaluation*. The methods used to evaluate the value of a property may not clearly specify what they measure and how. For example, ambiguous ways to measure software size exist, using the physical Source Lines of Code (SLOC) or the logical SLOC. Even though the definitions of the latter vary, SLOC are stated without a clear definition, and depending on the specific use they might include comments and bracket lines, or not [32].

In summary, the topic of classifying software properties is, to a large extent, subject to much controversy and on-going discussions. This makes selecting software components based on properties and evaluating the selections especially difficult [33]. In order to mitigate the challenges around the definition of EFPs and increase clarity in structure and measurement, the use of ontologies is proposed. To the best of our knowledge, no other effort on developing an ontology exists for EFPs that targets together with systematic and formal definitions as well as practical evaluation through industrial practitioners and academics. Our work proposes the PMO as a useful way to describe and reuse software properties and their evaluation methods, and also carries out an evaluation together with experts.

2.2. Ontologies in software engineering

In software engineering, the study of human-created knowledge, also known as epistemology, is performed through the use of ontologies. An ontology is a formal, often taxonomic, organization of concepts [34]. A recent literature study identifies an increased use of ontologies in areas of computer science, science and technology and engineering [35], aiming to improve software quality management. Extending ontologies, the article discusses how AI and machine learning can further help create ontologies that update and adapt in real-time in the future. Other recent examples of ontologies for requirements engineering [36] highlight the persistence of ambiguity in natural language for requirements definition.

In the literature and to cope with different purpose and focus, different types of ontologies exist. Their differences have been summarized in several classifications [37–39] although one simple classification differentiates between foundational/high-level/reference ontology and domain ontology, i.e., between generic and specific ontologies respectively.

Foundational ontologies are developed with the aim of *"representing the subject domain with truthfulness, clarity and expressivity, regardless of computational requirements"* [40]. The main goal is to help modellers externalize their knowledge about the domain, to make their ontological commitments explicit, to support meaningful negotiations, and to improve the tasks of domain communication, learning and problem solving as best as possible [41]. Simply said, it is a system of categories independent of implementation language or technical constraints [42], and the relation between these categories. Examples of foundational ontologies in software engineering include the Unified Foundational Ontology (UFO) [43], the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [44] and the software measurement ontology [45]. OntoGLOSE, an ontology of the IEEE Computer Science glossary for software engineering, is another example [46].

In contrast, a *domain ontology* is used as a concrete engineering artifact designed for a specific purpose, and represented in a specific language. As identified in [34], this type of

ontology is often inappropriately referred to just an "ontology". Nowadays, most of the research work dealing with ontologies in software engineering actually belong to domain ontology. This leads to confusions and ambiguity. Examples of domain ontology include the ontology to document a quality scheme specification of a software product [47] and the software maintenance ontologies by Anquetil and de Oliveira [48]. Many more examples can be found in domain ontology repositories such as the Protege Ontology Library[1].

Ontologies can also be classified according to their level of expressiveness [49]. Similarly to expressive power in programming languages, the level of expressiveness of an ontology refers to its ability to describe the concepts. More expressive ontologies allow specifying all the ontological concepts completely and unambiguously. This is especially useful to highlight the differences between often misunderstood and misused ontology-related terminology. At one side of the spectrum are the informally[2] defined ontologies such as vocabularies, glossaries and thesauruses. A *vocabulary* is a finite list of terms whereas a *glossary* adds definitions in natural language to the terms. *Thesauruses* are glossaries semantic additions such as related terms (e.g., synonyms, antonyms). *Folksonomies* are informal hierarchies created by tagging content. At the other end of the spectrum of expressiveness are the more formally defined ontologies. It includes formal hierarchies, frames, ontologies with value constraints and ontologies with logical constraints. *Formal hierarchies* relate to the concepts of generalisation, inheritance and instantiation in software modelling, by defining an "is-a" relationship between two terms. *Frames* are formal hierarchies in which concepts also include property information. In *ontologies with value constraints*, value restrictions are placed on the properties. Finally, *ontologies with logical constraints* are the most expressive type of ontologies by to enabling for example to specify first order logic constraints between terms.

Due to their nature, ontologies provide effective solutions to: i) clarify the structure of the knowledge, ii) reduce conceptual and terminological ambiguity, and iii) facilitate knowledge sharing and communication. As a result, they can play a key role to mitigate current fundamental problems with ambiguity in the state-of-the-art EFP descriptions in software quality and move a step forward in automated intelligent requirement creation and validation.

3. Architecting the ontology

Our research methodology followed the four basic steps for ontology construction described in [50], augmented with an additional step for the evaluation. The evaluation step is detailed in Section 5. Our overall methodology process is illustrated in steps in Figure 1.

Step 1 was conducted first, iteratively followed by the execution of step 2, and steps 3 and 4 in parallel. For each iteration of steps 2–3–4, step 5 was then performed. Details on each step is provided below.

3.1. Step 1: Define ontology domain and scope (purpose)

The scope/purpose of the ontology was defined by answering the question *"What is the ontology going to be used for?"*. Initially, we defined the domain and purpose of the ontology to be relevant for targeting to facilitate objective and informed trade-off decisions for

[1] https://protegewiki.stanford.edu/wiki/Protege_Ontology_Library#OWL_ontologies

[2] Here formal refers to the ability to be interpreted by a machine.

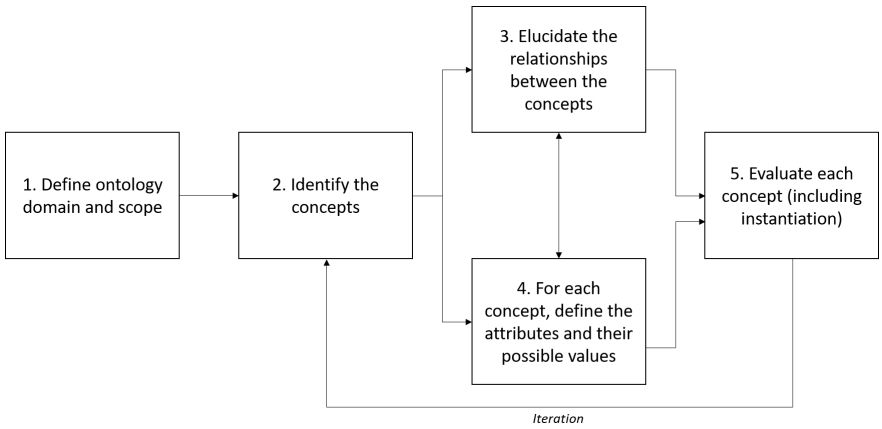


Figure 1. Process for building and evaluating the ontology

software assets based on software quality. The reason for developing an ontology is to explicitly defining the key characteristics of properties and evaluation methods through well-defined taxonomies. An important aspect for the ontology is thus to capture what practitioners need to know for trading off properties to select the most suitable assets and make more informed decisions for software-intensive systems. Answering this, requires identifying of the basic concepts to describe properties and also their evaluation methods (to make possible the trade-off, comparison, estimation, calculation and measurement), which is the next step. However, these aspects are not limited to software-intensive systems. They are also relevant for any type of software systems developed according to software engineering's principles. Therefore, we define our ontology domain and scope (purpose) more generally to any area of software engineering where evaluation and trade-off of software quality is necessary.

3.2. Step 2: Identify the concepts

Identifying the concepts naturally entailed ways to describe different types of properties. Given the previous step, the notions included into our ontology were made clear: the concepts we selected needed to support trade-off and evaluation of properties. We have thus identified that "property model" "extra functional properties" and "evaluation method" are the core concepts that should be part of our ontology. The concepts were identified during a workshop held between the authors where after discussion consensus was created. The concepts are tightly related to the purpose of the ontology, i.e., to support trade-off and evaluation of properties, as mentioned above. The reasoning behind selecting these concepts is that they can constitute the basis for collecting definitions and measurement methods within the software and systems engineering domain, that can increase the usability of EFPs and evaluation methods in practice. In addition, as explained in Section 2.1 there is currently no commonly acknowledged definition of what EFPs precisely are and how can they be evaluated. Next step, was clarifying the relationships between the concepts (Step 3).

3.3. Step 3: Elucidate the relationships between the concepts

To elucidate the relationships between the concepts of the ontology, a list of questions were developed: *"Which extra functional properties should I consider when choosing an*

architectural alternative or component?", *"How is suitability of a component evaluated?"*, *"Which critical extra functional properties appear in specific industrial domains, such as the automotive (as an example)?"*, *"How are extra functional properties formalised in scientific publications?"*, *"How is the concept of property models formalised in scientific publications"*, *"Which aspects of an evaluation method are important that affect its selection for assessing properties?"* and *"How many evaluation methods are preferred to use to sufficiently evaluate extra functional properties?"*.

The questions were developed during the workshop held between the authors where after discussion there was agreement on how the PMO could be used by researchers and practitioners. The questions were used to formulate the relationship between the core concepts and to see whether the ontology contains enough information to answer such type of questions. The list of questions was not exhaustive, in any way, but was used to clarify the required level of details or representation of the concepts and their relationships.

In the related work (Section 2) we mention works that have tried to define the concepts. Additionally, in Section 5 we include a literature survey carried out in the automotive domain to select important properties for the domain and instantiate our PMO.

3.4. Step 4: For each concept, define the attributes and their possible values

This step involves establishing for each of the concepts their attributes (what pieces of information are important to describe them) and their possible values. Important questions that we used to carry out this were: *"Which attributes should be used to formalise property models?"*, *"Which attributes should be used to define EFPs?"*, *"Which attributes should be used to describe evaluation methods?"*, *"Are there any existing definitions that can be reused?"*, *"What characteristics should the attributes have?"* and *"What is important to include in these attributes to make them usable?"*. Answering these questions, led us to initial attribute definitions which were then evaluated with specific values.

For the concept of "property" our starting point was the definition proposed in [10]. We realised that since both works served the same purpose, i.e., providing an unambiguous definition and precise semantics for EFPs, we relied on the expert knowledge from that work of one of the authors, to enable our extensions to it. We worked initially as follows: we adapted the definition of the concepts of "attributables" (i.e., the entities of a component model a property refers to) and we disregarded the concept of "support mechanism" (i.e., the set of underlying mechanisms used to manipulate EFPs in a consistent way) for the development of our ontology.

Then, we started by defining the attributes of the concept of "evaluation method", such as *MethodID*, *Name*, *Output*, and *Unit*. We then instantiated the evaluation methods based on a few evaluation method descriptions that we were familiar with and we also reviewed each other's interpretation of the descriptions. Details on the evaluation step are in the next subsection (step 5). Based on this evaluation step, we carried out improvements and then we followed an iterative process, where we executed again steps 2, 3, and 4, until no more improvements were identified.

In these iterations we were able to complement the definitions of the concepts with additional attributes gradually (e.g.,*Applicability, Formula*, etc.) and carry out refinements of the existing definitions. In the end, we created a generic textual template that was used as basis for describing property models in a systematic way.

As previously mentioned, this step was carried out in parallel with the step above, i.e., while carrying out this step, we evaluated the result of our previous step, so as to makes

sure that the core concepts defined were still relevant and sufficient to build upon them our ontology. By enumerating the important attributes for defining our core concepts we were able to expand top-down and then review bottom-up the PMO. The details of our evaluation are provided in Section 5.

Due to the current unstructured state of the body of knowledge in software quality and software engineering, we decided to not provide list of possible values for each of the identified attributes. Defining values is not mandatory for ontologies as it requires having a clear, well-structured and explicit domain knowledge. This is not the case with software quality and software engineering, where many discussions are on-going towards the definitions of core concepts (e.g., software components [21], software assurance [51], software security [52]). As a result, we decide to let the list of values open-ended and with unrestricted format, which facilitate recording currently available information, without compromising the ontology. This restrict the expressive power of the ontology but foster its use.

3.5. Step 5: Evaluate each concept (including instantiation)

The evaluation consisted of two sub-steps and after each of them we carried out amendments to the ontology. These were suggested and then rated (voted individually by the researchers involved in the process) and disagreements were resolved after discussions between the researchers. The two sub-steps performed for evaluation are:

1. Perform a literature survey to find relevant to the industry concepts and attributes (details are found in [9]);
2. Perform an evaluation with academics (collect feedback and instantiations from researchers) and practitioners (collect feedback and instantiations from industrial practitioners).

   The details of the steps and the results of the evaluation are presented in Section 5.

4. The property model ontology (PMO)

In this section, we provide the formal definition for our Property Model Ontology (PMO). It starts by defining the concept of property model in Section 4.1, and continues by formalising its key sub-concepts, i.e. EFPs in Section 4.2 and evaluation methods in Section 4.3. The version presented in this section already incorporates the changes made based on the feedback gathered during the evaluations.

4.1. Property model

In this work, we consider a property model as an EFP and the non-empty set of methods that can be used to assess the property. A property model should be described through well-specified characteristics that would allow:

1. comparing different property models;
2. describing an EFP unambiguously;
3. ensuring that an evaluation method is associated with the appropriate EFP definition;
4. comparing different evaluation methods based on facts; and
5. identifying the most suitable methods to evaluate an EFP in a given context (e.g., quality of inputs, applicability in the project, effort).

Thus, we formally specify a property model as a pair consisting of an EFP and a set of evaluation methods so that: $PropertyModel = \langle Property_{Type}, Method^{+} \rangle$ where,

- $Property_{Type}$ is the specification of the EFP used in the property model;
- $Method^{+}$ is a non-empty set of evaluation methods that can be used to concretely assess (estimate, calculate, measure, etc.) the EFP.

In particular, this implies that different evaluation methods can be used for the evaluation of a given EFP. This allows using several competing methods to evaluate an EFP and compare their outcomes.

4.2. Extra-functional property (EFP)

Properties are the expression of "how–well" a software entity performs, i.e., its inherent quality. Concretely, a property is a, qualitatively or quantitatively, quantifiable characteristic specified through several elements. Examples of properties that we aim to support in this work include both comprehensive ones, such as dependability, performance, cost and resource consumption, and more specific ones, such as worst-case execution time, development effort, class coupling, etc.

We, thus, defined a property *EFP* by a tuple $Property_{Type}$ so that:

$$Property_{Type} = \langle PropertyID, Name, Output, Unit, Description \rangle$$

where,

- *PropertyID* is a unique identifier for the property;
- *Name* is a human-readable name of the property (examples: development effort, reliability, performance, etc.);
- *Output* is the data type of the property (examples: integer, string);
- *Unit* is the standard unit of measure or data type for the output of the property (examples: person-month, second);
- *Description* describes the property in natural language. It must supply enough information to clarify the meaning of the property and its intended usage. Preferably supported by scientific references (Reference(s)).

4.3. Evaluation method

Evaluation methods are any approach used to give a value to a given property, be they estimation methods, measurements, mathematical calculations, prediction methods, model-checking, etc.

An evaluation method is defined by a tuple *Method* as follows:

$$Method = \langle MethodID, Name, Output, Unit, Applicability^{*}, Formula, Input^{*}, Driver^{*}, Description, Implementation^{*} \rangle$$

where,

- *MethodID* is a unique identifier for the evaluation method;
- *Name* is the name of the evaluation method used for the property (examples: Basic COCOMO 1);

- *Output* is the data type of the output of the evaluation method (examples: number, string);
- *Unit* is the unit of the result or output of the evaluation method (examples: Person-month, second);
- $Applicability^*$ specifies the elements for which the evaluation method is applicable (example: familiar projects, ambitious projects, tightly constrained/complex projects).
- *Formula* defines how the property is calculated by the evaluation method. It can include mathematical formulae, models, etc.;
- $Input^*$ is the set of inputs required by the evaluation method;
- $Driver^*$ is the set of contextual or environmental factors affecting the output of the evaluation method that are not directly involved in the calculation (using the formula). It can include project characteristics, calibration factors, etc.;
- *Description* is the description of the evaluation method including the information necessary to understand and use the evaluation method;
- $Implementation^*$ refers to available implementation, such as existing tools implementing the evaluation method.

In order to provide useful information to decision makers, the *Description* element is actually a tuple specified as follows:

$$Description = \langle Assumption^*, Advantage^*, Disadvantage^*, Source^* \rangle$$

where,

- $Assumption^*$ is the list of assumptions the evaluation method is based on;
- $Advantage^*$ is the advantages of the evaluation method;
- $Disadvantage^*$ is the disadvantages of the evaluation method;
- $Source^*$ is the list of references where the description, assumptions, disadvantages and advantages are based on.

4.4. The PMO Metamodel

According to Henderson-Sellers [34], foundational ontologies such as the PMO are equivalent in nature to the metamodel of a modelling language. Therefore, we also specify the PMO in terms of a metamodel to broaden its accessibility and extend its usage possibilities. This is a fairly common practice; previous works have introduced ontology-based metamodels for, e.g., software patterns [53] and agent-based simulations [54].

A metamodel is the set of modelling concepts, the relations among them and the rules by which the metamodel can be instantiated to achieve models conforming to it. Metamodels are at the core of any modelling language, being general-purpose or domain-specific. In this specific case, we define a metamodel that represents the basis for a domain-specific modelling language (DSML) for the description of property models, as this fits the nature of foundational ontologies.

Figure 2 shows the PMO metamodel specified in the Eclipse Modeling Framework (EMF[3]). The concepts specified in the formal definition are directly realised through corresponding entity classes, namely *Property, Method, Description, Input, Driver, Assumption, Advantage, Disadvantage, Implementation, Formula, Applicability*, and *Source*.

[3]https://www.eclipse.org/modeling/emf/

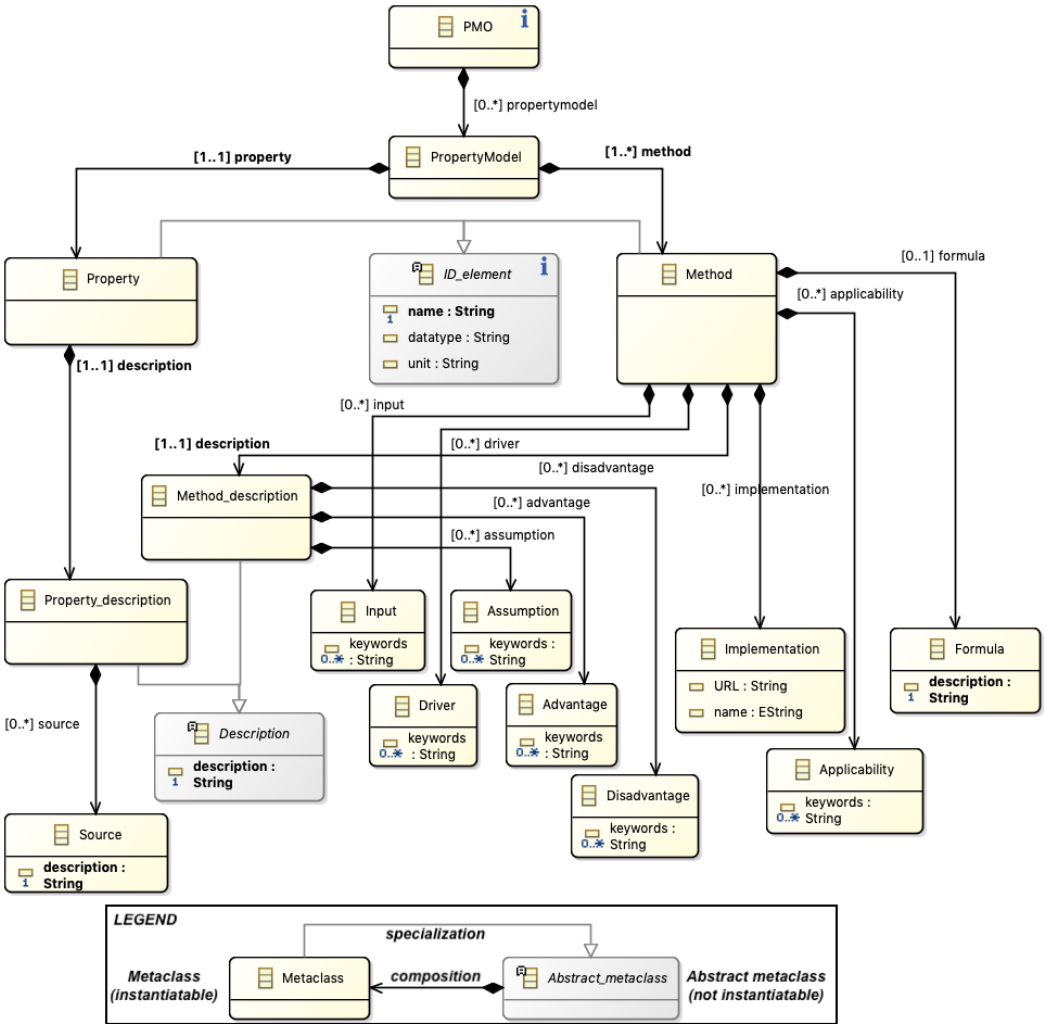


Figure 2. A metamodel in EMF for the PMO

Targeting instantiation, a metamodel offers the opportunity to refine the specification of the concepts through attributes and inheritance to be understandable by a computer[4]. As a result, *Property* and *Method* inherit attributes from the abstract class *ID_element*: *Name, Datatype* and *Unit*. Ideally, *Datatype* and *Unit* should be attributes of *PropertyModel*. However, given that (evaluation) *Method* does not necessarily comply with the output of the property we wanted to be able to record these possible divergences. *Name* is the unique identifier for properties and methods. To completely conform to the formal definition of the PMO, *Name* should have been split into 'ID' and 'Name'. However, the purpose of distinguishing the two is to allow two properties to have the same name, which is not regarded as a good modelling practice.

Property and *Method* contain a specialization of *Description*, namely *Property_description* and *Method_description*. Other entity classes, more specifically *Input*, *Driver, Assumption, Advantage, Disadvantage*, contains a list of keywords (in string format) for populating them. *Implementation* consists of a *Name* and a *URL*, a string attribute to

[4]Note that, although other formalization notations could be used, such RDF and OWL, we chose to go for an EMF metamodel since it naturally fits in our development tools ecosystem.

encode a url to easily locate corresponding tools for the related evaluation method. *Formula* is currently simply represented by a string attribute where to write a non-empty set of formulae, an algorithm or a software artifact related to the method. We envision a more complex datatype for representing formulae and algorithms. Therefore, we might need to further investigate how to better model this information.

Although the PMO itself in its formal definition could be seen as DSML, describing it in terms of a metamodel within the most commonly used open-source modelling platform for DSMLs definition, i.e., the Eclipse Modeling Framework (EMF), discloses the opportunity to leverage all related model-driven technologies for the manipulation of PMO models. Through model transformations, PMO models can be transformed into input formats for specific analysis tools, be integrated into multi-domain process models, rendered through a wide variety of different notations (e.g., graphical, textual, tabular, web-based), and in general made as integral part of model-driven development processes based on EMF.

The conformity of the metamodel with the formal definition of the PMO was evaluated by instantiating the property models from [55]. It was possible to encode all available data in the instantiation[5]

The purpose of the keyword is to provide a word summary of the descriptions in natural language contained in the element.

Practical examples: applying the ontology concepts in real systems.
To make the ontology concrete, we illustrate how its core concepts (*Property*, *Evaluation Method* and their attributes) are used in typical engineering scenarios. Each example maps a real decision to PMO fields; the methods correspond to those summarised in Table 4.

Example 1 – Automotive braking-by-wire latency budget (Performance)
$Property.Name$: End-to-end response time $Property.Output$: number
$Property.Unit$: ms
$Method.Name$: End-to-end response-time analysis over CAN/fixed-priority tasks
$Method.Formula$: combine task/message worst-case terms (e.g., $R_i = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$) and CAN arbitration delays; see the schedulability and WCET methods in Table 4.
$Method.Input$: task periods and priorities, WCETs (from IPET/measurement), CAN message sizes and IDs, bus speed.
$Method.Driver$: bus load, scheduling policy (preemption), jitter bounds.
$Method.Assumption$: fixed-priority scheduling; bounded blocking; stable configuration.
Decision use: accept if $R_{\text{brake-chain}} \leq 20\,\text{ms}$ under 90% bus load; otherwise revise priorities or split messages.

Example 2 – OTA update reliability qualification (Reliability)
$Property.Name$: Failure intensity during system test $Property.Output$: number
$Property.Unit$: failures/hour
$Method.Name$: Musa-Okumoto reliability growth model

[5]For the interested reader, the PMO metamodel and a sample model can respectively be found at http://www.mrtc.mdh.se/promopedia/sites/default/files/PMO_metamodel.zip and http://www.mrtc.mdh.se/promopedia/sites/default/files/PMO_model_instances.zip.

$Method.Formula$: $m(t) = \frac{1}{\theta}\ln(1+\lambda_0\theta t)$, $\lambda(t) = \frac{\lambda_0}{1+\lambda_0\theta t}$.
$Method.Input$: execution time on operational profile, failure occurrence times.
$Method.Driver$: realism of operational profile; test environment fidelity.
$Method.Assumption$: independent defect removal; stable debugging process.
Decision use: proceed to pilot if $\lambda(t_{\text{end}}) \leq 0.05$ failures/hour with 95% CI; else extend testing focusing on high-usage features.

Example 3 – Feature sizing for a platform increment (Development Effort/Cost)
$Property.Name$: Development Effort $Property.Output$: number
$Property.Unit$: person-months
$Method.Name$: Basic COCOMO I (or Post-Architecture COCOMO II)
$Method.Formula$: $E = a \cdot \text{KLOC}^b$ (mode-dependent constants); Post-Arch adds effort multipliers.
$Method.Input$: estimated KLOC (or function points), project mode, cost drivers.
$Method.Driver$: team capability, tool maturity, required reliability.
$Method.Assumption$: calibrated coefficients for the organisation; 152h/person-month.
Decision use: if $E_{\text{in-house}}$ exceeds threshold, evaluate *buy/adapt* alternatives; track variance in a release review.

These use case examples show how practitioners take a concrete goal (e.g., meeting a latency or reliability target), select a $Property$ with clear $Output$ and $Unit$, choose an $Evaluation\ Method$ with explicit $Inputs$, $Drivers$, and $Assumptions$, and then apply an acceptance criterion. Embedding these artefacts in the PMO enables consistent recording, comparison, and reuse across projects.

5. Evaluation of the property model ontology

We first present the research method used for evaluation of the PMO (Section 5.1), then we describe the threats to validity (Section 5.2 , followed by the evaluation results (Section 5.3).

5.1. Method

According to [56], there is no single approach to ontology evaluation and the choice depends on the purpose, the application domain of the ontology, and the aspects to evaluate. Burton-Jones [57] describes qualities of ontologies, as well as proposing measurements for the evaluation of the ontology. The main categories of qualities are:

- *Social quality:* Social quality refers to the degree by which the ontology can be relied on and how often it has been used.
- *Pragmatic quality:* This quality focuses on the comprehensiveness/completeness and relevance of the information for the purpose it should be used. In other words, pragmatic quality focuses on determining whether the provided information is useful. It is also strongly linked with expressiveness (cf. [40]), which indicates that the ontology should have the capability to express comprehensive information.

– *Semantic quality:* Semantic quality focuses on whether the terms used in the ontology are meaningful, consistent and clear. Hence, this maps to clarity as defined by [40].
– *Syntactic quality:* Here the focus is on lawfulness (correct syntax) and richness (the complexity of the syntax).

Given that the ontology is relatively new and has not been widely spread in practice and research, we are not able to evaluate its social quality. Syntactic quality is neither in the focus because it is not feasible to check correctness and richness of an ontology that is only instantiated and not widely used in practice yet. Thus, we focus on pragmatic quality as well as semantic quality. Consequently, the following research question is formulated for the validation of the PMO: *To what degree does the ontology achieve pragmatic and semantic quality?*
Despite its first publication in 2016, uptake outside our own studies has remained limited; accordingly, we consider the PMO's social quality (use and reliance by a broader community) to be low at present. We therefore emphasise pragmatic and semantic quality in this evaluation and return to adoption barriers and mitigation actions in Section 6.

The qualities were first evaluated from the academic perspective (referred to as academic-driven) to identify potential flaws before involving subjects from industry who provided feedback from the industrial perspective (referred to as expert-driven). The research process is shown in Figure 3.

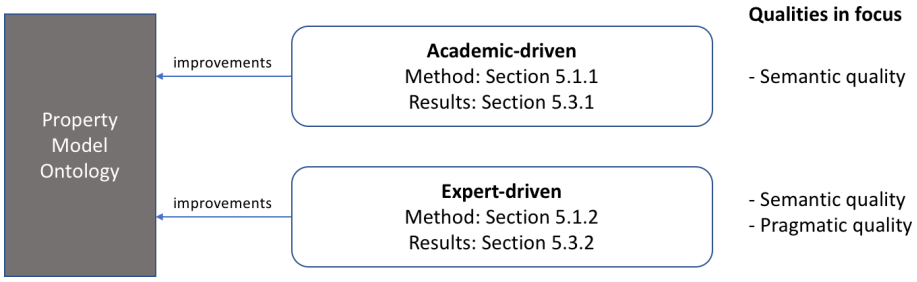


Figure 3. PMO qualities in focus per academic and expert evaluations

5.1.1. Academic-driven evaluation

Prior to evaluating a solution in an industrial context it is recommended to first carry out an evaluation in an academic environment [58]. As the focus during this step is on eliciting the information from the literature, it is not feasible to gather insights from industry about the usefulness of the ontology. Hence, the academic evaluation is focusing on ensuring semantic quality prior to the expert-driven evaluation. We conducted the academic-driven evaluation using the following activities:
1. Identify relevant properties for a selected domain.
2. Identify relevant evaluation methods for the previously identified properties.
3. Classify properties and evaluation methods based on the ontology.
4. Update the ontology based on what has been learned.

Activity 1. Identify relevant properties for a selected domain. A first activity towards evaluating the ontology was the identification of relevant properties and evaluation methods for these properties. As the proposed ontology was developed in the context of the research project ORION the initial emphasis was on identifying properties within the automotive

domain, which was in the focus of the project. Priority was given to these properties which were most frequently mentioned in the literature for the automotive domain [9].

Activity 2. Identify relevant evaluation methods for the previously identified properties. After a set of relevant properties was identified, we gathered three additional papers for the most frequently mentioned properties in the automotive literature and collected descriptions of their evaluation methods. The focus was on identifying real industrial cases, in at least one domain (here automotive), where evaluation methods are identified and discussed. The identification of such cases would enable us to obtain and reuse domain knowledge. Also, as our ontology is a model of the real world, so the concepts defined in it must reflect reality, and thus it is important when we defined an initial version of the ontology, to evaluate it using a real application or problem as suggested in [50]. Therefore, in our evaluation we followed the suggestions of Noy and McGuinness [50], and took into account real cases of considering EFPs in an actual domain, i.e., the automotive domain.

Activity 3. Instantiate the ontology for different properties and evaluation methods. For this purpose two iterations were realized to record data:

– Iteration 1: Instantiation with well-known properties and evaluation methods relevant to the automotive domain
– Iteration 2: Instantiation with security properties and their evaluation methods [59].

For this purpose two instruments were created to record data: a text document for iteration 1, where the data could be easily extracted and comments could be added, and a spreadsheet for Iteration 2.

Activity 4. Update the ontology based on what has been learned. The comments from Activity 3 were discussed among the researchers and a rejoinder was created to decide how to address the concerns raised.

Table 1 provides a brief overview of the researchers participating in the evaluation.

Table 1. Evaluation participants (researchers)

Id	Position	Research Topics
R#1	Assoc. Prof.	Component-based software engineering, model-based development, embedded systems
R#2	Senior researcher	Cost estimation, software architecture, agile software development
R#3	Prof.	Software processes, software metrics, empirical software engineering
R#4	Senior researcher	Software quality, enterprise architecture, cyber security

The revised ontology then was used as input for the interviews with the practitioners (expert-driven evaluation).

5.1.2. Expert-driven evaluation

The evaluation was completed with interviews of practitioners who have spent at least five years working in industry. Their demographic data collected is shown in Table 2. The partial elements filled-in the table are due to inability to collect all the data during the interviews due to technical failure of the system used for the survey. Interview from I#9 was also not carried out and thus is removed from the sample. The interviewees from industry were asked to reply to questions of an online survey[6], whereas one researcher was observing the process. The researcher was taking notes of any additional information or reflections that interviewees mentioned during the process of entering information.

[6]The survey elicitation form is available here: https://goo.gl/forms/6OfIjo4K1mfaypsR2.

Table 2. Interviewed experts from industry

Id	Position	Domain	Property	Evaluation Method
I#1	Tester/Developer	Telecom	Reliability	*[None]*
I#2	Product Manager/ Tester/Developer	Telecom	Maintainability	Peer review
I#3	QA	Transport	Safety	Safety standards assessment
I#4	Consultant/Researcher	IT consulting	Performance	Timing analysis
I#5	QA/Developer	Fin-Tech	Timing	Critical path response time
I#6	Researcher	IT consulting	Timing	Response time analysis
I#7	Tester/Researcher	RT systems	–	-
I#8	Researcher	Automotive	Timing	End-to-end timing analysis
I#10	Consultant/Researcher Researcher	IT	Development effort	Regression analysis
I#11	Consultant/ Researcher	IT	Development effort	Function points or objective points

In the beginning of the online survey, some general information was given about the survey, and the interviewees were asked to provide information of a specific (but of their choice) property model, evaluation method related to the chosen property and finally, general feedback. The feedback related to evaluation of the semantic quality of the elements and their evaluation methods, including how informative they are. The feedback also related to evaluation of pragmatic quality (i.e., focusing on usefulness) of the PMO. Pragmatic quality included utility scenarios in practice. Moreover, interviewees were asked to identify unclear and/or missing elements. During the data collection, as well as in the feedback part of the survey, we were able to collect change requests to specific PMO concepts and the survey itself, to help us improve the PMO.

The change requests helped in evaluating both the semantic and pragmatic quality of the ontology. The number of changes requested is an indication for the quality of the ontology. If the number of changes is high then the ontology needs to be further revised to improve its quality, both semantically and pragmatically. On the other hand, if the definition of an element needs to be changed for the element to become clear and meaningful, then the ontology needs to improve on semantic quality.

Overall, we counted the number of changes and classified them into four groups, namely structural, definition, example and reformulation, which are defined as follows:

- *Structural:* The structure of the ontology changes. New elements need to be added or existing elements need to be removed. It also includes changes where one element needs to be split. Structural changes are primarily linked to pragmatic quality as information needs to be added or removed to increase the relevance of the ontology.
- *Definition:* The definition (i.e., description) of an element of the ontology has to be changed for the element to become meaningful. This concerns semantic quality.
- *Example:* Together with the definition of the elements, an example was given. The example used for the elements of the ontology needs to be changed. Examples help to improve the understanding and hence clarity of the ontology (semantic quality).
- *Reformulation:* An element needed to be reformulated, though its meaning (definition) has not changed. Reformulations are mainly focused on increasing clarity (semantic quality).

The above list shows the change types according to their complexity and gravity, from high to low; i.e., a structural change requires to make more significant changes to the

ontology structure and probably rework multiple elements, whereas a reformulation can be more simple, involving cosmetic changes and does neither change structure nor meaning of elements.

5.2. Validity threats

The threats of validity were identified and evaluated during the research, based on the proposals by Maxwell [60] and the derived classification of threats by Petersen and Gencel [61], with a particular focus on qualitative research.

5.2.1. Descriptive validity

Descriptive validity refers to our ability to describe what is happening accurately. In the context of the study it is important to capture the interviews accurately. This was supported through the online survey where practitioners recorded their answers and also provided their comments. To avoid misunderstandings during the completion of the interview it was conducted face-to-face. Hence, the threat is considered under control.

5.2.2. Theoretical validity

Theoretical validity is threatened if we cannot capture what we intend to capture, e.g., due to poorly defined constructs or confounding factors. During the evaluation in the academic context members of the ORION project took the roles of the evaluators. Hence, they are familiar with the ontology and hence lack an external perspective. The evaluation, however, adds confidence prior to using the ontology externally with practitioners. We also did not only rely on the academic evaluation, but complemented it with the perspective of practitioners.

5.2.3. Generalizability (internal and external)

Internal generalizability refers to the ability to generalize within a context and external generalizability to other contexts. During the academic evaluation the target was a particular industrial domain, the automotive domain. Hence, a restricted set of properties and evaluation methods was in the focus. Within that domain, a range of methods has been elicited and documented. Hence, the internal generalizability is considered high. However, the external validity of the study is limited. The threat was to some degree reduced by complementing the work with practitioners from different domains (e.g., Telecom, see Table 2).

5.2.4. Interpretive validity

Interpretive validity is concerned with drawing valid conclusions from the data. A potential threat are the biases of researchers during the interpretation and analysis of the data. To reduce the threat we utilized observer triangulation during the academic evaluation and the practitioner interviews. In decisions and interpretations all authors were involved. Also, all comments, responses and decisions were recorded in a rejoinder, making the design decisions for updating the ontology traceable.

5.3. Results

We first present the results of the evaluation in the academic context, followed by the results of the evaluation with industry practitioners.

5.3.1. Evaluation in the academic environment

We show the results of each individual step of the evaluation in the academic environment. The first two steps (identify relevant properties and evaluation methods) of our research have already been presented in [9], while the systematic analysis of the feedback and subsequent updates to the ontology have been added in this study as a preparation for the industrial evaluation. We shortly repeat the main findings of our previous paper to ensure the traceability of the research steps.

Activity 1. Identify relevant properties for a selected domain. We identified nine properties and mentioned the related papers in which the properties have been presented. We also saw that specific properties have been present in most of the research papers. Table 3 summarizes the results of this study, whereas more details can be found in [9]. Numbers in the table highlight the priorities among the EFPs identified as relevant in the referenced literature, whereas a "+" symbol means that the property was identified as important but no explicit ranking was provided.

Table 3. Relevant EFPs for the automotive domain

Properties	[62]	[63] Scania	[63] Volvo	[64]	[65]	[66]	[67]	[68]	[69]	Final Ranking
Cost	1	1	1	+		+	+	+	+	#1
Performance	2	2	3	+	+		+	+		#2
Configurability, Variability	3			+						
Flexibility, Maintainability, Testability	4	6		+						
Power Consumption	5		2						+	
Reliability, Safety		3, 5	4	+	+	+	+	+	+	#3
Evolvability		4								
Security							+			

Activity 2. Identify relevant evaluation methods for the previously identified properties. For each property identified in the previous step, several evaluation methods have been extracted (in total 19 evaluation methods) as shown in Table 4. The researchers extracting which models are shown in the same table.

Activity 3. Classify properties and evaluation methods based on the ontology. The researchers extracted the property descriptions and corresponding evaluation methods (summarised in Table 4, also detailed in [55]).

To illustrate how the ontology is instantiated Table 5 presents three evaluation methods for development effort, namely the Basic COCOMO I, Estimation by Analogy (EbA) and Expert Estimation. Each model is described with its attributes and corresponding values. For example, the Basic COCOMO I is applicable for in-house development, EbA requires the availability of data from similar projects and expert estimates require experienced practitioners (values). The ontology also describes the method narratively (description) and provides the formula(s) to be used. Knowing the inputs may also provide valuable insights on

Table 4. Researchers, properties and elicited evaluation methods

Id	Property	Evaluation Method
R#1	Performance	Controller Area Network (CAN) schedulability analysis, End-to-end response time (or delay) analysis, Worst-case execution time (WCET) analysis based on Implicit Path Enumeration Technique (IPET), Measurement-based WCET analysis, Probabilistic Hybrid WCET Analysis, Probabilistic Worst-case Response-Time Analysis for CAN, Worst-Case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption
R#2	Cost	Basic COCOMO 1, Intermediate COCOMO 1, Advanced, Detailed COCOMO 1, Expert Estimation
R#3	Reliability	Logistic model, Inflection-S model, Musa-Okumoto, Jelinski-Moranda
R#4	Availability	Steady state availability analysis based on MTTF and MTTR, Component-based availability analysis, Markov state space models

whether we are able to apply the model (e.g., the ability to determine the required inputs). The drivers affect the output of the model beyond the formula and hence are confounding factors. For COCOMO these factors are explicitly defined and classifiable (development mode, application domain, project characteristics), while for expert estimation the drivers are mentioned, but do not have a classification/ measurement scale. Assumptions of using the methods (in particular with regards to statistical approaches) are important to not be violated, as it may result to invalid outcomes. Moreover, the advantages and disadvantages are stated. Finally, the source and implementations of the method are listed.

Based on the evaluation forms the four researchers provided their reflections using the data collection form. The reflections focused on the **semantic quality** of the ontology and improvements were made based on the feedback collected.

Overall, the PMO was instantiated with seven properties (development effort, development time, cost, performance, reliability, availability and security) with more than 40 evaluation methods (e.g., Post-Architecture COCOMO II, CAN schedulability and Worst-Case Execution Time), details of which can be found in [55].

Activity 4. Update the ontology based on what has been learned. For each comment raised by the reviewers a response was written and a decision was taken among the authors how to address the issue.

Hence, prior to the evaluation with the practitioners, the internal evaluation did not lead to major changes in the ontology, but to several improvements towards clarity. This also provided confidence in the ontology as a variety of evaluation methods was classified using it by different persons.

5.3.2. Result from interviews with practitioners

As mentioned in Section 3 the feedback during the evaluation was considered for the identification of concepts and revising their relationships and their attributes. Table 6 shows the number and type of changes collected from the feedback per each PMO concept. In total 28 change requests were made, a small figure compared to the vast majority of the concepts receiving no change request (82.5%). The change requests were implemented after an agreement process followed by the authors of the paper which included reviewing, discussing, suggesting changes and agreeing on the changes proposed, until no concepts were left to be agreed and all proposed changes were agreed.

Semantic quality. The evaluation of the presentation of the PMO in terms of informativeness, i.e., whether it served its purpose it was created for, was positive by the

Table 5. Excerpt of the property model for development effort

Property Name*	Development Effort		
MethodID	Basic_Cocomo_I	Analogy_Estimate	Expert_Estimate
Name	Basic Cocomo I	Estimation by Analogy (EbA)	Expert Estimation
Output	number	number	number
Unit	Person-month	Person-month	Person-month
Applicability	Familiar projects, ambitious projects, tightly constrained/complex projects, in-house development	Anytime, similar data is available	Similar projects, expertise and experience, to any case
Formula	$a(KLOC)^b$, where a and b are two constants which are set based on the development mode.	N/A	N/A
Input	– KLOC (thousands lines of code)	N/A	– Expert's value estimate – Expert's confidence, experience, skill – Expert's weights – Expert's background – Available checklist of information related to the project
Driver	– Development mode (organic, semi-detached, embedded) – Application domain – Project characteristics: Size, Innovation, Deadline, Dev. Environment	N/A	– Expert's experience and knowledge – Gut-feeling
Description	The method computes software development effort as a function of program size expressed in estimated kilo-line of code	The method estimates the property by analogy with similar completed projects in the same application domain	The method uses a person recognized as an expert on the task to estimate the value. Typically, it relies on tacit knowledge.
Assumption	– Configured for 152hours/ Person-month.	– Prior knowledge exists (stored or can be accessed).	– Expert is available
Advantage	– Transparency – Good for quick, early and rough estimate – Drivers provide insights to the decision maker on what affect the project	– Clear mathematical reasoning – Provides opportunities for weighing and obtaining local and global measures	– It is one of the dominant strategies when estimating software development effort – Simple to understand – Does not rely on size metrics (e.g. KLOC) to provide the estimate
Disadvantage	– Limited accuracy. – Difficult to estimate KLOC early in the project. – Vulnerable to the quality of model tuning.	– Similar projects exist and were documented. – A priori selection of the most important attributes.	– How the estimation has been performed remains typically undocumented. – Subjective to the expert's opinion and bias.
Source	[70]	[71]	[72]
Implementation	N/A	N/A	N/A

*For simplicity the Property details are omitted as they match the *Output, Unit* of the Method

Table 6. Change requests for each PMO concept

PMO concepts	Structural	Definition	Example	Reformulation	No change
Property.Name	1	1	1	1	7
Property.Output & Property.Unit	0	0	0	0	9
Property.Description	0	0	0	0	10
Method.Name	1	2	0	0	8
Method.Output	1	4	0	1	4
Method.Unit	0	0	0	0	9
Method.Applicability	0	5	1	0	4
Method.Formula	0	0	0	0	10
Method.Input	0	1	0	2	7
Method.Driver	0	2	0	0	8
Method.Description	0	0	1	1	8
Method.Assumption	0	0	0	1	9
Method.Advantage	0	0	0	0	10
Method.Disadvantage	0	0	0	0	10
Method.Source	0	0	0	1	9
Method.Implementation	0	0	0	0	10
Total	3	15	3	7	132

interviewees (in 8 out of 10 cases). One case (out of the 8) commented that it was not fully informative what the aim of the PMO was or which properties were mostly important (I#11). Regarding understandability of the property and evaluation methods concepts a likert-scale was given to the interviewees from 1–5 (1: strongly agree, 5: strongly disagree). On average (and mean) understandability was ranked at 2 (by 8 interviewees, as values were not available from all questionnaires), indicating partially high understanding.

Another respondent found some of the standards/patterns in the options a little unclear (I#2). Another request was for more elaborate baseline descriptions (i.e., on the aims of the PMO and their evaluation methods) as well as allowing more possible ways for filling-in responses (I#11). The same interviewee (I#11) also commented not liking to have to write so much, something that highlights the need for more elaborate property and evaluation methods descriptions to be made available. I#6 commented on how the questions were posed in the online form and didn't really understand the need for using such elicitation method for collecting data. I#10 commented on the use of the word "property" not reflecting well the term "variable" typically used in estimation, analysis and assessment contexts. However, the respondent was able to overcome this issue through the examples given throughout the questionnaire, commenting that they were very helpful, made the questions clearer and less ambiguous. Overall, in order to improve understandability, interviewees requested for 3 structural, 15 definition, 3 example changes and 7 reformulations of the questions (see Table 6) and these were implemented by the authors.

Pragmatic quality. The pragmatic quality of the PMO was evaluated by requesting some envisioned scenarios of use. I#10 would use them for "comparison of evaluation methods or for combining the output of different evaluation methods" to prioritise, for example, needs. I#11 was more specific in the PMO usage as "an analysis of effort estimation methods with around 15 characteristics, for example as the one found in Pfleeger et al. [73].

Along the same lines of thinking, I#1 found the PMO useful if it "can provide guidelines based on the properties, e.g., what mathematical models to use an when" which is a good distinctive value offered by the PMO, as identified by its receivers.

I#4 and I#7 found the PMO useful in several scenarios and especially as a reference or as a reference guide to definitions of properties and their evaluation methods. I#8

found it useful and helpful for their automotive customers. I#2 explains that if PMO "reduced review time and helped out in the automation of the review (i.e., according to pre-defined standards) then it would be really useful." I#3 said "it sets the scope and gives and understandable frame" and I#5 explained "it can help two individuals with a different understanding [of a property] to reach to a common language. A catalog of properties is something I often looked for my PhD studies.", whereas I#6 also agreed "it can be very useful especially for finding agreement in case of discordance."

Interviewees were asked for any information or concepts missing from the PMO. I#8 suggested that the context of the application domain might be important when discussing properties and I#4 the stage of development at which the evaluation method is applied/applicable. These concepts are however already included in the PMO, under *Method.Applicability*. I#1 suggested that we consider just the context of a team and think of ways to include properties related to team factors, such as performance and skills. I#10 suggested measures of accuracy as a concept missing from the PMO which could be a way for comparing the results of different evaluation methods (e.g., mean absolute error, or mean relative error). I#11 considered as nice to have a method to judge or provide input on a pre-made analysis of evaluation methods. It would also be good if expert-based methods were also described. The rest of the interviewees did not have anything more to add to the PMO.

6. Discussion

6.1. Summary of key findings

The impact of the academic evaluation, mostly targeting **semantic quality** of the ontology was rather low, as it did not lead to major changes to its structure or definition. The subject domain was representative (based on the feedback collected) and the meaningfulness was found high, as very few changes to improve clarity were proposed. The improvements mentioned were about more well-defined elements. Both the classification of the fields as optional or mandatory and the use of short examples, as requested by the researchers, addressed the issues of clarity and consistency. Therefore, the semantic quality of the PMO was found as satisfactory to proceed to the industrial evaluation.

The industrial evaluation resulted to a small number of changes (only 8% vs. 92% of no change requests) showing a high acceptance rate with regards to semantic quality of the PMO by the practitioners. Understandability was ranked on average as high. A few improvements that were requested concerned the level of elaboration for the elements (i.e., related also to consistency), something that is a concern in all crowd-sourced material, although we attempt to address it through the use of examples. Overall, the industrial experts were positive towards our approach to the definition of properties and evaluation methods, which provided us the confidence to proceed and develop PROMOpedia for subsequent evaluation of pragmatic quality.

6.2. Reflections on the quality of the ontology

Concerning the **pragmatic quality** of the PMO, the results of the interviews consolidated our idea of the potential usefulness of the PMO for data extraction purposes when carrying out systematic studies. The PMO was in fact perceived by interviewees as a reference guide

to definitions of concepts, a way to properly set scope and frame of specific aspects under study, and in general a way for individuals with different understandings and coming from different domains to reach a common language. The purpose of systematic studies is to present a body of knowledge in an unbiased and objective manner, and the PMO could be a useful tool to report on software-related properties. We evaluated this by employing the PMO to systematically investigate and report how software security-related properties are evaluated [52]. The PMO was used as backbone of the data extraction phase of our systematic study and proved itself to be a very useful tool in this respect, as initially presumed. Moreover, we perceived that the way the PMO is structured helped us in methodically retrieving information in an efficient manner although none of the authors had previous experience with the specific set of properties (related to security) under study.

With respect to improvements in the PMO structure, changes to the structure and to the definitions were carried out based on the feedback collected. These changes materialised improvement in the PMO, however still some issues remain. Ideally, the evaluation methods' unit and output should match the ones of the corresponding properties. However, this is often not the case. Therefore, we opt to record both of them in order to enable identifying possible mismatch in the state-of-the-art and practice. There's an implicit relation of the applicability and drivers used in the PMO with the context of where an evaluation method should or could be used. However, our work focused on implementing and evaluating the ontology and has not investigated in depth this relation.

6.3. Reflections on industry adoption

Adoption to date and barriers. While the academic and practitioner evaluations indicate that the PMO is understandable and useful as a structuring device, *its social traction has been modest since 2016*. In our experience, four factors have hindered broader uptake: (1) *On-ramp cost*-encoding properties and evaluation methods demands time and ontology literacy with unclear short-term payoff; (2) *Terminology fragmentation*-adjacent communities (e.g., ISO/IEC 25010 quality models, software measurement, safety/security) already use entrenched vocabularies, creating perceived overlap; (3) *Toolchain fit*-lack of out-of-the-box integration with requirements and modelling tools reduces everyday utility; (4) *Evidence gap*-few publicly available, end-to-end exemplars that show PMO accelerating decisions or reducing defects.

Actions taken to increase adoption. We have pursued several concrete steps: (i) *Open availability* via PROMOpedia with ready-to-use templates to minimise authoring overhead; (ii) *Executable artefacts* by instantiating the PMO as a metamodel in a modelling environment, easing reuse in projects; (iii) *Worked exemplars* beyond automotive (e.g., security-related properties) to demonstrate cross-domain applicability; (iv) *Process embedding* by trialling PMO templates in internal courses and project reviews. *Ongoing efforts* include packaging a small "PMO-core" subset for quick starts, providing machine-readable exports (e.g., JSON/OWL) and mapping guides to adjacent standards to reduce cognitive friction, and exploring light-weight integrations with requirements/issue trackers to meet practitioners where they work.

6.4. Comparison with related work

In relation to the concept of "property model" several scientific works exist which define it, i.e., [70, 74–79]. They point to interesting aspects of the concept but they do not touch

upon our vision and use of the concept of property model as a key factor to evaluate the suitability of a component from an extra-functional property viewpoint. For example, [77] only focuses on functional property models. In [74], the concept refers to a property only viewed as a constraint that can be verified whereas in [75–77] it is used to characterise a system through its perceived properties. In [78], the concept of property model is meant to be a formalisation of a set of properties (resources, mechanisms, policies) exploited at deployment time for the various deployment participants to share the same property semantics. Besides, the examples of property models used in [75] mix properties, evaluation models, metrics for properties.

Our definition of property and methods is deliberately broad and inclusive to encompass the different terminologies related to extra-functional properties and enable capturing their nuances. For example, in the case of worst-case execution time, two properties can be defined using the formal languange speficied in Section 4: one which focuses on the time as the data to be processed (i.e., < "WCET", "Worst-case execution time", "Real number", "s">, [80]) and one on the processor's clock cycle (i.e., < "WCET_2", "Worst-case execution time", "Real number", "clock cycle", [80]>). This enables easily associating the right evaluation methods to the right property and limit incompatibility risks. Furthermore, for many properties, many different definitions exist, often highlighting specific details. Again, for worst-case execution time, the following definitions can be found: *"The worst-case execution time (WCET) is the longest time it takes to execute a given program code* [81]," *"The worst-case execution time (WCET) is defined as the longest execution time of a program that could ever be observed when the program is run on its target hardware* [82]." As shown with both examples, i.e., with the differences in data and the differences in definitions, this has actual implication on the evaluation methods and the amount of information necessary to calculate the property and its validity in different context.

Being inclusive can potentially lead to information bloat. However, given that one of our primary goal with this research is to provide a starting point to unravel the state-of-the-art on software properties, listing up all available definitions is an important necessary step. It would allow identifying commonalities, differences and provide a common ground for fostering discussions.

7. Conclusion

In this work we presented in detail the Property Model Ontology (PMO) and how it was devised. We described its evaluation together with experts and exemplified its use. The evaluation carried out included semantic and pragmatic quality of the PMO and we conclude that: (a) the steps carried out towards an evaluation together with experts is a difficult activity to carry out, especially for a new ontology, having lack of wide and effective usage by practitioners; however it is a mandatory and important step to structure the knowledge on software quality properties and remove the obstacles that obfuscate the state-of-the-art; (b) by focusing on the assessment of pragmatic and semantic quality of an ontology we need, as next step, to assess social and syntactic quality to obtain an even more generalizable and complete evaluation.

Overall, we have achieved a consensus from the experts, on how to capture EFPs using the concepts, attributes, and relations in the PMO, consensus was reached on the concepts in the PMO and a shared understanding on its possible uses in practice. The definition of the PMO, provided in this paper, offers a framework so that individuals and organizations

may define their own property models and describe the related evaluation methods. They can form the preconditions for assessing quality in the development of software-intensive systems. While doing so, however, it is up to each individual or organization to define in precise detail and values of their own model according also to their scope. For instance, one usage may be to select assets from multiple alternatives based on specific values or range of values for EFPs which are considered acceptable by the organization.

The PMO is an ontology which aims to represent the subject domain with as much truthfulness, clarity and expressivity as possible, We support that it can result to increased expressivity of knowledge, and clarity of negotiations for practitioners. Moreover, based on the classification proposed in [49], the PMO also belongs to the frames category. However, by fostering its use through PROMOpedia, the online encyclopedia of property models for software engineering [83], our long term objective is to be able to identify set of value constraints for the different entities, hence transforming the PMO to an ontology with value constraints. The PMO can also be used for representing a domain body of knowledge, resulting from systematic studies for individuals with different understandings of properties and their evaluation methods to reach to a common language and improved clarity.

Recent rise in Artificial Intelligence (AI) and in particular GenAI (Generative AI), offer significant potential to support and enhance the development and application of application and domain-oriented PMOs. However, the lack of large dedicated datasets from multiple sectors and industries inhibits this and to fully reaping the benefits. Advances in Domain-Specific Languages (DSLs), may assist in automating parts of the PMO instantiation process for specific domains, where the relevant knowledge is available and well-documented. For example, GenAI can assist in generating tailored DSLs that encapsulate the key concepts, relationships, and value constraints defined within a PMO. This approach can facilitate in the creation of expressive, user-friendly languages for practitioners, enabling them to interact with and extending PMO-based models more efficiently. AI/GenAI can also assist in maintaining consistency and clarity between different PMOs instantiations by working with terminology and generating supporting tools, like chatbots, parsers and validators.

As future work, we plan to use the ontology and investigate further its expressivity capabilities, content clarity and quality, as well as identify means to enroll the research, academic and practitioner communities to foster its metamodel and instantiations through for instance, PROMOpedia. Future work will focus on concrete evidence of the PMO's practical utility and adoption, with comprehensive end-to-end case studies. A proposed approach could be employing A/B testing and comparing results between practitioners using PMO and not using it, to measure improvements in decision quality, speed and required investment effort. This approach will allow the generation of empirial data on the benefits and adoption barriers of the PMO is real-world conditions. Furthermore, we will seek to engage the research, academic, and practitioner communities in evolving the PMO metamodel and its instantiations through the PROMOpedia platform. Community involvement will be essential in refining the PMO's usability and ensuring that it meets the practical needs of its users.

Acknowledgment

Acknowledgements to our colleagues in the ORION project and the participants in the evaluation for the fruitful discussions and contributing in improving the work. We especially

acknowledge the contributions and inspirations from the late Séverine Sentilles throughout this work.

CRediT authorship contribution statement

E Papatheocharous: Conceptualization, data curation, formal analysis, investigation, methodology, software, validation, visualization, writing – original draft, writing – review and editing. S. Sentilles: Conceptualization, data curation, formal analysis, investigation, methodology, software, validation, visualization, writing – original draft, writing – review and editing. K. Petersen: Data curation, formal analysis, investigation, methodology, validation, writing – original draft, writing – review and editing. F. Ciccozzi: Conceptualization, data curation, formal analysis, investigation, methodology, software, validation, visualization, writing – original draft, writing – review and editing.

Declaration of competing interest

There are no competing interests.

Funding

The work is supported by a research grant for the ORION project (reference number 20140218) from The Knowledge Foundation in Sweden.

References

[1] N. Schneidewind, *Standard for a software quality metrics methodology*, IEEE Std. 1061-1998, 1998.
[2] T. Olsson, S. Sentilles, and E. Papatheocharous, "A systematic literature review of empirical research on quality requirements," *Requirements Engineering*, Vol. 27, No. 2, 2022, pp. 249–271.
[3] J. Mylopoulos and L. Chung, "Representing and reasoning with non-functional requirements: A retrospective," *IEEE Transactions on Software Engineering*, 2025.
[4] L. Chung, B.A. Nixon, E. Yu, and J. Mylopoulos, *Non-functional requirements in software engineering*, Vol. 5. Springer Science and Business Media, 2012.
[5] B. Regnell, R.B. Svensson, and T. Olsson, "Supporting roadmapping of quality requirements," *IEEE Software*, Vol. 25, No. 2, 2008, pp. 42–47.
[6] E. Yu, "Towards modelling and reasoning support for early-phase requirements engineering," in *International Symposium on Requirements Engineering*, 1997, pp. 226–235.
[7] X. Franch, J.C.S. do Prado Leite, G. Mussbacher, J. Mylopoulos, and A. Perini, *Social Modeling Using the i* Framework*. Springer, 2024.
[8] I.G. Ndukwe, S.A. Licorish, A. Tahir, and S.G. MacDonell, "How have views on software quality differed over time? Research and practice viewpoints," *Journal of Systems and Software*, Vol. 195, 2023, p. 111524.
[9] S. Sentilles, E. Papatheocharous, F. Ciccozzi, and K. Petersen, "A property model ontology," in *Software Engineering and Advanced Applications (SEAA)*. IEEE, 2016, pp. 165–172.
[10] S. Sentilles, *Managing Extra-Functional Properties in Component-Based Development of Embedded Systems*, Ph.D. dissertation, Mälardalen University, Västerås, Sweden, 2012.

[11] *Software engineering – Product quality*, ISO/IEC Std. 9126, 2001.
[12] S.T. Albin, *The art of software architecture: design methods and techniques*, Vol. 9. John Wiley and Sons, 2003.
[13] J.A. Mccall, P.K. Richards, and G.F. Walters, "Factors in Software Quality. Volume I. Concepts and Definitions of Software Quality." GENERAL ELECTRIC CO SUNNYVALE CALIF, Technical Report ADA049014, 1977. [Online]. http://stinet.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA049014
[14] B.W. Boehm, J.R. Brown, and M. Lipow, "Quantitative evaluation of software quality," in *Proceedings of the 2nd international conference on Software engineering*, ICSE '76. Los Alamitos, CA, USA: IEEE Computer Society Press, 1976, pp. 592–605. [Online]. http://dl.acm.org/citation.cfm?id=800253.807736
[15] *Systems and Software Engineering. Systems and Software Quality Requirements and Evaluation (SQuaRE). Product Quality Model*, ISO/IEC Std. 25 010:2023.
[16] J.C. Laprie, "Dependable computing and fault-tolerance," *Digest of Papers FTCS-15*, 1985, pp. 2–11.
[17] J. de AG Saraiva, M.S. De França, S.C. Soares, J. Fernando Filho, and R.M. de Souza, "Classifying metrics for assessing object-oriented software maintainability: A family of metrics' catalogs," *Journal of Systems and Software*, Vol. 103, 2015, pp. 85–101.
[18] I. Crnkovic, M. Larsson, and O. Preiss, "Concerning predictability in dependable component-based systems: Classification of quality attributes," 2005.
[19] S.S. Thapar, P. Singh, and S. Rani, "Challenges to development of standard software quality model," *International Journal of Computer Applications*, Vol. 49, No. 10, 2012.
[20] C. Izurieta, D. Reimanis, E. O'Donoghue, K. Liyanage, A.R.M. Muneza et al., "A generalized approach to the operationalization of Software Quality Models," *PeerJ Computer Science*, Vol. 10, 2024, p. e2357.
[21] I. Crnkovic, S. Sentilles, A. Vulgarakis, and M.R. Chaudron, "A classification framework for software component models," *IEEE Transactions on Software Engineering*, Vol. 37, No. 5, 2011, pp. 593–615.
[22] M. Glinz, "On non-functional requirements," in *15th IEEE International Requirements Engineering Conference*, 2007, pp. 21–26.
[23] F. Deissenboeck, E. Juergens, K. Lochmann, and S. Wagner, "Software quality models: Purposes, usage scenarios and requirements," in *2009 ICSE Workshop on Software Quality*. IEEE, 2009, pp. 9–14.
[24] N. Yılmaz and A.K. Tarhan, "Meta-models for software quality and its evaluation: A systematic literature review," in *International Workshop on Software Measurement and the 15th International Conference on Software Process and Product Measurement, Mexico*, 2020.
[25] M. Broy, F. Deissenboeck, and M. Pizka, "Demystifying maintainability," in *Proceedings of the 2006 International Workshop on Software Quality*. ACM, 2006, pp. 21–26.
[26] B. Kitchenham, "What's up with software metrics?–a preliminary mapping study," *Journal of Systems and Software*, Vol. 83, No. 1, 2010, pp. 37–51.
[27] F. Zhang, A. Mockus, Y. Zou, F. Khomh, and A.E. Hassan, "How does context affect the distribution of software maintainability metrics?" in *2013 IEEE International Conference on Software Maintenance*. IEEE, 2013, pp. 350–359.
[28] B. Kitchenham and S. Pfleeger, "Software quality: The elusive target," *IEEE Software*, Vol. 12, No. 9, 1996.
[29] J. Eckhardt, A. Vogelsang, and D.M. Fernández, "Are "non-functional" requirements really non-functional? an investigation of non-functional requirements in practice," in *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*. IEEE, 2016, pp. 832–842.
[30] N.D. Anh, D.S. Cruzes, R. Conradi, M. Höst, X. Franch et al., "Collaborative resolution of requirements mismatches when adopting open source components," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 2012, pp. 77–93.

[31] J. Saraiva, S. Soares, and F. Castor, "Towards a catalog of object-oriented software maintainability metrics," in *2013 4th International Workshop on Emerging Trends in Software Metrics (WETSoM)*. IEEE, 2013, pp. 84–87.
[32] C. Gencel, R. Heldal, and K. Lind, "On the relationship between different size measures in the software life cycle," in *2009 16th Asia-Pacific Software Engineering Conference*. IEEE, 2009, pp. 19–26.
[33] C. Wohlin, E. Papatheocharous, J. Carlson, K. Petersen, E. Alégroth et al., "Towards evidence-based decision-making for identification and usage of assets in composite software: A research roadmap," *Journal of Software: Evolution and Process*, Vol. 33, No. 6, 2021, p. e2345.
[34] B. Henderson-Sellers, "Bridging metamodels and ontologies in software engineering," *Journal of Systems and Software*, Vol. 84, No. 2, 2011, pp. 301–313.
[35] G. Stănescu and S.V. Oprea, "Recent trends and insights in semantic web and ontology-driven knowledge representation across disciplines using topic modeling," *Electronics*, Vol. 14, No. 7, 2025, p. 1313.
[36] C. Antoniou and N. Bassiliades, "A tool for requirements engineering using ontologies and boilerplates," *Automated Software Engineering*, Vol. 31, No. 1, 2024, p. 5.
[37] N. Guarino, *Formal ontology in information systems: Proceedings of the first international conference (FOIS'98), June 6-8, Trento, Italy*, Vol. 46. IOS press, 1998.
[38] D. Fensel, *Ontologies: A silver bullet for knowledge management and electronic-commerce*. Berlin: Spring-Verlag, 2004.
[39] G. Van Heijst, A.T. Schreiber, and B.J. Wielinga, "Using explicit ontologies in KBS development," *International Journal of Human-Computer Studies*, Vol. 46, No. 2-3, 1997, pp. 183–292.
[40] G. Guizzardi, M. Lopes, F. Baião, and R. Falbo, "On the importance of truly ontological distinctions for ontology representation languages: An industrial case study in the domain of oil and gas," in *Enterprise, Business-Process and Information Systems Modeling*. Springer, 2009, pp. 224–236.
[41] G. Guizzardi, F. Baião, M. Lopes, and R. Falbo, "The role of foundational ontologies for domain ontology engineering: An industrial case study in the domain of oil and gas exploration and production," *International Journal of Information System Modeling and Design (IJISMD)*, Vol. 1, No. 2, 2010, pp. 1–22.
[42] G. Guizzardi, "On ontology, ontologies, conceptualizations, modeling languages, and (meta) models," *Frontiers in Artificial Intelligence and Applications*, Vol. 155, 2007, p. 18.
[43] G. Guizzardi, *Ontological foundations for structural conceptual models*, Ph.D. dissertation, University of Twente, 2005. [Online]. https://research.utwente.nl/en/publications/ontological-foundations-for-structural-conceptual-models/
[44] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, and L. Schneider, "Sweetening ontologies with dolce," in *International Conference on Knowledge Engineering and Knowledge Management*. Springer, 2002, pp. 166–181.
[45] M.F. Bertoa, A. Vallecillo, and F. García, "An ontology for software measurement," in *Ontologies for software engineering and software technology*. Springer, 2006, pp. 175–196.
[46] J.R. Hilera and L. Fernández-Sanz, "Developing domain-ontologies to improve sofware engineering knowledge," in *International Conference on Software Engineering Advances (ICSEA)*. IEEE, 2010, pp. 380–383.
[47] M.J. Blas, S. Gonnet, and H. Leone, "An ontology to document a quality scheme specification of a software product," *Expert Systems*, Vol. 34, No. 5, 2017, p. e12213.
[48] N. Anquetil, K.M. de Oliveira, and M.G. Dias, "Software maintenance ontology," in *Ontologies for Software Engineering and Software Technology*. Springer, 2006, pp. 153–173.
[49] O. Lassila and D. McGuinness, "The role of frame-based representation on the semantic web," *Linköping Electronic Articles in Computer and Information Science*, Vol. 6, No. 5, 2001, p. 2001.
[50] N.F. Noy, D.L. McGuinness et al., "Ontology development 101: A guide to creating your first ontology," 2001.
[51] W.A. Conklin, "Software assurance: The need for definitions," in *International Conference on System Sciences (HICSS)*. IEEE, 2011, pp. 1–7.

[52] S. Sentilles, E. Papatheocharous, and F. Ciccozzi, "What do we know about software security evaluation? A preliminary study," in *6th International Workshop on Quantitative Approaches to Software Quality, APSEC*, 2018.
[53] S. Henninger and P. Ashokkumar, "An ontology-based metamodel for software patterns," University of Nebraska – Lincoln, Tech. Rep. TR-UNL-CSE-2006-00005, 2006.
[54] F. Béhé, S. Galland, N. Gaud, C. Nicolle, and A. Koukam, "An ontology-based metamodel for multiagent-based simulations," *Simulation Modelling Practice and Theory*, Vol. 40, 2014, pp. 64–85.
[55] S. Sentilles, E. Papatheocharous, F. Ciccozzi, and K. Petersen, "Property models for the automotive domain," Mälardalen University, Tech. Rep., 2016. [Online]. http://www.es.mdh.se/publications/4295-
[56] J. Brank, M. Grobelnik, and D. Mladenić, "A survey of ontology evaluation techniques," in *Proceedings of the Conference on Data Mining and Data Warehouses*, 2005, pp. 166–170.
[57] A. Burton-Jones, V.C. Storey, V. Sugumaran, and P. Ahluwalia, "A semiotic metrics suite for assessing the quality of ontologies," *Data and Knowledge Engineering*, Vol. 55, No. 1, 2005, pp. 84–102.
[58] T. Gorschek, P. Garre, S. Larsson, and C. Wohlin, "A model for technology transfer in practice," *IEEE Software*, Vol. 23, No. 6, 2006, pp. 88–95. [Online]. https://doi.org/10.1109/MS.2006.147
[59] S. Sentilles, E. Papatheocharous, and F. Ciccozzi, "What do we know about software security evaluation? a preliminary study," in *6th International Workshop on Quantitative Approaches to Software Quality*, 2018. [Online]. http://www.es.mdh.se/publications/5277-
[60] J. Maxwell, "Understanding and validity in qualitative research," *Harvard Educational Review*, Vol. 62, No. 3, 1992, pp. 279–301.
[61] K. Petersen and C. Gencel, "Worldviews, research methods, and their relationship to validity in empirical software engineering research," in *Joint Conference of the 23rd International Workshop on Software Measurement and the Eighth International Conference on Software Process and Product Measurement (IWSM-MENSURA)*. IEEE, 2013, pp. 81–89.
[62] J. Axelsson, "Evolutionary architecting of embedded automotive product lines: An industrial case study," in *European Conference on Software Architecture*, 2009, pp. 101–110.
[63] H. Gustavsson and U. Eklund, "Architecting automotive product lines: Industrial practice," in *Software Product Lines: Going Beyond*. Springer, 2010, pp. 92–105.
[64] R.A. McGee, U. Eklund, and M. Lundin, "Stakeholder identification and quality attribute prioritization for a global Vehicle Control System," in *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*. ACM, 2010, pp. 43–48.
[65] U. Eklund and J. Bosch, "Architecture for embedded open software ecosystems," *Journal of Systems and Software*, Vol. 92, 2014, pp. 128–142.
[66] M. Broy, "Challenges in automotive software engineering," in *Proceedings of the 28th International Conference on Software Engineering*, ICSE '06. New York, NY, USA: ACM, 2006, pp. 33–42.
[67] A. Pretschner, M. Broy, I.H. Kruger, and T. Stauner, "Software engineering for automotive systems: A roadmap," in *Future of Software Engineering*, FOSE '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 55–71.
[68] R. Etemaadi, K. Lind, R. Heldal, and M.R. Chaudron, "Quality-driven optimization of system architecture: Industrial case study on an automotive sub-system," *Journal of Systems and Software*, Vol. 86, No. 10, 2013, pp. 2559–2573.
[69] C. Ebert and C. Jones, "Embedded software: Facts, figures, and future," *Computer*, Vol. 42, No. 4, 2009, pp. 42–52.
[70] B.W. Boehm, *Software Engineering Economics*, 1st ed. Prentice Hall PTR, 1981.
[71] J. Li, G. Ruhe, A. Al-Emran, and M.M. Richter, "A flexible method for software effort estimation by analogy," *Empirical Software Engineering*, Vol. 12, No. 1, 2007, pp. 65–106.
[72] M. Jørgensen, "A review of studies on expert estimation of software development effort," *Journal of Systems and Software*, Vol. 70, No. 1–2, 2004, pp. 37–60.

[73] S.L. Pfleeger, F. Wu, and R. Lewis, "Software cost estimation and sizing methods: issues, and guidelines, Volume 269, Rand Comporation," http://www.rand.org/content/dam/rand/pubs/monographs/2005/RAND/MG269.pdf, 2005.
[74] V. Chapurlat, B. Kamsu-Foguem, and F. Prunet, "Enterprise model verification and validation: An approach." *Annual Reviews in Control*, Vol. 27, No. 2, 2003, pp. 185–197.
[75] R.S. Pressman, *A manager's guide to software engineering*. McGraw-Hill, Inc., 1993.
[76] R. Broek, J. Gorman, O. Haugen, G. Melby, B. Moller-Pedersen et al., "Quality by construction exemplified by TIMe-the integrated methodology," *Telektronikk*, Vol. 95, No. 1, 1999, pp. 73–82.
[77] R. Morgan, *Component library retrieval using property models*, Ph.D. dissertation, Durham University, 1991.
[78] M. Belguidoum and F. Dagnat, "Dependability in software component deployment," in *Dependability of Computer Systems*, 2007, pp. 223–230.
[79] A. Billig, S. Busse, A. Leicher, and J.G. Süß, "Platform independent model transformation based on triple," in *Middleware 2004*. Springer, 2004, pp. 493–511.
[80] V.A. Paun, *Precise and Adaptable Worst-Case Execution Time Estimation in Hard Real-Time Systems*, Ph.D. dissertation, Ecole Doctorale Polytechnique, 2014.
[81] I. Wenzel, R. Kirner, B. Rieder, and P. Puschner, "Measurement-based worst-case execution time analysis," in *Third IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (SEUS'05)*, 2005, pp. 7–10.
[82] A. Ermedahl, *A modular tool architecture for worst-case execution time analysis*, Ph.D. dissertation, Acta Universitatis Upsaliensis, 2003.
[83] S. Sentilles, F. Ciccozzi, and E. Papatheocharous, "PROMOpedia: A web-content management-based encyclopedia of software property models," in *40th International Conference on Software Engineering – Demonstrations track, ICSE*, 2018. [Online]. http://www.es.mdh.se/publications/5086-

Authors and affiliations

Efi Papatheocharous
e-mail: efi.papatheocharous@ri.se
ORCID: https://orcid.org/0000-0002-5157-8131
Digital Systems, RISE Research Institutes of Sweden, Sweden

Séverine Sentilles
e-mail: severine.sentilles@mdu.se
ORCID: https://orcid.org/00000-0003-0165-3743
School of Innovation, Design and Engineering, Mälardalen University, Sweden

Kai Petersen
e-mail: kai.petersen@hs-flensburg.de
ORCID: https://orcid.org/0000-0002-1532-8223
University of Applied Sciences Flensburg, Germany

Federico Ciccozzi
e-mail: federico.ciccozzi@mdu.se
ORCID: https://orcid.org/0000-0002-0401-1036
School of Innovation, Design and Engineering, Mälardalen University, Sweden

e-Informatica Software Engineering Journal
Volume 20, Issue 1, pages: 1–18
DOI: 10.37190/e-Inf260102
BibTeX

A Defect Classification Framework for AI-Based Software Systems (AIODC)

Mohammed Alannsary*
*Corresponding author: alannsary@hotmail.com

Article info

Dataset link: https://doi.org/10.5281/zenodo.17843077

Keywords:
software testing
analysis and verification
software quality
AI and knowledge based software engineering

Submitted: 29 Aug. 2025
Revised: 10 Dec. 2025
Accepted: 10 Dec. 2025
Available online: 12 Dec 2025

Abstract

Context: Artificial Intelligence (AI) is increasingly integrated into critical domains, making defect analysis essential to ensure system quality and reliability. Current defect classification frameworks do not adequately address the unique properties of AI systems.
Objective: This paper proposes AIODC, a defect classification framework inspired by the Orthogonal Defect Classification (ODC) that incorporates AI-specific characteristics.
Method: The framework extends ODC by introducing three new attributes – Data, Learning, and Thinking – and adds a "Catastrophic" severity level to account for risks associated with AI. Additionally, it modifies impact mapping utilizing AI/AIP quality models. The methodology was validated through a case study that examined 42 actual Keras defects.
Results: This study demonstrated the feasibility of modifying ODC for AI systems to classify its defects. The case study indicated that defects occurring during the Learning phase are the most prevalent and were significantly linked to high severity, whereas defects in the Thinking phase primarily impact trustworthiness and accuracy.
Conclusions: The results affirm the practicality and significance of AIODC in identifying high-risk defect categories, thus facilitating more focused and effective quality assurance strategies in AI-driven software systems.

1. Introduction

The domain of software quality constitutes a specialized branch within the field of software engineering that rigorously investigates, evaluates, quantifies, and enhances the quality attributes associated with a software product [1–3]. In general terms, a software product is regarded as possessing a high level of quality if it is devoid of, or exhibits a minimal number of, issues that originate from the software itself and result in limited adverse effects on its users.

Defect analysis is the branch of software engineering that analyzes and resolves discovered defects while developing the software or using it. When it comes to software quality improvement, defect analysis is known to be a major contributor. Several defect analysis techniques and methods are available such as defect classification, where found defects are

© 2026 The Authors. Published by Wrocław University of Science and Technology Publishing House.
This is an open access article under the CC BY-SA 4.0 license international.

analyzed and grouped based on their characteristics. One of the main and well-known defect classification methods is the Orthogonal Defect Classification (ODC) approach [1, 4–6].

Implementing defect analysis has been successful in several types of software, such as web applications, Software as a Service (SaaS), and traditional software, yet defect analysis for Artificial Intelligence (AI) systems needs to be addressed due to the additional and different characteristics that AI systems have over traditional systems, web applications, and SaaS.

AI plays a transformative role in improving several fields of life, such as medicine, engineering, economy, social etc. The main advantage of AI is that it helps and aids in data analysis, prediction, and decision making. AI solutions are built mainly using software, therefore it is crucial to evaluate its quality to assure its correctness. One could argue that software written to produce AI does not differ from traditional software, web applications, or SaaS. However, each of the aforementioned software types – including AI systems – has its own attributes and characteristics, and needs to be catered to accordingly [7–10].

The quality of the AI system encompasses both the quality of AI as a universal yet specific entity and the quality of AI Platforms (AIP) which are the software and hardware platforms that facilitate its implementation [11]. Defining attributes and characteristics of an AI system may be troublesome, it can learn, adapt, and make decisions based on complex unstructured data and provides solutions, which fundamentally emulates human cognitive capabilities [12], while other systems are based on static rules and predetermined algorithms, works with structured data, and follows established processes. Therefore, there is a need to customize defect analysis techniques to cater for AI systems.

Despite numerous studies utilizing ODC in traditional, web, and cloud-based software, there remains a lack of a well-defined defect classification framework that addresses the unique challenges of AI systems. Current methodologies do not adequately account for the dynamic features of AI – such as data dependency, continuous learning, and autonomous decision-making – which alter the nature and propagation of defects. This research fills that gap by introducing AIODC, an adaptation of ODC that incorporates AI-specific attributes, a domain-sensitive severity scale, and quality impact mapping. The objective of the study is to assess the applicability of AIODC and to uncover defect patterns that are unique to AI-driven systems through an empirical case study.

The remainder of this work is organized as follows: the next Section presents background and related work. Section 3 discusses ODC, and how it was modified and/or adapted to different types of software. Section 4 describes the proposed framework. In Section 5 a case study is provided to show the applicability of the framework. Section 7 contains the conclusion and prospective.

2. Background and related work

Ensuring the quality of AI requires recognizing its unique attributes, which differ from those of conventional software, web applications, and SaaS. AI typically integrates novel algorithms and functions, evolves through continuous implementation, and enables developers to use diverse datasets for system training [12].

Consequently, existing methodologies and frameworks for assessing the quality of conventional, web-based software, and SaaS require adaptation to account for AI's distinctive characteristics. Reliability is particularly critical when AI decisions may affect human lives [7].

AI is now a crucial component of the software development life cycle. Its integration marks a significant shift in how software is designed, built, tested, maintained, and deployed.

The growing use of AI across diverse sectors is driving a transition away from conventional development frameworks. Developers can now leverage AI to refine processes, reduce errors, and build more sophisticated systems using machine-learning algorithms, natural-language processing, and data analytics. This transition underscores the necessity for organizations to embrace and incorporate AI into various stages of the software development life-cycle (SDLC) [13].

Defect analysis for AI software remains in its early stages, even though AI has already been applied to classify and detect defects in domains such as product inspection, image processing, and sound analysis. Moreover, several studies have explored using AI as a tool within software engineering itself–for example, in code generation, software testing, defect detection, and defect classification [14–23]. However, there has not been a defect analysis approach developed for AI systems specifically, nor was one introduced in the literature.

Vinayagasundaram and Srivatsa [24] proposed metrics for evaluating the quality of AI software and noted that its components are reusable. Their architecture comprised four layers–task specification, problem solver, domain, and adapter. The main structure is segmented into sub components in a layered manner, allowing for the addition of more layers that can alter the system's behavior. Several metrics were used to measure components quality, such as: depth of inheritance, complexity level, and source code.

Tao et al. [25] highlighted that AI software's unique characteristics introduce new challenges for validating software and system quality. Their study discussed how these challenges affect the functional aspects of AI software. In addition, they articulated the comprehension of testing AI software in relation to new features and requirements. Moreover, they presented the current classifications of AI software testing and examined different testing strategies. Furthermore, they provided an illustration of test quality evaluation and criteria analysis. Finally, a practical examination of quality validation for an image recognition system was executed through a metamorphic testing technique.

Chillarege et al. [4] proposed Orthogonal Defect Classification (ODC) – a methodology that provides in-process feedback to developers by deriving defect signatures from the development process. Generally, ODC serves as a technique for categorizing defects to identify which phase of the software development life-cycle requires focus. Defects are categorized based on several dimensions: phase of discovery, defect type, severity, source, trigger, and impact.

Ma and Tian [5] developed a method for classifying and analyzing web errors through integrating ODC into the web environment. Attributes pertinent to the web context were selected from web access logs to serve as defect types within ODC. This approach facilitated the identification of issues and offered valuable feedback to the web applications development team.

Alannsary and Tian [6] proposed a framework for defect analysis tailored for a cloud-based SaaS. The framework was influenced by the original ODC model and acknowledges the distinctive features of SaaS. It was structured into three phases: data source analysis, classification, and results analysis. The entries from the web server log file are examined to pinpoint errors based on the values of the "Protocol status" field, the detected defects are then classified according to six defect attributes specifically designed for the cloud context. Successful implementation of one-way and two-way analyses has yielded improved insights into the identified defects and their solutions. A case study was provided to validate the effectiveness of the proposed framework.

Kharchenko et al. [11] aimed at establishing and showcasing the implementation of quality models for AI, AIP, and AIS, based on the definition and organization of character-

istics. The principles underlying the development of AI quality models and their sequence are thoroughly justified. Approaches for articulating definitions of AIS characteristics, as well as methods for depicting dependencies and hierarchies of characteristics, were discussed. In addition, they proposed definitions and harmonization strategies for the hierarchical relationships among 46 characteristics of AI and AIP. Moreover, the quality models pertaining to AI, AIP, and AIS are detailed, with a focus on the most critical characteristics. Finally, two examples of AIS quality models were elaborated upon. It is worth mentioning that they pointed out that experts have the liberty of including or eliminating characteristics based on the context of the AI system.

Morovati et al. [26] examined the reproducibility and verifiability of bugs found in Machine Learning (ML) systems, which is part of AI systems, highlighting the key factors associated with each aspect. Subsequently, they delve into the difficulties of creating a benchmark for bugs in ML software systems and introduce a bug dataset to be used as a benchmark called "defect4ML". The dataset meets all the criteria of a standard benchmark, including relevance, reproducibility, fairness, verifiability, and usability. The dataset contained 100 bugs reported by ML developers on platforms such as GitHub and Stack Overflow, utilizing two of the most widely used ML frameworks: TensorFlow and Keras. Cross-platform defect reporting (for instance, GitHub compared to Stack Overflow) poses a risk of duplication, since developers frequently repost issues [27]. The process of duplication continues to be difficult in the absence of issue-ID cross-referencing.

While earlier research has focused on predictive defect detection–such as deep-learning or CodeBERT-based models that anticipate defect-prone components–these approaches differ fundamentally from the proposed AIODC framework. Unlike these approaches, AIODC does not seek to forecast future defects; rather, it classifies defects that have already been identified to ascertain their origin, severity, and impact on quality. This differentiation establishes AIODC as a supplementary tool that improves diagnostic comprehension rather than prediction, effectively connecting abstract AI quality models with practical defect analysis processes.

3. Orthogonal defect classification (ODC)

Since its first debut, ODC played a noticeable role in defect analysis for traditional software. Several researchers referenced the concept, and some adapted it to different fields. With the introduction of the Web and cloud computing, Software has changed in the way it is delivered and used. Therefore, This section describes how the concept was modified and/or adapted to other types of Software.

3.1. Original ODC

In traditional software and systems, Chillarege et al. [4] specified a cause effect relationship in ODC. There were two cause attributes: defect type and defect trigger, and several effect attributes such as severity, reliability growth, and impact areas. Table 1 shows the cause and effect relationship of the original ODC.

In addition, in the defect type attribute, defects were classified and associated with a phase in the development processes as shown below.

Function associated with the Design phase.

Interface associated with the Low Level Design phase.

Table 1. Original ODC cause-effect relationships [4], adapted as AIODC's foundation

CAUSE	⟺	EFFECT
Defect Type (Development Process)		Severity
		Impact areas (CUPRIMD)
		Reliability Growth
Defect Trigger (Verification Process)		Defect Density
		Rework on Fixes
		Etc.

Checking associated with the Low Level Design or Code phase.
Assignment associated with the Code phase.
Timing/Serialization associated with the Low Level Design phase.
Build/Package/Merge associated with the Library Tools phase.
Documentation associated with the Publications phase.
Algorithm associated with theLow Level Design phase.

Moreover, defects were classified in regards to the impact areas based on IBM's CUPRIMD [28]. Table 2 shows the attributes and metric areas associated with it.

Table 2. Traditional impact areas (IBM CUPRIMD) [4] – replaced by AI-specific characteristics in AIODC

Attribute	Metric Areas
Capability	Functionality delivered versus requirements Volume of function to deliver
Usability	Ease of learning important tasks Ease of completing a task Intuitiveness
Performance	Transaction throughput Response time to enquiry Size of machine needed to run the product
Reliability	Mean time between failures Number of defects Severity of defects Severity/impact of failures
Installibility	Ease of making product available for use Time to complete installation Skill level needed by installer
Maintainability	Ease of problem diagnosis Ease of fixing problem correctly
Documentation	Ease of understanding Ease of finding relevant information Completeness of information

3.2. Web ODC

Ma and Tian [5] adapted the ODC concept to the web environment through identifying web error attributes. The Web ODC attributes are listed below.

1. Response code

2. File type
3. Referrer type
4. Time period

3.3. Cloud ODC

Due to the special characteristics of SaaS, such as multi-tenancy and isolation, Alannsary and Tian [6] adapted the ODC concept to a SaaS running in the Cloud. In the cloud ODC framework one attribute (Layer affected) was added to the Original ODC attributes as depicted in Figure 1 including the newly added one which is **bold**.

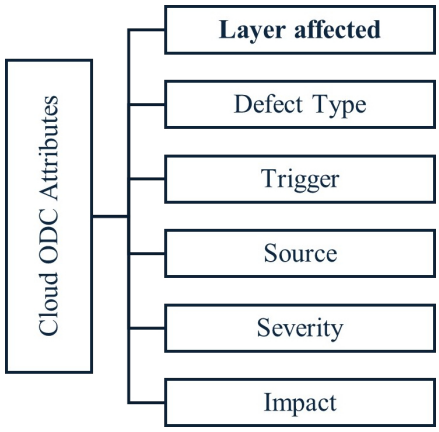


Figure 1. Cloud ODC attribute schema [6] – baseline for AIODC's extended structure

In addition, two new defect types (Isolation, IaaS/Paas) were introduced to the original ODC defect types as shown in Table 3, the newly added defect types are *italic*.

Table 3. Cloud ODC defect types [6] – extended in AIODC with data/learning/thinking dimensions

ODC Version	Defect type
Original ODC	Function. Interface Checking Assignment Timing/serialization Build/package/merge Documentation Algorithm
Cloud ODC	*Isolation* *IaaS/PaaS*

4. Proposed framework

Defining attributes and characteristics of an AI systems is essential to allow classification of its defects. However, known software/system characteristics are not sufficient due to the

special nature of AI systems. Several published literature discussed AI quality and AIP quality models such as [9, 10].

An AI system is different from other systems in the sense that it emulates human cognitive capabilities, it can learn, adapt, make decisions, and provide solutions based on complex unstructured data. To utilize the classification paradigm of the ODC model in AI system development projects, it is essential to undertake particular actions that cater to the distinctive features of AI. Therefore, the original ODC needs to be modified to be adapted to the defect analysis process of AI systems.

The AIODC classification process employs a systematic and repeatable methodology to guarantee consistency. Initially, each reported defect is scrutinized to ascertain whether it relates to AI-specific behavior by categorizing it into one of four AI attribute categories: Data (issues within training/testing datasets), Learning (errors during model training), Thinking (mistakes in inference or decision-making), or Not Related (general software defects). Each category is accompanied by explicit examples, such as mislabeled training data for Data defects, unstable convergence in Learning defects, or incorrect decision thresholds in Thinking defects. Severity ratings are allocated based on established decision rules that take into account the application domain, potential downstream effects, and the reversibility of outcomes, ensuring that "Catastrophic" severity is designated for defects with a significant risk of irreversible damage in critical domains. Ultimately, the impact mapping process utilizes Kharchenko's AI/AIP quality model, with annotators directed by a rubric that connects defect types to quality characteristics grounded in functional dependencies and observed system behavior.

First, the list of attributes is modified by introducing a new attribute called AI. The new attribute allows classifying defects based on Data, Learning, Thinking, and Not Related as explained in Table 4. The attribute allows classifying defects based on the Data used to train the AI, the AI's learning process, and the AI's training process, or not related to AI respectively. Figure 2 depicts the suggested AIODC attributes including the newly added one which is **bold**.

Table 4. AIODC's novel AI attribute
– classifying defects by data, learning, thinking, or not related phases

Classification	Description
Data	Issues with training / testing data.
Learning	Faults in the AI model training process.
Thinking	Faults in inference, logic, or decision making.
Not Related	Defects unrelated to AI logic or behavior.

Second, the severity attribute is changed from a four scale measure to a five scale measure by adding a new severity level named "Catastrophic." The severity attribute in the original ODC is measured on a scale from one to four (Critical, High, Medium, and Low), since defects in an AI systems may be catastrophic due to the fact that they may be used in numerous significant applications including ones related to human health monitoring, and illness diagnosing [29]. Therefore, the severity attribute is modified to be measured on a one to five scale (Catastrophic, Critical, High, Medium, and Low). Figure 3 depicts the suggested Severity attribute scale including the newly added level which is **bold**. To address context-sensitive risk profiles in AI systems, severity classifications encompass three key factors:

1. Application Criticality:

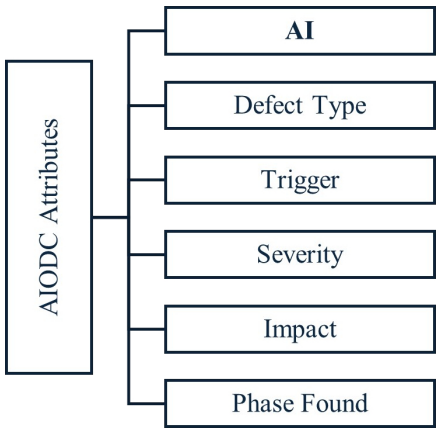


Figure 2. AIODC's attribute framework
– integrating novel AI dimension with ODC core

- Catastrophic/Critical: Systems that affect human safety (healthcare, autonomous vehicles).
- High/Medium: Enterprise systems (finance, security)
- Low: Non-critical applications (entertainment, productivity)

2. Harm Reversibility:
   - Irreversible harm, such as: fatal misdiagnosis (Catastrophic)
   - Reversible errors, such as:transient data corruption (High/Medium)
3. Failure Scope:
   - Systemic failures, such as: poisoned training data (increase severity)
   - Localized errors, such as: single inference fault (allow for lower severity)

This methodology is consistent with recognized risk management standards, including ISO 14971:2019 for medical devices [30], which highlights severity levels that depend on context. Similarly, harm taxonomies specific to AI (for instance, irreversible systemic failures) are informed by fundamental safety literature [31].

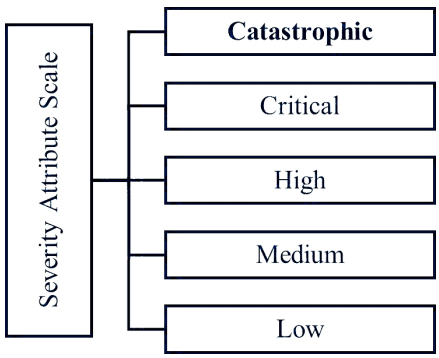


Figure 3. AIODC's 5-tier severity scale
– with domain-adaptive "catastrophic" level for AI risks

Finally, the impact areas attribute is modified to accommodate the new characteristics of AI system quality. As explained previously, Kharchenko et al. [11] developed a quality model and suggested a list of characteristics that are appropriate for AI systems, where the quality of an AI system is composed of the quality of AI and the quality of AIP. The

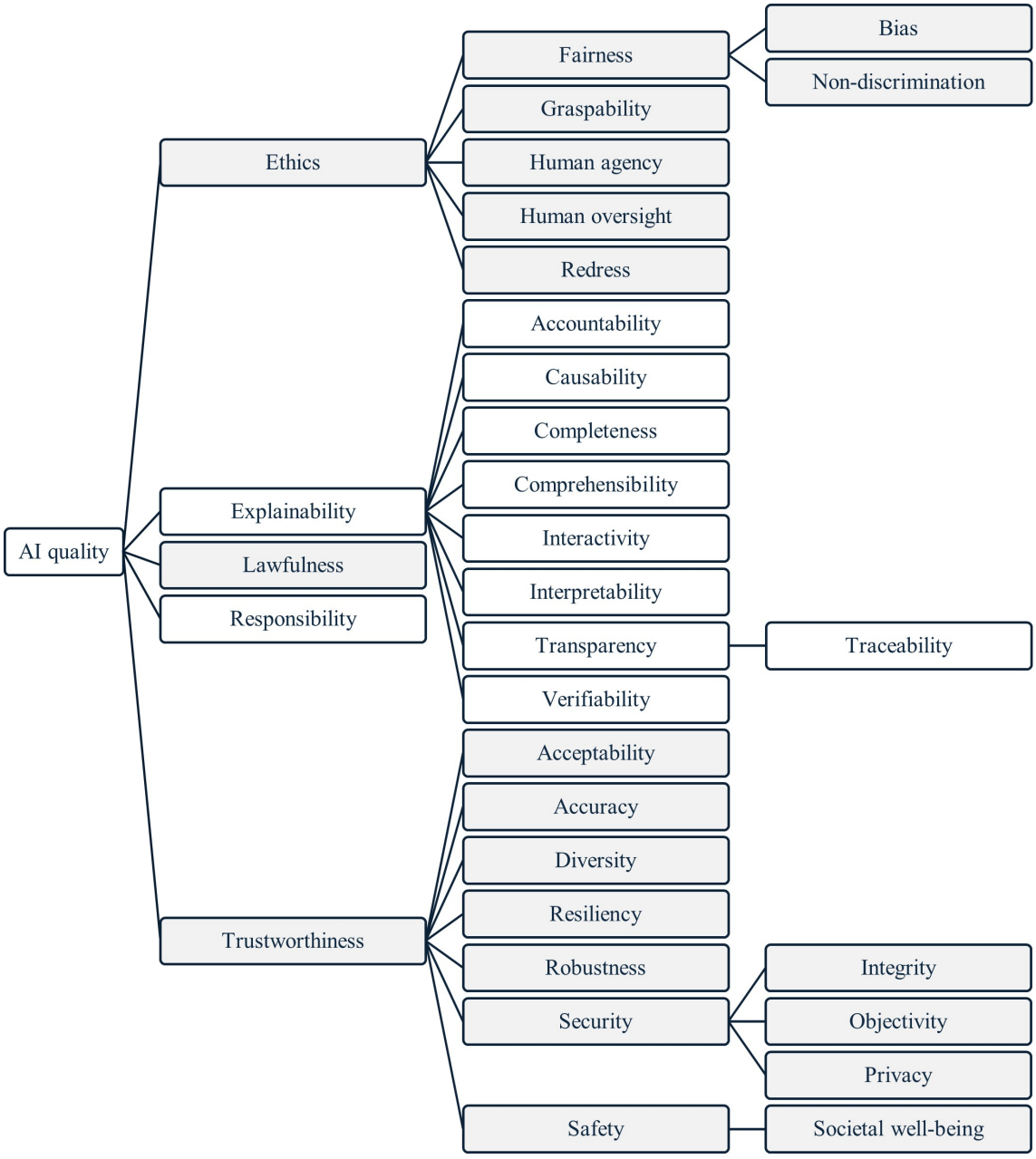


Figure 4. AI quality characteristics (Kharchenko et al. [11]) – foundation for AIODC impact mapping

characteristics were distributed among three layers, so if characteristic C1 depends on characteristic C2, then C2 will be in the next lower layer that C1 is in. In addition, some characteristics were assigned to AI quality, some assigned to AIP quality, and some were shared between both. Figure 4 depicts AI quality characteristic, while Figure 5 depicts AIP quality characteristic.

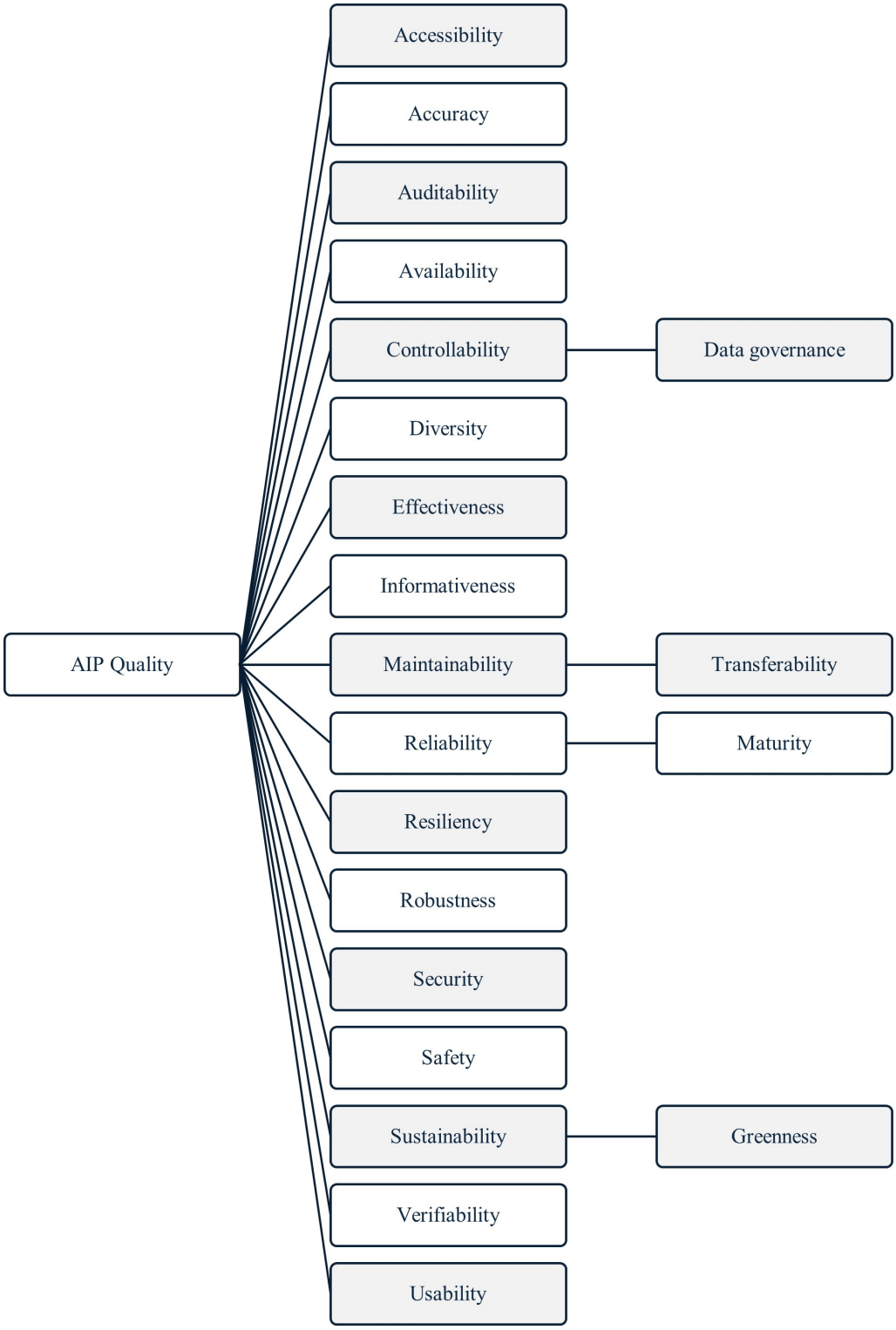


Figure 5. AIP quality characteristics (Kharchenko et al. [11]) – foundation for AIODC impact mapping

5. Case study

To test the applicability and suitability of the AIODC framework for AI systems, the benchmark dataset created by Morovati et al. [26] is utilized. The dataset comprises 100 ML defects sourced from TensorFlow/Keras on GitHub and Stack Overflow. For this study, 42 unique Keras defects from GitHub were selected for detailed analysis. This subset provided the highest count of non-duplicate, well-documented cases. The selection of Keras defects from GitHub was based on three primary criteria:

1. Highest defect count: 42 compared to TensorFlow's 20 on GitHub;
2. Uniformity of platform: to prevent duplication in cross-platform reporting;
3. Consistency within the framework: to minimize confounding variables arising from differences between frameworks.

Therefore, defects reported for TensorFlow or on Stack Overflow were excluded to prevent duplication bais.

As mentioned above, the dataset contains 100 bugs reported on two platforms: GitHub and Stack Overflow, for two widely used ML frameworks: TensorFlow and Keras. In this study, defects reported on GitHub for the Keras framework will classified based on the proposed AIODC framework. Such decision was made based on the number of defects that each framework has on each platform. Table 5 shows the number of defects for each framework on both platforms.

Table 5. Defect distribution in benchmark dataset [26]
– Keras/GitHub selected ($n = 42$) for case study due to maximal unique defects

Platform	TensorFlow	Keras
GitHub	20	42
Stack Overflow	3	35

5.1. Classification methodology

To classify defects to implement AIODC, Two separate annotators examined defects according to predefined guidelines:

1. Training: A two-hour session featuring sample defect mappings.
2. Procedure:
   – Annotators evaluated descriptions of GitHub issues;
   – Designated AI attributes (Data/Learning/Thinking/Not Applicable);
   – Assigned severity ratings (Catastrophic-Low).
3. Dispute Resolution: A third expert evaluated the conflicting labels.
4. Reliability Assessment: Cohen's Kappa = 0.82 (indicating substantial agreement).

5.2. Bug classification based on the AI attribute

The number of defects reported for the Keras framework on GitHub is 42 defects. Classifying defects based on the AI attribute resulted in having 2 defects related to Data, 18 related to the Learning process of the AI framework, 14 related to the Thinking process of the AI framework, and 8 defects not related to AI. Classification results are shown in Table 6.

Table 6. AIODC classification of unique Keras defects (GitHub subset, $n = 42$)

Defect Description	Count	AI Attribute	Total	%
Deprecated API	3	Not Related	8	19.05%
Missing API call	2	Not Related		
Missing argument scoping	1	Not Related		
Wrong API usage	2	Not Related		
Missing dense layer	1	Think	14	33.33%
Suboptimal network structure	4	Think		
Wrong size for convolutional layer	1	Think		
Wrong layer type	2	Think		
Wrong network architecture	3	Think		
Wrong type of activation function	3	Think		
Wrong tensor shape	1	Data	2	4.76%
Missing pre processing step	1	Data		
Suboptimal batch size	4	Learn	18	42.86%
Suboptimal number of epochs	4	Learn		
Wrong loss function calculation	1	Learn		
Wrong optimization function	4	Learn		
Wrong selection of loss function	5	Learn		

5.3. Bug classification based on the severity attribute

The severity of defects was categorized based on domain-adaptive thresholds outlined in Section 4, which evaluate:

1. Application Criticality: Keras (general ML development) was classified as medium-criticality (non-safety-critical).
2. Harm Reversibility: Irreversible errors (such as model collapse) heightened the severity level.
3. Failure Scope: Systemic defects (for instance, hyperparameter corruption) were given precedence over localized problems.

Therefore, the revised Severity Assignment Logic is:

1. Catastrophic (5): Defects that result in irreversible systemic damage (e.g., "incorrect loss function" causes permanent divergence of the model).
2. Critical (4): Defects that lead to recoverable yet significant outcomes (e.g., "incorrect tensor shape" causes failure in training; can be rectified through data correction).
3. High (3) to Low (1): Localized or transient defects (e.g., "suboptimal epochs" causes decrease in accuracy; can be resolved through retuning).

Severity classifications were aligned with the domain-adaptive standards outlined in Section 4, which encompass criticality levels inspired by ISO 14971 [30] and the AI failure modes detailed in [31]. In the case of Keras, categorized as medium-criticality, defects were classified as "Catastrophic" solely under the following conditions:

1. Irreversible systemic damage,
2. Equivalence of life/financial risk (according to [30] Annex C),
3. Uncontainable propagation of errors (as specified in [31] Section 3.1).

The application of these criteria to defects in Keras produced the classification shown in Table 7.

Table 7. Severity classification – validating domain-adaptive thresholds ($n = 42$)

Severity level	Defects ($n = 42$)	%
Catastrophic	9	21.43%
Critical	10	23.81%
High	12	28.57%
Medium	11	26.19%
Low	0	0.00%

5.4. Bug classification based on the impact attribute

Expanding upon the classification of defects, we subsequently examine the manner in which AI defects adversely affect the fundamental quality characteristics of AI systems and their associated platforms. Unlike traditional defect analysis, this approach isolates AI-specific failure vectors through quality-attribute-centric mapping. Although categorization serves to pinpoint the origins of failures (Data/Learning/Thinking/Not Related), effective remediation necessitates a comprehension of the specific quality characteristic that these defects undermine. This section correlates defects identified in the case study with empirically observed degradations in the quality characteristics of AI/AIP (following Kharchenko's quality model [11] for characteristic definitions), thereby uncovering deterministic patterns between defect types and impacted attributes. These consistent failure signatures facilitate accurate localization of quality risks – thereby affirming AIODC's core proposition that the sources of defects dictate impact profiles across functionally orthogonal quality dimensions. Table 8 shows the classification of found defects based on their impact on the Kharchenko AI/AIP Characteristics.

The classification of defects in relation to AI/AIP quality characteristics shown in Table 8 was performed through a two-phase expert evaluation procedure. Annotators employed a specified rubric that correlated defect types with Kharchenko's hierarchical quality characteristics, taking into account both the functional role of the defect's location (such as model architecture and training loop) and the type of failure (for instance, robustness degradation and accuracy loss). In instances of uncertainty, annotators referred to documentation and discussions from issue threads to gain a clearer understanding of the defect's context prior to finalizing the labels. This systematic mapping process minimizes subjectivity and improves reproducibility, thereby ensuring that subsequent researchers can utilize AIODC on other datasets with similar consistency.

Analysis indicates clear failure patterns across AI defect categories: Defects originating in the Learning phase predominantly degrade operational efficiency metrics (such as computational resource usage and training duration), while those emerging from the Thinking phase primarily compromise system integrity outcomes (including decision reliability and output consistency). This illustrates how defect genesis in AI systems dictates failure modes across functionally orthogonal impact domains.

This division illustrates the architectural distinction between the training and inference subsystems. Learning defects present themselves as resource-related issues, whereas Thinking defects arise as behavioral irregularities.

Table 8. Impact mapping of Keras defects to AI/AIP quality characteristics [11]

AI Defect	Quality Model	Impacted characteristic	
		Layer1	Layer2
Deprecated API	AIP	Maintainability	=
Missing API call	AIP	Reliability	=
Missing argument scoping	AI	Security	Integrity
Wrong API usage	AIP	Accuracy	=
Missing dense layer	AI	Trustworthiness	Accuracy
	AIP	Accuracy	=
Suboptimal network structure	AI	Effectiveness	=
Wrong size for convolutional layer	AI	Trustworthiness	Robustness
	AIP	Robustness	=
Wrong layer type	AI	Trustworthiness	Accuracy
	AIP	Accuracy	=
Wrong network architecture	AI	Trustworthiness	Accuracy
	AI	Explainability	Completeness
	AIP	Accuracy	=
Wrong type of activation function	AI	Trustworthiness	Accuracy
	AIP	Accuracy	=
Wrong tensor shape	AIP	Reliability	=
Missing pre processing step	AI	Trustworthiness	Robustness
	AIP	Robustness	=
Suboptimal batch size	AI	Effectiveness	=
Suboptimal number of epochs	AI	Trustworthiness	Accuracy
	AIP	Accuracy	=
	AIP	Effectiveness	=
Wrong loss function calculation	AI	Trustworthiness	Accuracy
	AIP	Accuracy	
Wrong optimization function	AI	Effectiveness	=
Wrong selection of loss function	AI	Trustworthiness	Accuracy
	AI	Trustworthiness	Robustness
	AIP	Accuracy	=
	AIP	Robustness	=

5.5. Two-way classification

While analyzing defects classified using one attribute (one-way classification) can be insightful. However, it may not adequately inform the development team about defects and the associated high-risk areas. Implementing two-way classification will enhance the understanding of the observed defects, which may lead to more informed decisions. Therefore, by utilizing two-way classification, it becomes feasible to concentrate on rectifying high-priority defects, ultimately improving the reliability and quality of the AI system. Two way classification was implemented on the AI and Severity attributes, results are depicted in Figure 6.

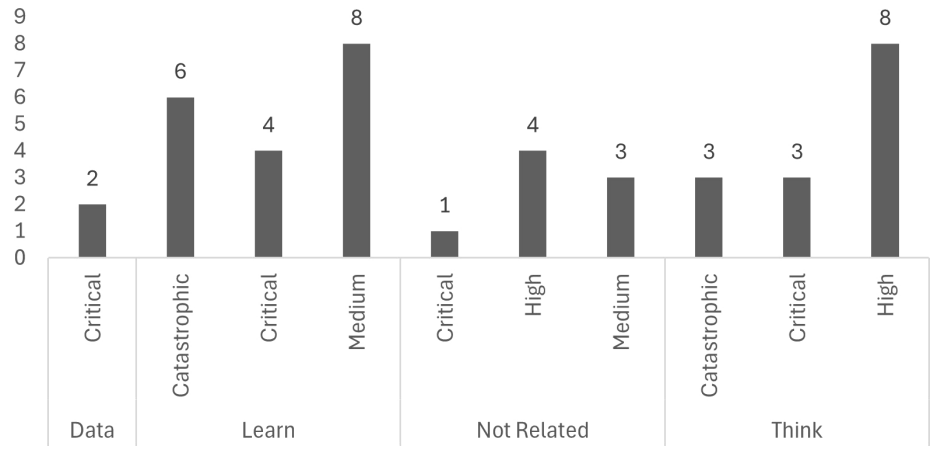


Figure 6. AI/severity correlation
– revealing high-risk defect clusters (e.g., learning + catastrophic)

6. Discussion

The findings of this research have significant implications for the practices of quality assurance in AI software. By establishing a clear connection between the origins of defects and their severity outcomes, as well as the quality characteristics affected, AIODC empowers development teams to focus on defects that present the greatest operational and safety hazards. For instance, defects categorized under Thinking that compromise trustworthiness or accuracy may require urgent intervention in healthcare AI, whereas defects arising during the Learning phase that reduce efficiency might be prioritized in large-scale industrial AI applications where computational expenses are paramount. Additionally, the organized AIODC framework could be incorporated into continuous integration workflows, facilitating real-time assessments of defect risks and proactive quality management in AI development.

From a methodological standpoint, the introduction of a new AI attribute (Data, Learning, Thinking, Not Related) alongside a five-tier severity scale offers a more comprehensive and nuanced defect profile compared to current ODC adaptations. The capacity to distinguish AI-specific challenges from general software defects enables more focused debugging approaches and supports the development of specialized test cases for data management, model training, and inference processes. The identified correlation between AI attribute categories and severity levels strengthens the hypothesis that the type of defect serves as an indicator of potential harm, which is essential in risk-based testing and quality strategy formulation.

The classification of impact utilizing Kharchenko's AI/AIP quality model has also uncovered persistent failure patterns: defects related to Learning were found to diminish both efficiency and robustness, whereas defects associated with Thinking had a disproportionate effect on trustworthiness and accuracy. This observation reinforces the architectural separation between the training and inference subsystems of AI and indicates that measures for defect prevention ought to be tailored to each subsystem. For example, tools for automated training monitoring may prove to be more effective in identifying anomalies during the Learning phase, while audits focused on explainability could more effectively address irregularities occurring in the Thinking phase.

From a theoretical perspective, this research adds to the developing domain of Software Engineering for AI by transforming quality models into a practical framework for defect

analysis. Although previous studies have suggested quality metrics or general defect taxonomies, AIODC serves to connect abstract quality attributes with tangible workflows for defect classification. This linkage is crucial for enhancing quality assurance processes in AI initiatives, where the detection of defects is complicated by non-deterministic behaviors and evolving models.

6.1. Limitations

This study is constrained by a relatively small sample size and its concentration on a singular AI framework (Keras). While this approach reduces confounding variables like cross-platform duplication, it limits the generalizability of the findings. Subsequent research should assess AIODC across more frameworks (e.g., TensorFlow, PyTorch) and larger datasets to verify its scalability and robustness.

7. Conclusion and prospective

Artificial Intelligence systems are predominantly deployed as critical software solutions across a wide range of domains, making the identification and resolution of software defects essential to ensure system quality.

This study presented AIODC, a defect classification framework specifically tailored to the unique nature of AI-based systems. Through extending ODC with AI-specific attributes (Data, Learning, and Thinking), a five-level severity scale, and impact mapping based on AI/AIP quality model, AIODC offers a structured methodology for analyzing AI defects.

The analysis of 42 Keras defects substantiated the frameworks feasibility and revealed consistent correlations among defect origins, severity, and quality impacts. Defects related to Learning were found to be the most common and were linked to greater severity, while defects associated with Thinking primarily affected trustworthiness and accuracy.

Future research will aim to validate AIODC using larger and more diverse datasets, incorporate it into automated CI/CD pipelines for real-time defect classification, and investigate the application of large language models for the automation of annotation and analysis.

CRediT authorship contribution statement

The author is responsible for the conceptualization of this research work. He created the framework, prepared the figures of the manuscript, wrote the main manuscript, and reviewed the manuscript draft.

Declaration of competing interest

The author certifies that he has no affiliations or involvement in any organization or entity with any financial or non-financial interest in the subject matter or materials discussed in this manuscript.

Data and code availability

The full dataset and reproduction package used in this study – including the curated set of 42 Keras defects and all AIODC classification materials – are publicly available on Zenodo at: https://doi.org/10.5281/zenodo.17843077.

Funding

No funds, grants, or other support was received for this work.

References

[1] M.O. Alannsary, *Quality improvement of SaaS (Software as a Service) in the Cloud*, Ph.D. dissertation, Southern Methodist University, 2016.
[2] J. Tian, *Software quality engineering: Testing, quality assurance, and quantifiable improvement*. John Wiley and Sons, 2005.
[3] P.J. Denning, "What is software quality?" *Communications of the ACM*, Vol. 35, No. 1, 1992, pp. 13–15.
[4] R. Chillarege, I.S. Bhandari, J.K. Chaar, M.J. Halliday, D.S. Moebus et al., "Orthogonal defect classification – A concept for in-process measurements," *IEEE Transactions on software Engineering*, Vol. 18, No. 11, 1992, pp. 943–956.
[5] L. Ma and J. Tian, "Web error classification and analysis for reliability improvement," *Journal of Systems and Software*, Vol. 80, No. 6, 2007, pp. 795–804.
[6] M. Alannsary and J. Tian, "Cloud-ODC: Defect classification and analysis for the Cloud," in *Proceedings of the International Conference on Software Engineering Research and Practice (SERP)*, 2015, p. 71.
[7] S. Martínez-Fernández, J. Bogner, X. Franch, M. Oriol, J. Siebert et al., "Software engineering for AI-based systems: A survey," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, Vol. 31, No. 2, 2022, pp. 1–59.
[8] S.C.G. Varma, "Bridging the gap between traditional software engineering and modern AI development practices," *International Journal of Emerging Trends in Computer Science and Information Technology*, Vol. 4, No. 3, 2023, pp. 32–40.
[9] B. Gezici and A.K. Tarhan, "Systematic literature review on software quality for AI-based software," *Empirical Software Engineering*, Vol. 27, No. 3, 2022, p. 66.
[10] M.A. Ali, N.K. Yap, A.A.A. Ghani, H. Zulzalil, N.I. Admodisastro et al., "A systematic mapping of quality models for AI systems, software and components," *Applied Sciences*, Vol. 12, No. 17, 2022, p. 8700.
[11] V. Kharchenko, H. Fesenko, and O. Illiashenko, "Quality models for artificial intelligence systems: Characteristic-based approach, development and application," *Sensors*, Vol. 22, No. 13, 2022, p. 4865.
[12] V. Lenarduzzi, F. Lomio, S. Moreschini, D. Taibi, and D.A. Tamburri, "Software quality for AI: Where we are now?" in *Software Quality: Future Perspectives on Software Engineering Quality: 13th International Conference, SWQD 2021, Vienna, Austria, January 19–21, 2021, Proceedings 13*. Springer, 2021, pp. 43–53.
[13] L. Bailey, *The Impact of AI on Software Development*, Ph.D. dissertation, Worcester Polytechnic Institute, 2024.
[14] H. Wang and L. Yuan, "Software engineering defect detection and classification system based on artificial intelligence," *Nonlinear Engineering*, Vol. 11, No. 1, 2022, pp. 380–386.

[15] M. Pandit, D. Gupta, D. Anand, N. Goyal, H.M. Aljahdali et al., "Towards design and feasibility analysis of DePaaS: AI based global unified software defect prediction framework," *Applied Sciences*, Vol. 12, No. 1, 2022, p. 493.
[16] A. Tosun, A. Bener, and R. Kale, "AI-based software defect predictors: Applications and benefits in a case study," in *Proceedings of the AAAI conference on artificial intelligence*, Vol. 24, No. 2, 2010, pp. 1748–1755.
[17] J. Pachouly, S. Ahirrao, K. Kotecha, G. Selvachandran, and A. Abraham, "A systematic literature review on software defect prediction using artificial intelligence: Datasets, data validation methods, approaches, and tools," *Engineering Applications of Artificial Intelligence*, Vol. 111, 2022, p. 104773.
[18] R.G. Hussain, K.C. Yow, and M. Gori, "Leveraging an enhanced codebert-based model for multiclass software defect prediction via defect classification," *IEEE Access*, 2025.
[19] H.V. Pandhare, "Future of software test automation using AI/ML," *International Journal Of Engineering And Computer Science*, Vol. 13, No. 05, 2025.
[20] X. Dong, J. Wang, and Y. Liang, "A novel ensemble classifier selection method for software defect prediction," *IEEE Access*, 2025.
[21] S. Stradowski and L. Madeyski, "'Your AI is impressive, but my code does not have any bugs' Managing false positives in industrial contexts," *Science of Computer Programming*, 2025, p. 103320.
[22] X. Chen, X. Hu, Y. Huang, H. Jiang, W. Ji et al., "Deep learning-based software engineering: Progress, challenges, and opportunities," *Science China Information Sciences*, Vol. 68, No. 1, 2025, pp. 1–88.
[23] A. Güzel and A. Egesoy, "Development of an AI-driven model for advancing software engineering practices," *International Journal of Innovative Research in Computer Science and Technology*, Vol. 13, No. 1, 2025, pp. 1–11.
[24] B. Vinayagasundaram and S. Srivatsa, "Software quality in artificial intelligence system," *Information Technology Journal*, Vol. 6, No. 6, 2007, pp. 835–842.
[25] C. Tao, J. Gao, and T. Wang, "Testing and quality validation for AI software – perspectives, issues, and practices," *IEEE Access*, Vol. 7, 2019, pp. 120 164–120 175.
[26] M.M. Morovati, A. Nikanjam, F. Khomh, and Z.M. Jiang, "Bugs in machine learning-based systems: A faultload benchmark," *Empirical Software Engineering*, Vol. 28, No. 3, 2023, p. 62.
[27] T. Zhang, D. Han, V. Vinayakarao, I.C. Irsan, B. Xu et al., "Duplicate bug report detection: How far are we?" *ACM Transactions on Software Engineering and Methodology*, Vol. 32, No. 4, 2023, pp. 1–32.
[28] R.A. Radice and R.W. Phillips, *Software engineering: An industrial approach*, Vol. 1. Prentice-Hall, Inc., 1988.
[29] V. Lenarduzzi and M. Isomursu, "AI Living Lab: Quality assurance for AI-based health systems," in *IEEE/ACM 2nd International Conference on AI Engineering – Software Engineering for AI (CAIN)*, 2023, pp. 86–87.
[30] International Organization for Standardization, *Medical devices – Application of risk management to medical devices*, Std. ISO 14 971:2019, 2019.
[31] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman et al., "Concrete problems in AI safety," *arXiv preprint arXiv:1606.06565*, 2016.

Authors and affiliations

Mohammed Alannsary
e-mail: alannsary@hotmail.com
ORCID: https://orcid.org/0000-0003-2254-1441
Department of Digital Transformation Programs,
The Institute of Public Administration, Saudi Arabia

e-Informatica Software Engineering Journal
Volume 20, Issue 1, pages: 1–27
DOI: 10.37190/e-Inf260103
BibTeX

Bridging Humans and LLMs: Investigating Human-AI Collaboration in Multi-agent Requirements Analysis for Organizational AI Adoption

Malik Abdul Sami*, Zheying Zhang*, Muhammad Waseem, Kai-Kristian Kemell, Zeeshan Rasheed, Tomas Herda, Pekka Abrahamsson
*Corresponding authors: malik.sami@tuni.fi, zheying.zhang@tuni.fi

Article info

Dataset link: https://github.com/GPT-Laboratory/multiagent-story-generation

Keywords:
large language model (LLM)
multi-agent systems
requirements analysis
organizational ai adoption
strategic planning
case study
human-AI collaboration
human-in-the-loop

Submitted: 31 Jul. 2025
Revised: 12 Nov. 2025
Accepted: 4 Jan. 2026
Available online: 7 Jan. 2026

Abstract

Context: Organizations adopting Artificial Intelligence (AI) face challenges in eliciting and analyzing requirements that align with strategic objectives, especially when human oversight and iterative refinement are needed. Large Language Models (LLMs)-based Multi-agent systems provide a potential solution by supporting structured and collaborative Requirements Engineering (RE) processes for AI adoption planning.
Objective: The objective of this study is to investigate whether a multi-agent system, built on LLMs and supported by human input, can assist in requirements analysis for AI adoption.
Method: We used a mixed-method approach: (i) designed and developed a multi-agent system to support the generation and prioritization of requirements for AI adoption, (ii) conducted multiple case studies with four companies to evaluate the system, and (iii) collected data through post-session questionnaires from nine participants and follow-up interviews, one per company.
Results: Questionnaire and interview findings together indicate that the system may assist in identifying relevant and goal-aligned requirements. Seven participants considered the generated requirements relevant, and six found them aligned with organizational goals. Participants noted that iterative feedback improved completeness and feasibility, often within two feedback rounds. Both data sources show that human input was essential to clarify technical details, ensure contextual accuracy, and validate prioritization results. Participants from all companies also identified usability, transparency, and scalability as areas requiring further refinement for broader organizational use.
Conclusions: LLM-based multi-agent systems can support strategic AI planning by enabling iterative refinement with human experts. Future work will include more interviews with stakeholders and adjustments to system features to improve transparency, usability, and scalability.

© 2026 The Authors. Published by Wrocław University of Science and Technology Publishing House.
This is an open access article under the CC BY-SA 4.0 license international.

1. Introduction

Organizations across diverse sectors increasingly integrate Artificial Intelligence (AI) into their strategic planning. This integration aims to achieve goals such as improving operational efficiency, creating new business models, or enhancing customer experience [1, 2]. AI technologies are used in industry, but advances in Generative AI (GenAI) have accelerated organizational adoption efforts due to its reasoning capabilities and its potential for integration into workflows and systems [3, 4].

Despite growing interest, companies face various challenges in trying to adopt GenAI tools in real-world contexts. These include concerns about data privacy and the perceived unpredictability of AI model outputs [5, 6]. In this study, we utilize GenAI's natural language processing capabilities to simulate stakeholders and generate interactive multi-agent dialogues. This enables functions such as summarization, clarity checking, classification, and prioritization within requirements analysis. While high-level AI strategy can guide organizational efforts, adopting concrete tools to realize these strategies requires rigorous planning, with Requirements Engineering (RE) providing structured processes for identifying, analyzing, and prioritizing requirements that support organizational AI strategies [7].

RE plays a key role in the early planning phase of AI adoption by helping companies articulate and prioritize needs, constraints, and concerns [8, 9]. Specifying requirements for AI-based systems includes unique challenges that differ from traditional software development [10]. These include ensuring data availability and quality [11] and validating and interpreting model behavior [12].

Recent advances in LLMs show potential for supporting RE tasks such as elicitation, specification, and prioritization [13, 14]. Prior work has explored multi-agent LLM frameworks for generating user stories and analyzing requirements [14]. Zhang et al. [14] showed how LLM agents can collaborate to improve user story quality. Role-based multi-agent frameworks such as the multi-agent collaboration framework for RE (MARE) [15] have been proposed, but their integration into company workflows, particularly for strategic AI planning and Human-in-the-Loop (HITL), remains underexplored [11]. Multi-agent approaches can improve coordination, adaptability, and human oversight in planning tasks. In industrial contexts, agentic systems have supported monitoring, scheduling, and dependency management in manufacturing operations, improving responsiveness and reducing downtime. Similarly, in enterprise sales, agent-based workflows help organize proposal generation and stakeholder communication, supporting human-led validation [16].

This paper extends our earlier work [17], which introduced a LLM-based multi-agent system for user story generation and prioritization. Building on that foundation, the present study extends the approach from software-level user stories to organizational-level requirements. The goal is to explore how multi-agent system can support human–AI collaboration in company-level AI adoption, where requirements represent strategic goals rather than product features. We propose a system that supports strategic AI planning by simulating multiple stakeholder roles and implement it.

In the implemented system, each LLM-based agent is assigned a stakeholder-inspired role such as Product Owner, Compliance Officer, or AI Strategist. The agents collaborate to generate and prioritize high-level requirements for AI adoption through two core activities: (i) generating requirements based on predefined role instructions and (ii) prioritizing these requirements using ranking methods. Stakeholders configure the roles, review the outputs, and refine the requirements to ensure alignment with company strategy.

The research is guided by three questions:

- **RQ1:** How can a multi-agent system be designed to support companies in analyzing requirements for strategic AI adoption planning?
- **RQ2:** To what extent does human involvement influence the quality, relevance, and prioritization of AI-generated requirements for strategic AI adoption?
- **RQ3:** What challenges, limitations and, improvement suggestions do companies identify when using the multi-agent system for AI adoption planning?

To answer these questions, we designed multi agent system and conducted a multiple-case study in four companies. We used a mixed-methods approach, combining post-session questionnaires and semi-structured interviews to capture participant feedback and contextual insights. The core contributions of this research are:

- We propose and evaluate a novel LLM-based multi-agent system for supporting strategic AI adoption planning through requirements generation and prioritization.
- We provide insights and lessons learned on human-AI collaboration, system usability, and practical challenges based on a multi-case study across four companies.

The rest of the paper is structured as follows. Section 2 provides the background and related work. Section 3 presents the research methodology, which consists of two parts: (A) the design and implementation of the proposed multi-agent system and (B) its evaluation through multi-case studies, detailed in Section 4. Section 5 reports the results from the four companies. Section 6 discusses the implications of the findings. Section 7 presents the threats to validity. Finally, Section 8 concludes the paper.

2. Background

This study examines strategic organizational requirements for AI adoption, supported by a multi-agent system that combines generative AI and human-in-the-loop collaboration.

2.1. Organizational AI adoption and challenges

Adopting AI in organizations extends beyond technical implementation. It requires strategic alignment, data governance, and readiness across departments [18–21]. Smit et al. [20] emphasize that trust, data quality, and stakeholder coordination are essential for effective AI planning. Russo [6] formalizes these relationships through the Human–AI Collaboration and Adaptation Framework, showing that workflow compatibility is the strongest determinant of GenAI adoption, outweighing traditional factors such as perceived usefulness or social influence. His findings demonstrate that integration success depends on how well AI tools align with existing processes and roles rather than on the technology itself. Ahmad et al. [8] similarly note that conventional RE tools remain ill-suited for human-centered AI projects, where reasoning and contextual interpretation are required. Leadership commitment, shared understanding, and collaboration remain critical to sustainable AI integration [22, 23].

Empirical evidence from industrial studies confirms these challenges. Kemell et al. [5] identify four recurring barriers in European software companies: data privacy and regulation, unclear value measurement, fragmented tool ecosystems, and limited employee guidance. Vaz Pereira et al. [24] report similar issues in a large media company, where developers valued GenAI for automation but expressed concerns about reliability, data security, and skill erosion. Flyckt et al. [25] highlight comparable barriers in manufacturing firms, including weak data governance, limited explainability, and lack of process integration.

The evidence demonstrates that adopting AI is not only a technical task but a strategic transformation that affects structures, roles, and practices across the organization. Consequently, requirements analysis in this context must extend beyond defining product features and instead capture organizational changes, governance mechanisms, and activities that enable sustainable AI adoption. Such analysis involves multiple stakeholders in different departments of an organization, and coordination and shared understanding are essential. This calls for tools that support strategic planning at the organizational level rather than only specification at the system level. LLM-based multi-agent systems provide a promising means to address this need. By facilitating multi-level requirements analysis to align with diverse stakeholder inputs, they can help organizations plan AI adoption in a transparent and coordinated manner

2.2. Generative AI and multi-agent collaboration in requirements engineering

Building on the organizational context, GenAI provides mechanisms to support RE by generating, summarizing, and refining textual content for tasks such as elicitation, prioritization, and validation. Large Language Models (LLMs) can generate user stories and related artifacts in zero- and few-shot settings [26, 27]. Ferrari et al. [28] note that while LLMs can support RE tasks, their outputs raise concerns about correctness and trustworthiness. They propose using formal methods to ensure reliable and verifiable results. Abed et al. [29] find that AI-generated user stories improve through human review.

To extend the capabilities of GenAI in RE, multi-agent systems can coordinate autonomous entities with defined roles. In RE, such systems distribute tasks like elicitation, refinement, verification, and prioritization among agents. MARE [15] defines role-based RE agents for negotiation and validation. LDB [30] supports debugging through verification-based reasoning. Hong et al. [31] developed a GPT-based system for requirements refinement and code generation. Sanwal et al. [32] proposed a pipeline for user story generation and sprint planning with LLM agents. These studies show the potential of multi-agent collaboration but remain limited to simulated or academic contexts.

Human involvement improves reliability and oversight in AI-assisted RE. Hymel et al. [33] report that users find it difficult to assess requirement correctness without feedback. Ferrari and Spoletini [28] suggest integrating LLMs with formal methods to improve accountability. Ahmad et al. [8] note that current RE tools do not address human-centered aspects in AI projects. This highlights the need for human-in-the-loop (HITL) mechanisms that maintain stakeholder understanding and alignment. Few studies have explored LLM-based multi-agent systems with HITL refinement in real organizations.

Table 1 summarizes recent multi-agent frameworks proposed for requirements analysis and software development. These frameworks differ in scope, coordination method, evaluation type, and the extent of human-in-the-loop (HITL) integration. Most remain conceptual or simulation-based, with no industrial validation. Only a few include partial human feedback or traceability mechanisms. Frameworks such as MARE [15], AutoGen-based systems, and LDB [30] define agent roles for elicitation and collaborative debugging but have been tested only in academic settings. Similarly, ProAgent [34], ALAS [14], Goal2Story [35], MAGIS [36], and RTADev [37] focus on conceptual design or prototype-level demonstrations without industrial application or systematic human validation.

Existing RE approaches provide limited support for the human, technical, and strategic dimensions of AI adoption planning within organizations [8]. Addressing this gap requires methods that combine automation with human oversight to enable multi-level analysis

Table 1. Comparison of multi-agent approaches in requirements engineering

Framework	Scope	Method	Evaluation type	Industrial validation	HITL
MARE [15]	Multi-agent RE framework	Role-based coordination and task assignment	Conceptual	No	Not addressed
LDB [30]	Debugging and elicitation	Verification-based reasoning and error tracing	Prototype	No	Partial reasoning trace
ProAgent [34]	Cooperative agent orchestration	Task coordination with limited feedback	Simulation	No	Limited visibility
ALAS [14]	Designed agent roles for automated user story enhancement	Role-based coordination and task assignment	Prototype	No	Not addressed
Goal2Story [35]	Goal-driven RE via Impact Mapping	LLM-based user story generation with adaptive prompts	Prototype	No	Partial HITL traceability
MAGIS [36]	Issue resolution and maintenance planning	Role-specialized agents for planning, coding, and QA review	Prototype	No	Not addressed
RTADev [37]	Software development pipeline	Multihyp agent consensus and alignment-checking process	Simulation	No	Not addressed
Our study	Requirements generation and prioritization	Role-based multi-agent system with iterative feedback	Empirical multi-case	Yes	HITL multi-agent collaboration

and informed decision-making. This study empirically evaluates a role-based multi-agent system that supports collaborative requirements analysis and prioritization in strategic AI adoption. Human-in-the-loop feedback is integrated throughout to refine outputs and align them with organizational objectives. Evaluation across four companies demonstrates practical applicability and identifies key challenges in human–AI collaboration within RE. To the best of our knowledge, this is the first empirical evaluation of a multi-agent RE tool in industrial settings.

3. Research methodology

The methodology consists of two sections: (A) the design and implementation of the proposed system and (B) its evaluation through multi-case studies across four organizations. Part A describes the system architecture, implemented features, and technical components. Part B outlines the case study design, including case selection, data collection, and analysis procedures. Figure 1 illustrates the methodology, highlighting the system lifecycle (design and development) and its subsequent evaluation in real-world settings.

3.1. Proposed and implemented system

3.1.1. System design

Human-in-the-loop workflow: As illustrated in Figure 2, the framework comprises multiple LLM-based agents, each assigned a distinct role for task completion. They collaborate by exchanging intermediate outputs and iteratively refining their responses based on

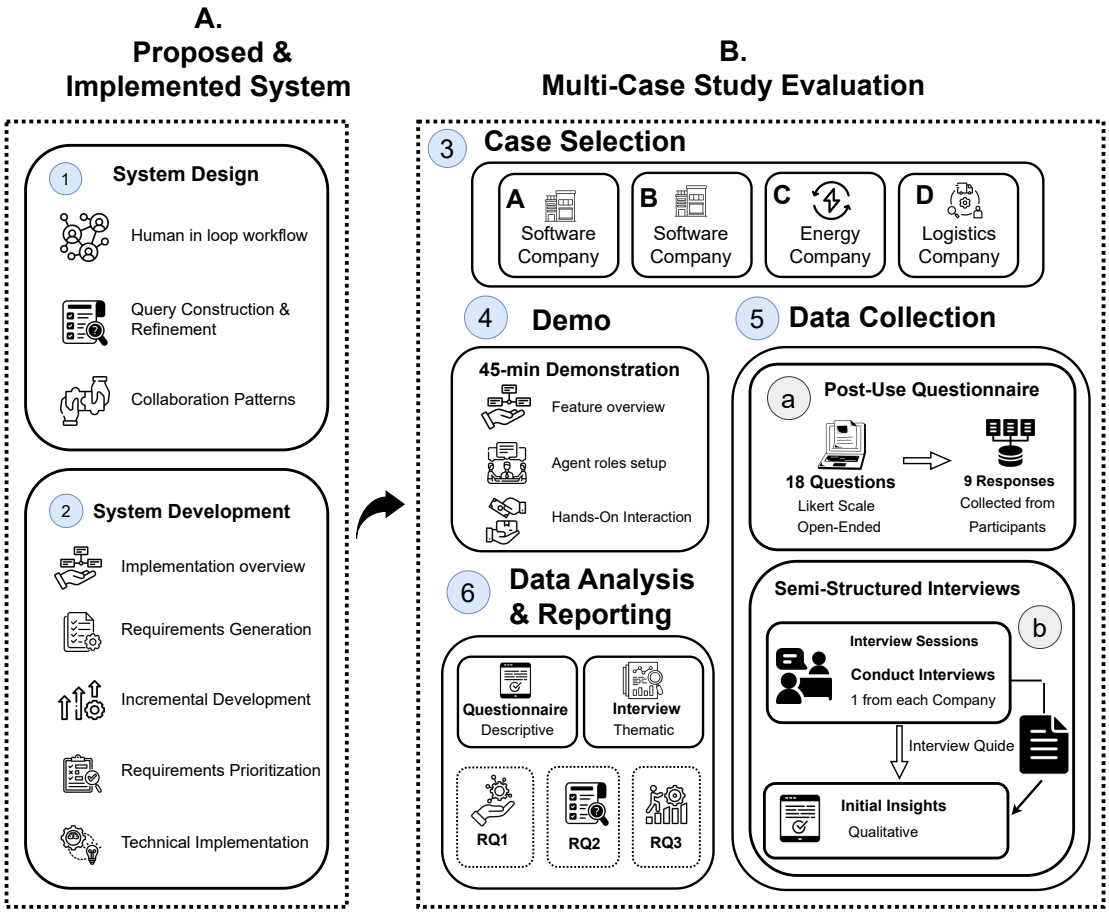


Figure 1. Overview of the research methodology, illustrating (A) system design and implementation and (B) multi-case study evaluation process

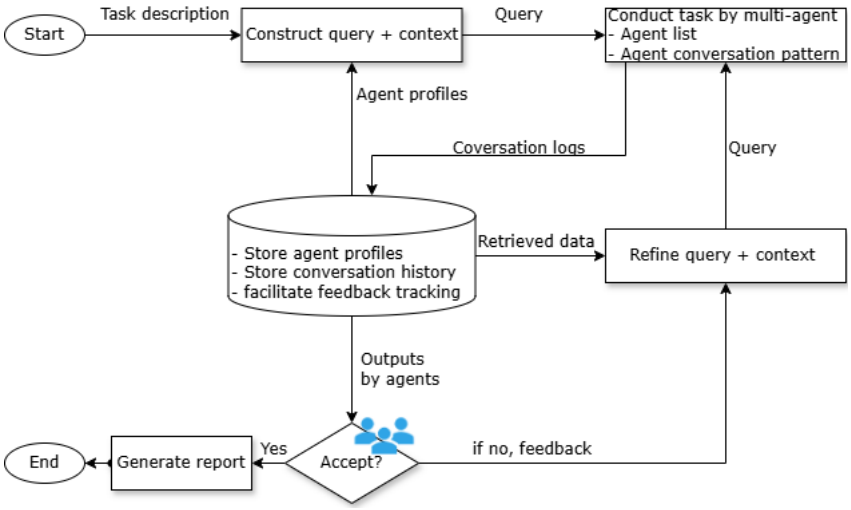


Figure 2. Human intervention with LLM agents for task completion

human feedback. Integrating human feedback into the response refinement process fulfills an effective HITL mechanism in improving the quality of task outputs.

The process begins with a task description and associated contextual information, which are transformed into a structured query. This query is then processed by multiple agents, where agents interact according to a predefined interaction pattern to complete the task. The generated outputs are evaluated by human users. If the outputs are accepted, a final requirements are saved to system; otherwise, user feedback is used to revise the query and the context, and the process iterates again.

The framework includes a data store that records and maintains agent profiles, conversation logs, user feedback, and history outputs. The repository supports iterative query refinement by enabling the system to retrieve relevant contextual information and incorporate user feedback effectively. This framework supports dynamic agent interaction and continuous outputs improvement through feedback loops. The following subsections detail how queries are constructed and refined for task completion, and the collaborative patterns agents use to execute these queries.

Query construction and refinement: The proposed framework relies on prompt engineering to define and control how agents perform their assigned tasks. Prompts serve as the primary mechanism for guiding LLMs in role interpretation, contextual input processing, and response generation [34]. To balance system-wide consistency with agent-specific customization, the structure of queries sent to the agents comprises two types of prompts, i.e. user prompts and system prompts.

- **User prompts** specify agent-specific roles, task descriptions, and contextual information relevant to each task.
- **System prompts** specify shared instructions applicable to all agents involved in a task, including the response structure, tone, output constraints, and formatting conventions.

At runtime, the user and system prompts are merged into a single query that enables differentiated agent behavior without modifying the underlying model parameters [38]. If the output generated by agents fails to meet user expectations, a query refinement is triggered. Rather than reconstructing a new query from scratch, the framework reuses the original query and adds user feedback to improve the resulting response. Therefore, the refinement query incorporates two additional components, i.e. previous output and user feedback.

- **Previous outputs** are responses generated by agents based on the original query.
- **User feedback** includes clarifications, suggestions, or corrections provided by the user to guide the output enhancement.

The updated prompt may optionally revise the original task, role assignments, or contextual information based on the feedback. All other components remain consistent with the initial prompt structure. The iterative human-in-the-loop mechanism is expected to support the incremental enhancement of agents' outputs so that the framework can improve the responsiveness to evolving user feedback and enhance the quality and relevance of task execution.

Multi-agent collaboration patterns: The agents in the proposed framework follow multi-agent design principles and support multiple interaction patterns, each aligned with a general strategy for coordinating task execution [39, 40]. Upon task initiation, the user identifies a group of agents based on the nature of the task and the roles required for task completion. Each agent operates under a predefined role and a corresponding set of instructions specified within the query. The selection of conversation pattern is decided by the degree of inter-agent dependency needed for task completion. As illustrated in Figure 3, we use three multi-agent collaboration patterns from the existing literature [41]. When three agents execute a task, the system supports three collaboration patterns: role-delegation, peer coordination, and independent execution. The role-delegation pattern

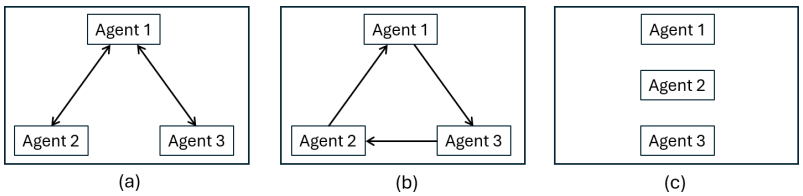


Figure 3. Collaboration patterns between agents: (a) role delegation, (b) peer coordination, and (c) independent interaction

follows a hierarchical structure. A commander agent (Agent 1) receives the task input, assigns sub-tasks to subordinate agents (Agent 2 and Agent 3), and synthesizes their outputs into a final result. This pattern fits modular tasks with minimal inter-agent dependency.

The peer coordination pattern enables decentralized collaboration, where all agents communicate directly, exchange intermediate results, and refine responses through dialogue. It is ideal for tasks requiring consensus, comparison, or integration of diverse viewpoints, such as eliciting and analyzing requirements from multiple stakeholders.

The independent execution pattern involves agents working autonomously. Each receives the same input, performs its task independently, and produces a separate output. This pattern fits tasks requiring parallel analysis. These configurable collaboration patterns provide flexibility in orchestrating agent workflows and enable the system to accommodate a broad range of task structures, from tightly integrated to fully independent processes.

3.1.2. System development

To demonstrate the implementation of the system and evaluate its use in practice, we developed a multi-agent system to support human-in-the-loop strategic AI planning. The system focuses on high-level strategic organizational requirements that support decision-making for strategic AI adoption. As shown in Figure 4, the system addresses two core tasks: (1) requirements generation and (2) requirements prioritization. Requirements are represented as user stories, which facilitate communication of stakeholder expectations and support agile, incremental planning even at strategic levels. This approach bridges high-level organizational goals with concrete implementation details.

Task 1: Requirements generation. Requirements generation follows a role delegation pattern. Senior stakeholders, such as executives, strategic planners, and IT leaders, initiate this task by providing strategic context through a vision statement and a process-oriented Minimum Viable Product (MVP) [42]. The MVP outlines an iterative roadmap emphasizing processes, stakeholder interactions, governance structures, and scalability pathways rather than specific technological solutions. Figure 5 presents excerpts from the vision statement and MVP. Senior stakeholders also define strategic AI planning roles and required expertise.

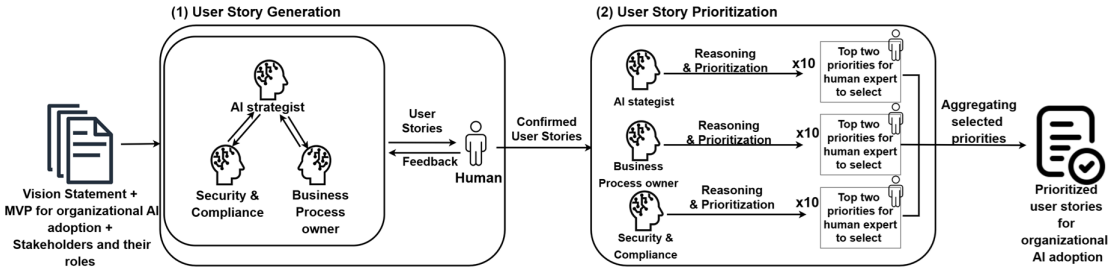


Figure 4. AI-driven multi-agent system for requirements analysis and prioritization

Vision and MVP for Stakeholder-Driven Strategic AI Planning

Vision statement
Our vision is to transform our IT company into a leading AI-enabled organization that leverages cutting-edge artificial intelligence to optimize processes, empower data-driven decision making, and deliver exceptional value to stakeholders. By implementing a stakeholder-driven, LLM-based multi-agent approach to requirements analysis for each AI adoption use case, we ensure that every AI solution addresses real business challenges and evolves with the dynamic market and technological landscape…

MVP
Minimum Viable Plan (MVP) for AI-Enabled Organizational Transformation. 1. Vision Statement: Transform our IT company into a leading AI-enabled organization that leverages cutting-edge artificial intelligence to optimize processes, empower data-driven decision making, and deliver exceptional value to stakeholders. 2. Strategic Objectives: Establish a structured AI adoption framework to guide investments, develop a stakeholder-driven AI requirements process, implement governance and ethical frameworks, and define a roadmap for scalable AI integration across business units. 3. Stakeholders and Their Goals: Executives align initiatives with business strategies, IT Leaders develop scalable infrastructure, Data Scientists optimize AI models, Compliance Officers ensure governance, and Business Units leverage AI for decision intelligence and process automation. 4. MVP Scope (Initial Phase): Deploy a stakeholder-driven, LLM-based multi-agent system for AI use case analysis, identify and prioritize initiatives using structured criteria, and develop governance policies. 5. Success Metrics: Strategic AI Integration Index, AI Governance Readiness Score, Business Impact Metrics, and Scalability Readiness. 6. Iterative Roadmap: Phase 1 involves requirements elicitation and prioritization with LLM agents, while Phase 2 focuses on feasibility assessment and scalable implementation…

Figure 5. Vision and MVP for stakeholder-driven strategic AI planning

Agents such as AI Strategist, Compliance Officer, and Business Owner are then assigned to generate candidate user stories reflecting business value, regulatory compliance, and technical feasibility. The AI Strategist coordinates and disseminates contextual information to subordinate agents, enabling parallel outputs. Human users review, refine, and confirm these generated requirements to ensure alignment with organizational objectives. Table 2 summarizes the agent roles implemented in the system.

Table 2. LLM-based agent roles and corresponding responsibilities and tasks

Agent	Role	Responsibilities	Tasks
AI Strategist	Lead strategic agent	Identify high-impact AI use cases, align initiatives with strategic goals, and foster cross-departmental collaboration. Continuously reassess and refine AI direction to stay ahead in a rapidly evolving landscape.	Translate business challenges into AI opportunities; generate and prioritize requirements related to strategic alignment and innovation; coordinate agent activities across the system.
Security and Compliance Officer	Regulatory oversight agent	Ensure AI initiatives adhere to legal frameworks and ethical guidelines, mitigating risks related to bias, privacy, security, and transparency.	Identify regulatory constraints; generate compliance-focused requirements; validate system outputs for regulatory alignment and risk mitigation.
Business Owner	Business value agent	Translate operational needs into actionable requirements for AI solutions. Ground initiatives in business processes and practical value. Facilitate collaboration between business and technical stakeholders.	Contribute business-centric requirements; prioritize based on business value; refine based on practical challenges and measurable outcomes.

Task 2: Requirements prioritization. Requirements prioritization is implemented as an independent interaction pattern. The user selects three agents and chooses a prioritization technique, e.g., the Weighted Scoring Model (WSM), Weighted Shortest Job First (WSJF), or the $100 method [43], along with the number of iterations (three to ten). As shown in Figure 4, each agent independently repeats the prioritization process X times using the same prompt, generating one ranking list per iteration. To evaluate the stability of these rankings, the system computes Kendall's Tau correlation for every pair of lists across all iterations. After all iterations, it calculates the average Kendall's Tau distance $(1 - \tau)$ of each ranking to the others [44, 45]. The ranking with the lowest average distance is identified as the most consistent, indicating convergence toward a stable prioritization. For each agent, the system identifies the two most stable rankings, i.e., those with the lowest average Tau distances, thereby filtering out random variation and ensuring reproducible prioritization outcomes. The user then reviews these top two rankings per agent and selects the one that best reflects their judgment.

Users may assign weights to the selected rankings to reflect the relative importance of each agent's viewpoint. For example, the AI strategist may prioritize feasibility and alignment with AI objectives, the business process owner may focus on business value, and the security/compliance specialist may emphasize technical and regulatory constraints. Weights are fully configurable, to select a value from a drop-down, such as 40–35–25, and can be adapted to the decision context. The system aggregates the chosen rankings using the provided weights to produce the final prioritized list. This workflow combines statistical stability analysis with expert input, ensuring that the outcome reflects both data-driven consistency and human judgment. Screenshots and source code for the full workflow are available in the public repository[1].

Technical implementation details: This system extends our earlier prototype [17]. The current implementation comprises a eact frontend and a Python backend built with the Starlette framework [46, 47], using real-time WebSocket communication. The backend manages user sessions, agents profiling, processes OpenAI API calls, and structures the generated outputs. GPT-4o [48] was used for all agent roles through the OpenAI API, selected for its performance and cost efficiency in real-time multi-agent interaction. The temperature parameter was set to 0.1 to reduce generation variability, while other parameters such as top-p, frequency penalty, and presence penalty remained unchanged.

Supabase is used for role-based authentication, and MongoDB stores the application data. The system integrates human-in-the-loop mechanisms that enable iterative feedback during requirements refinement. Agents operate under predefined roles such as AI Strategist, Business Owner, and Compliance Officer.

The multi-agent environment was developed iteratively to support organizational AI adoption. Development progressed through staged integration of agent roles, communication logic, and feedback mechanisms to ensure stable and efficient operation. Its performance was evaluated in earlier work [17], which confirmed technical feasibility and provided the foundation for this study.

4. Multi-case study evaluation

The evaluation aimed to assess how the multi-agent LLM system supports strategic AI adoption planning through human–AI collaboration in the requirements analysis process.

[1] https://github.com/GPT-Laboratory/multiagent-story-generation/tree/main/screenshots

Following the guidelines of Runeson and Höst [49], we conducted case studies in four companies. Data collection included post-use questionnaires and semi-structured interviews. Questionnaire responses ($n = 9$) were analyzed using descriptive statistics, while interview data were analyzed thematically using Braun and Clarke's method [50].

The study timeline was coordinated with the participating organizations. The initial evaluation was conducted in Company D during February and March 2025, during which the system was refined based on participants' feedback. The updated version was then used in the remaining three companies for evaluation in April and May 2025. In total, the study spanned four months, with all interviews completed in May 2025. The following subsections describe the demonstration, data collection process, analysis methods, and data reporting for each research question.

4.1. Case selection

The study was conducted in four organizations: two software consultancies, one energy provider, and one logistics company. Following established case study guidelines [49, 51], we applied a convenience sampling strategy [52], selecting organizations that were both accessible and actively engaged in AI adoption initiatives. The cases were chosen to reflect variation in industry sector, company size, and level of AI maturity.

Each organization agreed to participate voluntarily and nominate individuals involved in AI-related planning or decision-making processes. These included IT leaders, product owners, an agile coach, directors, and development managers responsible for planning or implementation tasks.

Company A is a small software consultancy providing development and technology services to clients in the USA, Pakistan, and the UAE. It uses generative AI tools such as Cursor and Copilot to improve productivity in software development and is exploring AI for product planning, marketing, and client engagement. The main challenge is integrating AI into daily workflows to achieve measurable business impact. Motivated by these goals, the company joined the study to evaluate how the tool could support strategic AI planning and requirements analysis. The Director of Technology (P2) and Product Owner (P1) participated, focusing on AI-assisted requirement elicitation and planning.

Company B is a mid-sized software consultancy providing product development and process automation services in the USA and Pakistan. It uses generative AI tools such as ChatGPT, Codium, Cline, and Cursor AI across departments, with most teams using them to accelerate workflows and improve customer satisfaction. The company is also developing an agent-based HR automation system for CV prioritization and AI-supported interviews. The main challenge is aligning AI initiatives with existing systems and governance processes. Motivated by these goals, the company joined the study to explore how the tool could support AI-assisted prioritization and planning. The IT Leader (P3), Business Process Owner (P4), and IT Leader (P5) participated in the study.

Company C operates in the European energy sector, providing electricity, heating, and renewable energy services. It has an internal software unit managing digital systems and data-driven operations. The company is identifying AI use cases to optimize energy management and improve productivity across departments. About half of the employees use tools such as Copilot and ChatGPT to enhance efficiency and reduce routine work. The main challenge is managing change and ensuring smooth technology adoption. Motivated by these goals, the company joined the study to explore how the tool could support strategic

Table 3. Overview of case companies and participants

Company	Region	Industry	#Employees	Participant (ID, Years of Experience)
A	South Asia	Software Consultancy	10–49	Product Owner (P1, 10+), Director of Technology (P2, 12+)
B	South Asia	Software Consultancy	50–249	IT Leader (P3, 8+), Business Process Owner (P4, 7+), IT Leader (P5, 5+)
C	Europe	Energy Sector	50–249	Development Director (P6, 10+), Development Manager (P7, 10+)
D	Europe	Logistics	499–750	Scrum Master (P8, 5+), AI Strategist (P9, 9+)

AI planning. The Development Director (P6) and Development Manager (P7) participated, focusing on aligning strategic and technical planning.

Company D is an international postal, logistics, and service provider with a strong presence in Central and Eastern Europe. It delivers a broad range of mail, parcel, and freight solutions, supported by dedicated internal software teams responsible for routing, operational efficiency, and service analytics. The organization is committed to high-quality standards and continuously adapts its offerings to meet evolving customer needs and is piloting AI to improve operational efficiency, automation, and decision-making while maintaining compliance with security and legal standards. A key challenge lies in identifying AI use cases with measurable business value and clear return on investment. Motivated by these goals, the company joined the study to explore how the tool could support strategic AI planning and prioritization. The Scrum Master (P8) and AI Strategist (P9) participated in the study, focusing on aligning operational and strategic AI initiatives.

Table 3 summarizes the context of the four participating companies, including their region, industry, size, and participant experience. The cases represent diverse sectors and organizational scales, ranging from small consultancies to large enterprises, with participants holding senior roles in AI and technology management.

4.2. Demo

Following system implementation, participants from all four organizations were introduced to the tool through guided demonstration sessions. Each organization took part in a 45-minute session that included an overview of the system's features, configuration of agents and roles, and hands-on interaction. The purpose of these sessions was to familiarize participants with the system and ensure they were prepared for independent use.

During the demonstration, participants explored key capabilities, including assigning agent roles, generating and refining user stories using given vision and MVP statements, and prioritizing them. This phase served both to collect initial user feedback and to support participants in applying the system autonomously within their own planning workflows.

4.3. Data collection

Post-use questionnaire: In this phase, participants completed a post-session questionnaire comprising 18 questions organized into five sections aligned with the research questions. The first section collected demographic and organizational background information (Q1–Q7). The second and third sections included 16 five-point Likert-scale statements (Q8–Q9, Q12–Q13) examining requirement generation, refinement, prioritization, and strategic alignment. Evaluation followed a subset of requirement quality criteria from ISO/IEC/IEEE

Table 4. Summary of data collection, participants, sessions, and average durations

Instrument	#Participants	#Sessions/#Responses	Avg. Duration
Demonstration and Q&A	9 (P1–P9)	4 sessions (one per company)	45 min
Post-Session Questionnaire	9 (P1–P9)	9 responses	14:24 min
Semi-Structured Interviews	4 (P2, P3, P7, P9)	4 interviews (one per company)	35 min

29148:2011 [53]. These sections measured perceptions of relevance, correctness, completeness, feasibility, transparency, and alignment with organizational goals. The fourth section assessed human–AI collaboration and integration through Likert and open-ended questions (Q13–Q14). The fifth section contained multiple-selection and open-ended questions (Q15–Q17) capturing perceived benefits, challenges, and improvement suggestions, followed by one numerical rating question on overall recommendation likelihood (Q18). The complete questionnaire and interview are available in the project's GitHub repository[2].

Semi-structured interviews: After the post-session questionnaire, we conducted interviews with one participant from each company, invited based on prior questionnaire completion and informed consent. These interviews explored deeper reflections on system usefulness, organizational fit, and suggestions for improvement. Discussions began by establishing organizational context, including current GenAI tool usage, roll out stage, and the participant's role in AI adoption. Participants then shared initial impressions of the system, focusing on ease of use and alignment with strategic planning goals.

The interviews examined the feedback process in detail, including types of revisions to system-generated requirements, number of feedback rounds required, and how human input shaped final outputs. Further, the sessions investigated how well the system fit within organizational workflows, addressing trust in AI-generated logic, team-specific needs, and internal resistance. Finally, participants reflected on the system's overall value and proposed concrete suggestions for improving functionality, usability, and integration.

Table 4 summarizes the data collection process. Each company participated in one demonstration session, followed by a post-session questionnaire completed by nine participants, with an average completion time of approximately 14 min. Four follow-up interviews, one per company, provided deeper insights into system use and organizational fit. Interview durations ranged from 33 to 38 min (P2: 38 min, P3: 33 min, P7: 34 min, P9: 34 min), with an average duration of approximately 35 min.

4.4. Data analysis and reporting

We applied both quantitative and qualitative methods to analyze the collected data, including post-session questionnaire responses and semi-structured interviews. These two sources of data were used for triangulation to strengthen validity and ensure consistency between quantitative and qualitative findings.

We performed descriptive statistical analysis, calculating means and standard deviations to summarize participants' ratings, while frequencies and percentages were used for categorical items. Responses were grouped by research question to examine perceptions of system usefulness, effectiveness of feedback, and integration into organizational workflows.

For qualitative data, we analyzed interview transcripts using thematic analysis following Braun and Clarke's six-phase method [50]. We adopted an inductive, data-driven approach

[2] https://github.com/GPT-Laboratory/multiagent-story-generation/tree/main/instrument

where codes and themes were developed directly from the data. ATLAS.ti[3] was used for coding. Coding decisions were discussed among the research team to ensure consistency. Participant quotations were edited for readability by removing filler words and applying minor grammatical refinements without altering meaning.

The analysis identified one high-level theme for RQ1 (first-use impressions and initial support), two themes for RQ2 (feedback-driven refinement and benefits of human-in-the-loop), and four themes for RQ3 (usability barriers, scalability, trust and transparency, and user suggestions). A total of 37 unique codes were applied to 175 coded quotations across four interviews. The number of quotations per code ranged from 1 to 20. The qualitative findings supported RQ2 and RQ3 by capturing feedback on user trust, refinement processes, human-in-the-loop interactions, integration challenges, and design suggestions. The full list of codes available on GitHub[4].

5. Results

In total, nine participants completed the post-session questionnaire, which included Likert-scale and open-text questions across five sections. Four participants (P2, P3, P7, and P9), one from each company, took part in follow-up semi-structured interviews. The interviews followed RQ-aligned prompts focusing on system use, human-in-the-loop interaction, trust, and improvement suggestions.

Nine participants used the system within their organizational planning workflows. For example, in Company D, participant P8 received 21 initial requirements generated by the system. In the feedback rounds, one duplicate was removed, four overlaps were merged, and missing contextual details were added. At the end, the process produced 16 approved user stories. Table 5 presents representative examples of initial and refined user stories, illustrating how user feedback improved the relevance and precision of outputs. Practitioner comments mainly focused on clarifying scope, removing duplicates, and ensuring alignment

Table 5. Examples of requirements before and after refinement during iterative development

ID	Requirements	Feedback summary	Refined requirements
1	As a compliance officer, I aim to establish AI governance practices to ensure ethical and legal AI use in the organization.	Add reference to applicable regulatory frameworks.	As a compliance officer, I aim to establish AI governance practices aligned with the EU AI Act and internal ethical standards to ensure legal and responsible AI use.
2	As a project stakeholder, I aim to validate AI project priorities through structured review sessions.	Clarify evaluation criteria and link to decision process.	As a project stakeholder, I aim to validate AI project priorities through structured evaluation sessions using defined criteria for business value and feasibility.
3	As an executive, I want to establish a structured AI adoption framework to guide organizational planning.	Add measurable outcome or acceptance criteria.	As an executive, I want to establish a structured AI adoption framework that defines key milestones and measurable outcomes for responsible deployment.
4	As a project stakeholder, I need AI governance policies to guide system design.	Avoid duplication and make scope explicit.	As a project stakeholder, I need AI governance policies specifying data handling, accountability, and audit mechanisms to guide AI system design.

[3] https://atlasti.com/

[4] https://github.com/GPT-Laboratory/multiagent-story-generation

with compliance and strategic context. The complete set of generated and refined stories with feedback notes is available in the project repository.

5.1. System support for requirement analysis (RQ1)

The analysis identifies one main theme: *System Support for Requirement Analysis*. It describes how the system assists users in generating, reviewing, and aligning requirements with organizational goals, showing that it enables structured formulation of requirements while allowing users to provide feedback and maintain control during refinement.

Questionnaire responses indicate generally positive perceptions of the generated requirements. Seven participants (78%) agreed or strongly agreed that the requirements were relevant ($M = 4.2, SD = 0.6$), and six participants (67%) agreed that they aligned with organizational goals ($M = 4.1, SD = 0.7$). Five participants (56%) reported that only minor modifications were needed before using the generated requirements. Overall, the quantitative results suggest that the system was perceived as supportive for early requirement formulation and for maintaining alignment with organizational objectives.

The interview data provide further insight into how this support was experienced in practice. Participants emphasized the relevance and usefulness of the generated requirements. As P2 noted, *"When I started exploring it, it was quite impressive… it gives me some promising results as well."* Pariticipant P3 also highlighted the efficiency of the tool: *"It was very fast in giving the requirements results quickly, at least for short projects.".*

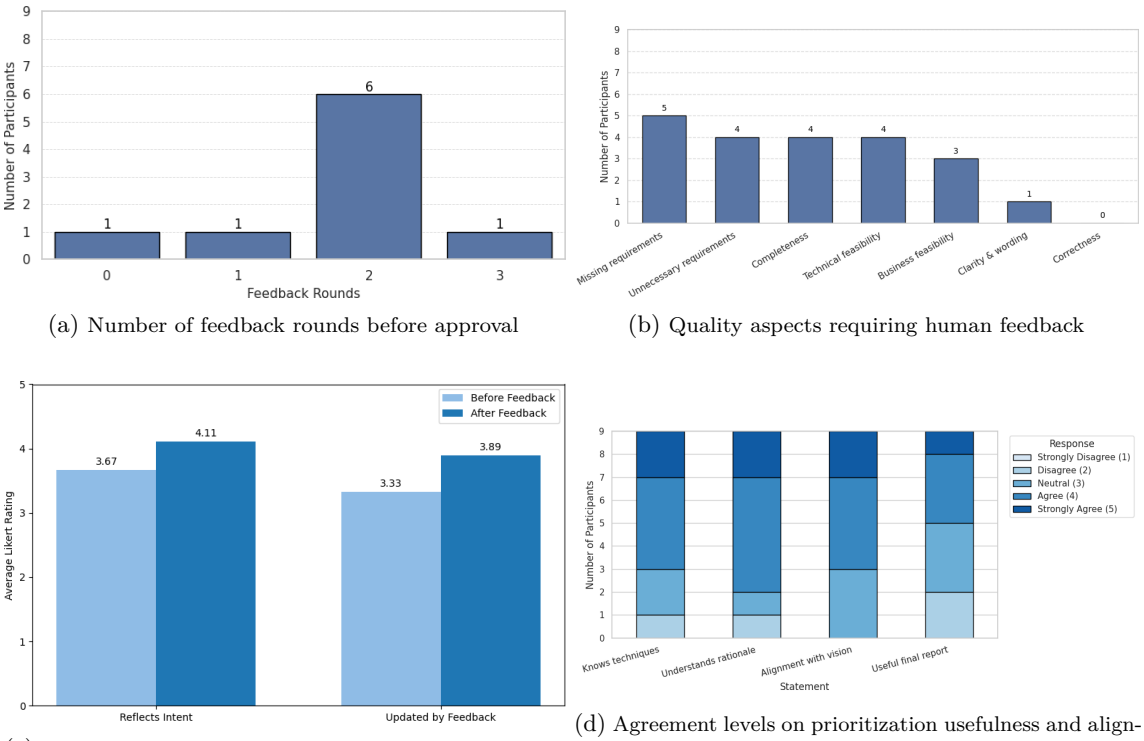
Participants also described how the system contributed to organizational goal alignment, as P7 reflected, *"I was curious about how it would bring more knowledge… it shows us how agents perform different roles."* This indicates that the multi-agent design helped broaden requirement perspectives.

Finally, the interviews also support the finding that only minor refinement was needed. For example, P9 explained, *"I did like the human-in-the-loop… it's possible for a human to adjust the results that are not satisfactory. That was really something good."* This illustrates how feedback mechanisms enabled efficient revision rather than extensive rework. Together, the questionnaire and interview data converge: the system helps generate relevant, goal-aligned requirements while allowing users to refine outputs with minimal effort.

5.2. Human-in-the-loop for refinement and prioritization (RQ2)

This question reports how human–AI interaction supports requirement refinement and prioritization. The analysis integrates quantitative and qualitative evidence to understand how human feedback influences the quality and acceptance of generated requirements. Questionnaire data indicate that iterative feedback improves perceived requirement quality and alignment with project goals.

As illustrated in Figure 6a, six of nine participants (67%) approved the requirements after two feedback rounds ($M = 2.1, SD = 0.9$), indicating that two iterations were generally sufficient to reach an acceptable outcome. The interview data provide explanatory context for this pattern. Participants consistently reported needing two to three refinement cycles, with the exact number depending on the clarity of the initial input. As P2 explained, "*Most of the requirements are at a high level… some take two rounds, and for others more than three.*" P3 similarly observed, "*For the small project, it was just two to three rounds.*" P5 noted that unclear prompts increased iteration effort: "*If the input is vaguely described, the system stays at the same level. The better the preparation, the better the output.*" They



(a) Number of feedback rounds before approval

(b) Quality aspects requiring human feedback

(c) Requirement improvements needing human feedback

(d) Agreement levels on prioritization usefulness and alignment

Figure 6. User-reported feedback cycles and quality aspects: (a) feedback rounds before approval, (b) quality aspects requiring human feedback, (c) requirement improvements, and (d) agreement levels on prioritization outcomes

also observed that longer discussions sometimes led to "*losing track of missing points*" requiring additional iterations.

Moreover, As shown in Figure 6b, missing requirements were reported by 5 participants, unnecessary requirements, completeness, and technical feasibility by 4 participants each, business feasibility by 3 participants, and clarity and wording by 1 participant. No participant mentioned correctness as a concern. These quantitative results suggest that while the system generated technically sound outputs, additional human input was necessary to ensure contextual completeness and feasibility.

Interview data further clarify why such refinements were required. We identified theme "Feedback-driven refinement" where P2 (Company A) explained, *"Some minor improvements will be there, like the technical feasibility should be there, and if I think about the correctness, most of the things are correct, but need to be aligned with the, you know, requirements. And so obviously the human involvement will be there."* Similarly, P3 (Company B) observed, *"Mostly, when the requirements were big enough, then it missed some technical points. So I needed to clarify it to come back to the context. Sometimes it loses focus, so that's what I did."* These insights indicate that participants viewed their role as ensuring contextual and technical alignment, particularly for larger or more complex requirements.

The questionnaire and interview data show that while the system produced accurate and well-structured outputs, users needed to verify and adapt them for organizational and technical relevance. Human review thus complemented system performance by addressing contextual completeness and ensuring feasibility across diverse project scopes.

Table 6. Perceived improvements in requirements before and after human feedback

Requirement quality aspect	Before feedback ($M \pm SD$)	After feedback ($M \pm SD$)	Change in mean
Reflects Intent	3.67 ± 0.47	4.11 ± 0.57	+0.44
Updated by Feedback	3.78 ± 0.63	3.89 ± 0.74	+0.11
Clarity and Wording	4.00 ± 0.47	3.67 ± 0.47	–0.33

Building on the previous findings on feedback rounds and quality aspects, the next analysis examines how participants perceived improvement in the generated requirements before and after human feedback. Figure 6c and Table 6 summarize these perceptions. The average score for "Reflects Intent" increased from $M = 3.67$ ($SD = 0.47$) to $M = 4.11$ ($SD = 0.57$), and "Updated by Feedback" rose from $M = 3.78$ ($SD = 0.63$) to $M = 3.89$ ($SD = 0.74$). "Clarity and Wording" decreased slightly from $M = 4.00$ ($SD = 0.47$) to $M = 3.67$ ($SD = 0.47$), indicating mixed perceptions of phrasing simplicity after refinement. Overall, the quantitative results suggest that participants perceived improvement through feedback, especially in terms of intent reflection and iterative updates.

Interview evidence provides further explanation for these patterns. Participants described the feedback process as useful for removing redundant or unclear requirements while improving overall accuracy. As P9 (Company D) stated, *"It can be time-consuming if there are duplicates. When I was generating some requirements, I got two that were almost the same, which required another round of editing. The human should be in the loop. It improved the output, for example by approving selected requirements or completely removing them from the system."* This comment illustrates how iterative human review contributed to improving output quality through selection and refinement.

Figure 6d shows agreement levels on prioritization usefulness and alignment. Five of nine participants agreed that prioritization outcomes aligned with organizational goals, while four remained neutral. Participants familiar with prioritization techniques (e.g., WSJF, Weighted Scoring, $100 allocation) reported better understanding of the rationale behind prioritization outcomes. Interview data further illustrate this point. As P7 (Company C) stated, *"In some cases, it's good to have the human opinion in your decision-making process, of course. But it's sometimes good to have also that… nonhuman opinion,… think again about your own idea of prioritization and gives another perspective."* This reflects how human-in-the-loop prioritization encouraged reflection and comparative reasoning, allowing participants to validate and reassess their prioritization logic in relation to the system's suggestions.

Overall, the questionnaire and interview data indicate a consistent pattern. Two feedback rounds were sufficient for most participants (6 of 9) to finalize requirements. Perceived quality improved after feedback, particularly in capturing intent and integrating user input. Interview insights confirm that human review addressed omissions and redundancies, enhanced completeness and feasibility, and guided prioritization. Collectively, the findings demonstrate that human–AI collaboration strengthened refinement while maintaining contextual accuracy and user control.

5.3. Challenges and suggestions (RQ3)

The analysis identified challenges and improvement areas for system adoption, as illustrated in Figure 7. Four participants selected *employee resistance to adoption* and *difficulty customizing the system* as the most frequent issues. *Lack of transparency in prioritization*, *trust in model output*, and *AI–human misalignment* were each mentioned by three participants.

Two participants reported *difficulty integrating feedback*, and one noted *too many refinement iterations*. These results indicate that challenges were mainly human and organizational rather than technical. The higher frequencies for resistance and customization suggest that usability and workflow alignment were central to adoption concerns.

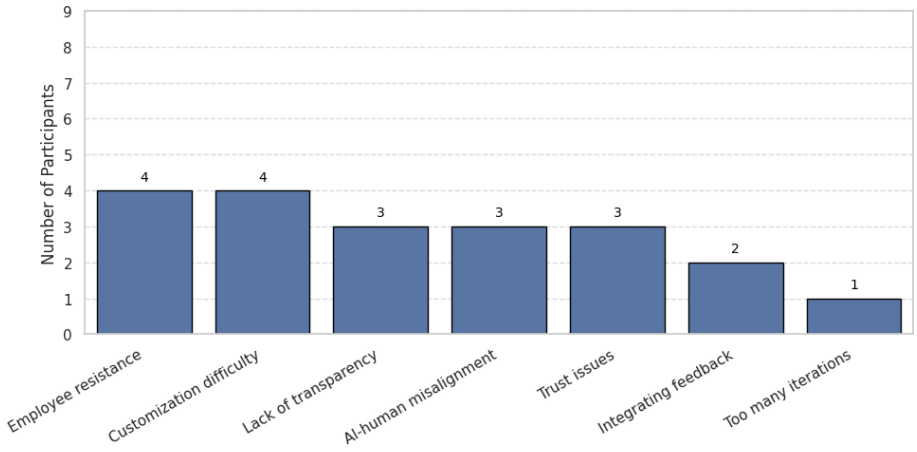


Figure 7. Most commonly reported challenges identified in the questionnaire

Interview findings supported these results and explained why they occurred. Three themes were identified: *usability and interface barriers*, *scalability and adaptability constraints*, and *trust and transparency limitations*. Participants also proposed practical improvements that align with these themes.

Usability and interface barriers. The interview data provide further detail on usability challenges. Participants explained that navigation and layout issues reduced efficiency during requirement review and prioritization. P2 (Company A) explained that the interface caused friction, noting, *"When I tried to assign weightings in the prioritization view, it did not work properly."* P3 (Company B) similarly stated that the system *"needs more development to reach a usable quality level."* These statements clarify that resistance and customization concerns reflected workflow interruptions rather than reluctance toward AI assistance. Participants suggested improving layout consistency, clearer task flow, and visual indicators for new results to enhance speed and reduce friction.

Scalability and adaptability constraints. Participants also discussed performance limitations when handling larger datasets. Although scalability appeared less frequently in questionnaires, it was discussed in every interview. P3 (Company B) noted, *"When I tested it for larger concepts, it lost focus. I guess it's a general issue with LLM."* P9 (Company D) added, *"If you're generating 100+ requirements, grouping or deduplication becomes hard. There needs to be more hierarchy."* These views indicate that the system performed effectively for smaller datasets but lost structure as data volume increased. Participants recommended hierarchy views, filters, and duplicate-detection tools to preserve clarity in large-scale projects.

Trust and transparency limitations. Interview participants raised concerns about transparency of reasoning behind prioritization outcomes. P3 (Company B) pointed out that *"It doesn't show how WSJF scores are calculated. That undermines the trust."* P2 (Company A) similarly stated, *"The team was concerned about how the agent was making decisions, especially in prioritization."* These insights help explain the medium frequency of trust-related issues reported in the questionnaire. Participants stressed the need for visible

reasoning to sustain confidence in outputs and suggested adding intermediate calculations, manual adjustment options, and rationale logs to improve traceability and trust.

These interview findings reinforce the quantitative results shown in Figure 7. Participants valued the system's capability to analyze and prioritize requirements but emphasized that adoption required smoother interface design, better scalability, and greater transparency. Addressing these needs through improved layout, hierarchical organization, and clear explanation of reasoning processes was considered essential for broader organizational use.

6. Discussion

This section discusses the key findings by interpreting results in context, identifying implications for practice and research, and summarizing lessons learned. It also reflects on challenges and limitations observed during system use in four companies.

6.1. Interpretation of findings

The results indicate that the proposed multi-agent system that across the four companies, similar patterns were observed in how the system supported requirement generation, refinement, and prioritization. In all cases, participants confirmed that human oversight was necessary to ensure contextual and organizational accuracy. First, participants aligned AI-generated requirements with domain terminology and internal objectives. Second, requirements were adjusted for compliance and feasibility constraints that were not captured by prompts. Third, cross-department alignment required human judgment to maintain consistency with organizational priorities. These refinements indicate that human input focused on contextual interpretation rather than only on error correction. The feedback mechanism was consistently valued for enabling targeted revisions without extensive rework, typically within two feedback rounds. These commonalities suggest that human–AI collaboration was effective in maintaining alignment between generated outputs and organizational goals, regardless of company type or domain.

Differences across cases were linked mainly to organizational context, sector characteristics, and participant roles. The software consultancies (Companies A and B) emphasized efficiency and task-level integration into ongoing projects, reflecting their agile development environments. In contrast, the energy and logistics organizations (Companies C and D) focused on compliance, governance, and strategic alignment due to their regulated and large-scale operational contexts. Participant backgrounds also influenced these differences: technical leaders prioritized feasibility and iteration speed, whereas strategic and managerial roles emphasized explainability, traceability, and long-term value. These contextual factors shaped how participants perceived the system's strengths and areas for improvement.

Overall, it indicates that organizational maturity in AI adoption and participant expertise guided the nature of human feedback. While all companies recognized the system's value in structuring early requirement analysis, their emphasis differed according to internal objectives and operational settings. This variation demonstrates that human–AI collaboration in RE must adapt to sector-specific needs, governance structures, and user expectations to ensure reliable and contextually appropriate results.

6.2. Implications and lessons for practitioners

This study identifies practical aspects for applying LLM-based multi-agent systems in organizational AI planning and requirements analysis. A structured human–AI workflow supports requirement generation, refinement, and prioritization. Participants can approve, modify, delete, or request clarification for individual requirements. This controlled interaction maintains human oversight throughout, ensuring contextual alignment, compliance, and quality assurance while reducing effort for initial drafting.

Role-based configurable agents distribute work between specialized roles such as generation, verification, and prioritization. This configuration supports accountability and allows adaptation to different organizational contexts and stakeholder responsibilities.

Explainability and auditability should be embedded in system design to maintain trust and enable traceability of decisions [54]. Integration with existing planning and documentation environments can reduce adoption effort and support iterative improvement [18, 55]. Scalability and governance mechanisms are needed to preserve clarity and accountability in large projects. Human oversight remains central for contextual judgment, compliance, and verification in agentic workflows [56]. Organizations should define clear human roles that balance automation and accountability, ensuring that agent-generated outputs are reviewed and adjusted before implementation.

For practitioners, the key lessons are: (i) multi-agent systems should function under defined human oversight, (ii) iterative feedback cycles should be part of the review process to improve output quality, (iii) transparency mechanisms should make intermediate reasoning visible for review, (iv) workflow compatibility and interface clarity should support organizational adoption, and (v) governance structures should ensure responsible and traceable decision-making.

These implications guide the development and use of transparent, auditable, and context-aware multi-agent systems for requirement analysis in industrial environments.

6.3. Implications for research

This study offers preliminary empirical insights into the operation of LLM-based multi-agent systems within real organizational contexts, extending earlier conceptual and experimental work [15, 57]. While prior studies focused mainly on technical feasibility, our findings illustrate how human-in-the-loop interaction and contextual adaptation may influence system usefulness and user acceptance. These results provide early empirical support for further investigation into AI-supported RE and decision-making in organizational settings.

The findings also complement existing theories of human–AI collaboration [58, 59], suggesting that user feedback can function both as a quality control mechanism and a means of trust calibration. The observed role of transparency is consistent with work on explainable AI [54], indicating that explainability-by-design remains an important research direction for RE tools. Future research should examine systematic approaches for tracing agent reasoning and assessing how visibility of decision logic affects user trust and adoption.

Finally, the study highlights the value of integrating socio-technical perspectives into the design of agentic systems. Role-based configuration and transparent feedback loops may provide a useful foundation for studying distributed reasoning and accountability in human–agent teams. These observations align with emerging industry perspectives that emphasize deliberate collaboration between humans and agents [56]. Further work

should expand on these exploratory findings to develop structured evaluation methods and theoretical models connecting AI engineering and organizational research.

6.4. Limitations and future work

This study has several limitations that constrain interpretation. Current large language models remain limited by hallucination, weak domain adaptation, and incomplete reasoning transparency. Although human-in-the-loop validation reduced inconsistencies, generation errors and redundancy persisted, showing that the system cannot yet operate autonomously. Agent coordination was constrained by the absence of shared memory and adaptive communication, limiting collaborative reasoning and continuity across refinement cycles. The system also operated under short context windows, affecting traceability during iterative interaction. We use GPT-4o model for evaluation, restricting comparison across architectures and excluding analysis of model bias and fairness. The participant sample was small (nine questionnaires and four interviews across four organizations), limiting generalizability beyond similar contexts.

Future studies expand the participant base to improve representativeness and assess generalizability across industries. Integrating retrieval-augmented memory or extended-context models could improve reasoning continuity and traceability during multi-round refinement. Embedding explainability features such as reasoning logs, decision-tracing, or rationale visualization can strengthen interpretability and trust [54]. Comparative evaluation using multiple open and proprietary models is needed to assess robustness and reproducibility. Longitudinal and domain-specific studies can examine system adaptation over time and under regulatory constraints. Finally, developing adaptive coordination mechanisms [57] may enhance information sharing and collaborative reasoning among agents, supporting scalability in organizational deployment.

7. Threats to validity

There are a number of threats to the validity of our findings, despite our efforts to mitigate them. In reporting these, we adopt the classification scheme proposed by Runeson and Höst [49], which builds on Yin's work [60] and is widely used in software engineering case study research. The scheme includes four dimensions: construct validity, internal validity, external validity, and reliability. While originally developed for positivist studies, Runeson and Höst note that it can also be applied to flexible design studies. We therefore use it here to structure our validity discussion

7.1. Construct validity

The evaluation focused on a role-based multi-agent system using GPT-4o to support AI planning tasks. The questionnaire was aligned with the research questions and reviewed by multiple authors for coverage and clarity. For requirements-related quality aspects, constructs such as relevance, completeness, and refinement were informed by ISO/IEC/IEEE 29148:2011 guidelines. Triangulation with qualitative interview data supported the interpretation of participant responses. Variation in participant roles and company contexts helped ensure consistent application of the constructs. Future studies will include more

companies, a larger and more diverse participant pool, and additional stakeholder roles to evaluate construct stability across different organizational settings.

7.2. Internal validity

The study conducted to evaluate LLM-based multi-agent system in companies for strategic AI planning. Participant responses may have been influenced by expectations, organizational roles, or group dynamics during planning activities. Company-specific factors such as team composition, AI maturity, and strategic priorities may also have shaped perceptions of the system. We used a standardized task structure and fixed agent prompts across all case studies to reduce variation. Data triangulation across questionnaires, and interviews reduce interpretation bias. Similar patterns across companies suggest findings are not context-specific. No detailed interaction logs captured; future work wil incorporate these to support behavioral analysis. Furthermore, researcher bias represents an additional internal validity threat. The authors participated in both system design and evaluation, which may influence the interpretation of participant feedback. Anonymization and analysis help reduce such effects, although complete neutrality cannot be ensured.

7.3. External validity

The study included four case companies from different industrial sectors, and the consistency of their feedback strengthens the external validity and overall rigor. Each company contributed post-session questionnaire responses and one follow-up interview. Although participants held relevant roles in AI planning, the small sample size and early-stage deployment context limit generalizability. The limited number of participants (nine questionnaires and four interviews) further constrains external validity. Future studies with broader representation, including small and medium-sized enterprises, would enhance generalizability and allow deeper assessment of transferability across industrial contexts.

7.4. Reliability

System configuration, agent roles, task prompts, and instruments were consistent across all cases. The first author conducted and thematically analyzed all interviews, while questionnaire data were examined using descriptive statistics. All protocols were designed with the second and other authors, though minor variation in facilitation may have occurred. A shared coding scheme was applied to reduce subjectivity. Future studies could strengthen reliability by using automated instrumentation and expanding data collection and analysis.

7.5. Ethical concerns

All respondents and case companies were anonymized to protect confidentiality and encourage honest feedback. Only essential contextual details were retained. This reduced the risk of biased or overly positive responses. One participating company also signed a Non-Disclosure Agreement (NDA) to ensure proper handling of sensitive data.

8. Conclusions

This study examined a LLM-based multi-agent system for requirements analysis in the context of strategic AI adoption. The system assigns stakeholder-inspired roles that generate, refine, and prioritize requirements in collaboration with human users. Evidence from a multi-case study involving four companies suggests that the system can support early requirement structuring and facilitate planning discussions. Participants valued the ability to review agent-generated drafts but emphasized that human oversight remained necessary to ensure contextual relevance, compliance alignment, and strategic fit. Reported challenges included limited transparency of prioritization logic, difficulty in adapting outputs to organizational standards, and the lack of integration with existing enterprise tools.

Overall, the results indicate that such a system could assist early planning activities by reducing manual effort and providing structured starting points for discussion, but it does not replace human judgment or domain expertise. Future research investigate transparency mechanisms such as decision logs and traceable reasoning, explore collaborative multi-user operation, and examine integration with enterprise platforms. Broader evaluations across diverse organizations may provide stronger evidence on the system's practical adoption, scalability, and organizational readiness.

CRediT authorship contribution statement

M. A. Sami: system implementation, methodology, data collection and analysis, writing the original draft. Z. Zhang: conceptualization, methodology, review and editing, supervision. M. Waseem: conceptualization, methodology, review and editing, supervision. K-K. Kemmel: conceptualization, methodology, review and editing, supervision. Z. Rasheed: review and editing. T. Herda: conceptualization, review and editing. P. Abrahamsson: conceptualization, methodology, review and editing, supervision.

Declaration of competing interest

One of the authors is affiliated with an industrial partner organization. However, the research was conducted independently, and the company was not involved in the design, data analysis, or interpretation of the findings. The authors declare no competing financial interests or personal relationships that could have influenced the work reported in this paper.

Data and code availability

The source code, experimental scripts, and data used in this study (excluding confidential data) are publicly available on GitHub at https://github.com/GPT-Laboratory/multiagent-story-generation.

Funding

This work has been supported by the Research Council of Finland under the project "SYNTHETIC/SW", decision number 312135871211. The research was conducted at Tampere University, GPT-Lab.

Declaration of AI assistance in the writing process

ChatGPT and Grammarly were used for language editing and summarization to improve clarity and grammar. No generative AI was used for figures. All AI-assisted content was reviewed and finalized by the authors.

References

[1] A. Dasgupta and S. Wendler, "AI adoption strategies," Working Paper Series No. 9, Centre for Technology and Global Affairs, University of Oxford, March 2019. [Online]. https://www.politics.ox.ac.uk/sites/default/files/2022-03/201903-CTGA-Dasgupta%20A-Wendler%20S-aiadoptionstrategies.pdf

[2] S. Kaggwa, T.F. Eleogu, F. Okonkwo, O.A. Farayola, P.U. Uwaoma et al., "AI in decision making: Transforming business strategies," *International Journal of Research and Scientific Innovation*, Vol. 10, No. 12, 2024, pp. 423–444.

[3] Gartner, "Gartner survey finds generative AI is now the most frequently deployed AI solution in organizations," https://www.gartner.com/en/newsroom/press-releases/2024-05-07-gartner-survey-finds-generative-ai-is-now-the-most-frequently-deployed-ai-solution-in-organizations, 2024, accessed: 2025-04-06.

[4] Gartner, "Gartner survey finds 79% of corporate strategists see AI and analytics as critical to their success over the next two years," 2023. [Online]. https://www.gartner.com/en/newsroom/press-releases/2023-07-05-gartner-survey-finds-79-percent-of-corporate-strategists-see-ai-and-analytics-as-critical-to-their-success-over-the-next-two-years

[5] K.K. Kemell, M. Saarikallio, A. Nguyen-Duc, and P. Abrahamsson, "Still just personal assistants? A multiple case study of generative AI adoption in software organizations," *Information and Software Technology*, Vol. 186, 2025, p. 107805.

[6] D. Russo, "Navigating the complexity of generative AI adoption in software engineering," *ACM Transactions on Software Engineering and Methodology*, Vol. 33, No. 5, Jun. 2024. [Online]. https://doi.org/10.1145/3652154

[7] D. Herremans, "aiSTROM – A roadmap for developing a successful AI strategy," *IEEE Access*, Vol. 9, 2021, pp. 155 826–155 838.

[8] K. Ahmad, M. Abdelrazek, C. Arora, M. Bano, and J. Grundy, "Requirements practices and gaps when engineering human-centered artificial intelligence systems," *Applied Soft Computing*, Vol. 143, 2023, p. 110421.

[9] A. Choudhury and H. Shamszare, "Investigating the impact of user trust on the adoption and use of ChatGPT: Survey analysis," *Journal of Medical Internet Research*, Vol. 25, 2023, p. e47184.

[10] F. Zambonelli, N.R. Jennings, and M. Wooldridge, "Developing multiagent systems: The Gaia methodology," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, Vol. 12, No. 3, 2003, pp. 317–370.

[11] U.e. Habiba, M. Haug, J. Bogner, and S. Wagner, "How mature is requirements engineering for AI-based systems? A systematic mapping study on practices, challenges, and future research directions," *Requirements Engineering*, 2024, pp. 1–34.

[12] X. Franch, A. Jedlitschka, and S. Martínez-Fernández, "A requirements engineering perspective to AI-based systems development: A vision paper," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 2023, pp. 223–232.
[13] A. Mehraj, Z. Zhang, and K. Systä, "A tertiary study on AI for requirements engineering," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 2024, pp. 159–177.
[14] Z. Zhang, M. Rayhan, T. Herda, M. Goisauf, and P. Abrahamsson, "LLM-based agents for automating the enhancement of user story quality: An early report," in *International Conference on Agile Software Development*. Springer Nature Switzerland Cham, 2024, pp. 117–126.
[15] D. Jin, Z. Jin, X. Chen, and C. Wang, "MARE: Multi-agents collaboration framework for requirements engineering," *arXiv preprint arXiv:2405.03256*, 2024.
[16] P. Bornet, J. Wirtz, T.H. Davenport, D.D. Cremer, B. Evergreen et al., *Agentic Artificial Intelligence: Harnessing AI Agents to Reinvent Business, Work, and Life*. Singapore: World Scientific Publishing, 2025. [Online]. https://www.amazon.com/dp/B0CQYZN7QM
[17] M.A. Sami, Z. Zhang, M. Waseem, K.K. Kemell, Z. Rasheed et al., "A multi-agent LLM system for automated requirements analysis: A study on user story generation and prioritization," in *Euromicro Conference on Software Engineering and Advanced Applications*. Springer, 2025, pp. 178–187.
[18] M.C. Lee, H. Scheepers, A.K. Lui, and E.W. Ngai, "The implementation of artificial intelligence in organizations: A systematic literature review," *Information and Management*, Vol. 60, No. 5, 2023, p. 103816.
[19] P. Hamm and M. Klesel, "Success factors for the adoption of artificial intelligence in organizations: A literature review," in *27th Americas Conference on Information Systems, AMCIS: Digital Innovation and Entrepreneurship*. Association for Information Systems, 2021.
[20] D. Smit, S. Eybers, and A. van der Merwe, "Towards human-AI symbiosis: Designing an artificial intelligence adoption framework," *South African Computer Journal*, Vol. 36, No. 1, 2024, pp. 76–104.
[21] A. Tursunbayeva and H.C.B. Gal, "Adoption of artificial intelligence: A TOP framework-based checklist for digital leaders," *Business Horizons*, Vol. 67, No. 4, 2024, pp. 357–368.
[22] J. Yang, Y. Blount, and A. Amrollahi, "Artificial intelligence adoption in a professional service industry: A multiple case study," *Technological Forecasting and Social Change*, Vol. 201, 2024, p. 123251.
[23] F. Selten and B. Klievink, "Organizing public sector AI adoption: Navigating between separation and integration," *Government Information Quarterly*, Vol. 41, No. 1, 2024, p. 101885.
[24] G. Vaz Pereira, V. Jackson, R. Prikladnicki, A. van der Hoek, L. Fortes et al., "Exploring GenAI in software development: Insights from a case study in a large Brazilian company," in *IEEE International Conference on Software Engineering (ICSE)*, 2025.
[25] J. Flyckt, T. Gorschek, D. Mendez, and N. Lavesson, "Identifying key AI challenges in make-to-order manufacturing organisations: A multiple case study," *Journal of Systems and Software*, Vol. 230, 2025, p. 112559.
[26] Y. Feng, S. Vanam, M. Cherukupally, W. Zheng, M. Qiu et al., "Investigating code generation performance of ChatGPT with crowdsourcing social data," in *Proceedings of the 47th IEEE Computer Software and Applications Conference*, 2023, pp. 1–10.
[27] F.A. Shah, A. Sabir, R. Sharma, and D. Pfahl, "How effectively do LLMs extract feature-sentiment pairs from app reviews?" in *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 2025, pp. 123–138.
[28] A. Ferrari and P. Spoletini, "Formal requirements engineering and large language models: A two-way roadmap," *Information and Software Technology*, Vol. 181, 2025, p. 107697. [Online]. https://www.sciencedirect.com/science/article/pii/S0950584925000369
[29] O. Abed, K. Nebe, and A.B. Abdellatif, "AI-generated user stories supporting human-centred development: An investigation on quality," in *International Conference on Human-Computer Interaction*. Springer, 2024, pp. 3–13.
[30] L. Zhong, Z. Wang, and J. Shang, "LDB: A large language model debugger via verifying runtime execution step-by-step," *arXiv preprint arXiv:2402.16906*, 2024.

[31] B. Wei, "Requirements are all you need: From requirements to code with LLMs," in *IEEE 32nd International Requirements Engineering Conference (RE)*. IEEE, 2024, pp. 416–422.
[32] S. Manish, "An autonomous multi-agent LLM framework for agile software development," *International Journal of Trend in Scientific Research and Development*, Vol. 8, No. 5, 2024, pp. 892–898.
[33] C. Hymel and H. Johnson, "Analysis of LLMs vs human experts in requirements engineering," *arXiv preprint arXiv:2501.19297*, 2025.
[34] C. Zhang, K. Yang, S. Hu, Z. Wang, G. Li et al., "ProAgent: Building proactive cooperative agents with large language models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38, No. 16, 2024, pp. 17 591–17 599.
[35] X. Zou, Y. Liu, X. Shi, and C. Yang, "Goal2Story: A multi-agent fleet based on privately enabled sLLMs for impacting mapping on requirements elicitation," *arXiv preprint arXiv:2503.13279*, 2025.
[36] W. Tao, Y. Zhou, Y. Wang, W. Zhang, H. Zhang et al., "MAGIS: LLM-based multi-agent framework for github issue resolution," *Advances in Neural Information Processing Systems*, Vol. 37, 2024, pp. 51 963–51 993.
[37] J. Liu, G. Wang, R. Yang, J. Zeng, M. Zhao et al., "RTADev: Intention aligned multi-agent framework for software development," in *Findings of the Association for Computational Linguistics: ACL 2025*, 2025, pp. 1548–1581.
[38] H. Takagi, S. Moriya, T. Sato, M. Nagao, and K. Higuchi, "A framework for efficient development and debugging of role-playing agents with large language models," in *Proceedings of the 30th International Conference on Intelligent User Interfaces*, 2025, pp. 70–88.
[39] K.T. Tran, D. Dao, M.D. Nguyen, Q.V. Pham, B. O'Sullivan et al., "Multi-agent collaboration mechanisms: A survey of LLMs," *arXiv preprint arXiv:2501.06322*, 2025.
[40] A. Singh, A. Ehtesham, S. Kumar, and T.T. Khoei, "Enhancing AI systems with agentic workflows patterns in large language model," in *World AI IoT Congress (AIIoT)*. IEEE, 2024, pp. 527–532.
[41] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li et al., "AutoGen: Enabling next-gen LLM applications via multi-agent conversation," *arXiv preprint arXiv:2308.08155*, 2023.
[42] N.A. Parikh, "Managing AI-first products: Roles, skills, challenges, and strategies of AI product managers," *IEEE Engineering Management Review*, 2025, pp. 1–11.
[43] N. Tasneem, H.B. Zulzalil, and S. Hassan, "Enhancing agile software development: A systematic literature review of requirement prioritization and reprioritization techniques," *IEEE Access*, 2025.
[44] F. Essam, H. El, and S.R.H. Ali, "A comparison of the Pearson, Spearman rank and Kendall tau correlation coefficients using quantitative variables," *Asian Journal of Probability and Statistics*, 2022, pp. 36–48.
[45] M.A. Sami, M. Waseem, Z. Zhang, Z. Rasheed, K. Systä et al., "Early results of an AI multiagent system for requirements elicitation and analysis," in *Product-Focused Software Process Improvement*, D. Pfahl, J. Gonzalez Huerta, J. Klünder, and H. Anwar, Eds. Cham: Springer Nature Switzerland, 2025, pp. 307–316.
[46] P. Murley, Z. Ma, J. Mason, M. Bailey, and A. Kharraz, "Websocket adoption and the landscape of the real-time web," in *Proceedings of the Web Conference 2021*, 2021, pp. 1192–1203.
[47] B. Lubanovic, *FastAPI*. O'Reilly Media, Inc., 2023.
[48] A. Hurst, A. Lerer, A.P. Goucher, A. Perelman, A. Ramesh et al., "GPT-4o system card," *arXiv preprint arXiv:2410.21276*, 2024.
[49] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, Vol. 14, 2009, pp. 131–164.
[50] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative Research in Psychology*, Vol. 3, No. 2, 2006, pp. 77–101.
[51] R.K. Yin, *Case Study Research and Applications: Design and Methods*, 6th ed. SAGE Publications, 2018.
[52] I. Etikan, S.A. Musa, R.S. Alkassim et al., "Comparison of convenience sampling and purposive sampling," *American Journal of Theoretical and Applied Statistics*, Vol. 5, No. 1, 2016, pp. 1–4.

[53] I. Garcia, C. Pacheco, A. León, and J.A. Calvo-Manzano, "A serious game for teaching the fundamentals of ISO/IEC/IEEE 29148 systems and software engineering –lifecycle processes – requirements engineering at undergraduate level," *Computer Standards and Interfaces*, Vol. 67, 2020, p. 103377.
[54] H. Zhao, H. Chen, F. Yang, N. Liu, H. Deng et al., "Explainability for large language models: A survey," *ACM Transactions on Intelligent Systems and Technology*, Vol. 15, No. 2, 2024, pp. 1–38.
[55] D. Russo, "Navigating the complexity of generative AI adoption in software engineering," *ACM Transactions on Software Engineering and Methodology*, Vol. 33, No. 5, 2024, pp. 1–50.
[56] McKinsey and Company, "One year of agentic AI: Six lessons from the people doing the work," https://www.mckinsey.com/capabilities/quantumblack/our-insights/one-year-of-agentic-ai-six-lessons-from-the-people-doing-the-work, 2025, accessed: 2025-10-13.
[57] J. He, C. Treude, and D. Lo, "LLM-based multi-agent systems for software engineering: Literature review, vision and the road ahead," *ACM Transactions on Software Engineering and Methodology*, 2024.
[58] S. Amershi, A. Begel, C. Bird, R. DeLine, G. Gall et al., "Software engineering for machine learning: A case study," *International Conference on Software Engineering (ICSE)*, 2019, pp. 291–300. [Online]. https://doi.org/10.1109/ICSE.2019.00042
[59] C.O. Retzlaff, S. Das, C. Wayllace, P. Mousavi, M. Afshari et al., "Human-in-the-loop reinforcement learning: A survey and position on requirements, challenges, and opportunities," *Journal of Artificial Intelligence Research*, Vol. 79, 2024, pp. 359–415.
[60] R.K. Yin, *Case Study Research: Design and Methods*, 5th ed. Thousand Oaks, CA: Sage Publications, 2014.

Authors and affiliations

Malik Abdul Sami
e-mail: malik.sami@tuni.fi
ORCID: https://orcid.org/0000-0001-5136-2587
Faculty of Information Technology and Communication Sciences, Tampere University, Finland

Zheying Zhang
e-mail: zheying.zhang@tuni.fi
ORCID: https://orcid.org/0000-0002-6205-4210
Faculty of Information Technology and Communication Sciences, Tampere University, Finland

Muhammad Waseem
e-mail: muhammad.waseem@tuni.fi
ORCID: https://orcid.org/0000-0001-7488-2577
Faculty of Information Technology and Communication Sciences, Tampere University, Finland

Kai-Kristian Kemell
e-mail: kai-kristian.kemell@tuni.fi
ORCID: https://orcid.org/0000-0002-0225-4560
Faculty of Information Technology and Communication Sciences, Tampere University, Finland

Zeeshan Rasheed
e-mail: zeeshan.rasheed@tuni.fi
ORCID: https://orcid.org/0000-0001-9655-3096
Faculty of Information Technology and Communication Sciences, Tampere University, Finland

Tomas Herda
e-mail: herda.tom@gmail.com
ORCID: https://orcid.org/0009-0005-2912-380X
AI Center of Excellence, Austrian Post, Vienna, Austria

Pekka Abrahamsson
e-mail: pekka.abrahamsson@tuni.fi
ORCID: https://orcid.org/0000-0002-4360-2226
Faculty of Information Technology and Communication Sciences, Tampere University, Finland

e-Informatica Software Engineering Journal
Volume 20, Issue 1, pages: 1–22
DOI: 10.37190/e-Inf260104
BibTeX

Realising the Role of Generative AI Among Project Management Professionals

Kari Sainio*, Petri Kettunen, Pekka Abrahamsson
*Corresponding author: kari.sainio@tuni.fi

Article info

Keywords:
generative AI
project management
project tasks
GenAI adoption
project tools

Submitted: 8 Dec. 2025
Revised: 22 Feb. 2026
Accepted: 20 Apr. 2026
Available online: 23 Apr. 2026

Abstract

Context: While conceptual research on AI in project management is advancing, empirical evidence of actual usage among IT project managers practising is limited.
Objective: We investigate how Finnish IT project and programme managers use generative AI in daily work, identifying practices, organisational constraints, and future visions through individual interviews.
Method: Using qualitative descriptive design, we conducted semi-structured interviews with 12 experienced Finnish IT project management consultants who work in multiple client organisations. Data were analysed through hybrid inductive-deductive thematic analysis to identify usage practices and future usage.
Results: GenAI adoption is fragmented and peripheral, mainly constrained by organisational policies rather than individual resistance. Participants use GenAI mainly for support tasks rather than core project management functions. Despite varied usage, participants converge on envisioning AI as an "assistant not replacement," reflecting professional boundary work that preserves human authority.
Conclusions: The adoption of GenAI in IT project management is limited by organisational constraints such as security policies and governance structures, which means that organisations should prioritise integration and data protection over bottom-up experimentation. More advanced capabilities remain aspirational, making incremental adoption through low-risk use cases more realistic than broad automation of core project management functions.

1. Introduction

The integration of artificial intelligence (AI) into information technology (IT) project management represents a technological shift with implications for professional practice, organisational efficiency, and the future of work. Recent years have witnessed growing interest in how generative AI (GenAI) tools can support or transform project management activities. Difference research papers have studied how generative AI (GenAI) can help and change the work of a modern IT project manager [1–5]. GenAI is evolving and enables new ways of working by providing assistance to project managers. However, today's IT project managers may find it difficult to benefit of the usage of GenAI or are at their early stage of adopting GenAI.

© 2026 The Authors. Published by Wrocław University of Science and Technology Publishing House.
This is an open access article under the CC BY-SA 4.0 license international.

The purpose of this research is to examine the current practices in which GenAI tools are used and to explore the future implications of how AI can help IT project managers working in client organisations on a temporary basis. The objective of this research is to concentrate on competent Finnish IT project and programme management consultants how they use GenAI in their daily project management activities within client organisation engagements in an individual perspective. Furthermore, to find out what their vision is for the future of co-existence of AI and IT project management.

In this paper, IT projects are defined as those that involve software development, enterprise resource planning (ERP) implementations, IT infrastructure changes, system integrations, and digitalisation initiatives. The participating IT project managers represent a range of industries, including telecommunications, banking, healthcare, insurance, gaming, and cybersecurity. They also work in the client organisation as hired external project managers. Taking into account the following research questions (RQ1, RQ2), this study aims to deepen the understanding of how project management consultants use GenAI in their work and what thoughts and opportunities AI will use in the future.

RQ1: How are IT project management professionals using GenAI tools today?

RQ2: What vision do IT project management professionals have for the future use of AI?

This study follows the qualitative descriptive design guidelines [6,7] using semi-structured interviews with 12 experienced Finnish IT project management consultants. The research focused on practical interview questions with the aim of systematically documenting practises as they are experienced by the interviewed participants. Given that GenAI tools became widely available in professional contexts in 2023 [8], this research can be seen as an early-stage adoption characterised by heterogeneous and individualised usage practices. The rest of this paper is organised as follows. In Section 2, we present a background. In Section 3, the research method is described. Section 4 presents the results. Section 5 contains a discussion of the results, and final Section 6 concludes the study with the conclusions.

2. Background

Project management can be challenging. Several publications have studied different challenges in the success of the projects [9–11]. Moreover, the skill set of a project manager is broad, including capabilities such as communication, stakeholder engagement, and team motivation [12,13]. The day-to-day work of IT project managers follows the stages of project management and has many different tasks to consider throughout the project lifecycle [12]:

- planning scope, objectives, deliverables and project schedules,
- monitoring progress, controlling risks, and making necessary adjustments,
- managing the team through task allocation, conflict resolution, and alignment of capabilities,
- communicating with stakeholders by reporting status, managing expectations, and maintaining feedback channels,
- and ensuring quality through metrics, reviews, and compliance activities [12].

The rise of GenAI has created an important research domain focused on the evolving role of project managers. Different studies show interest in how GenAI can be used to support project management [14–16]. In the related literature, the distinction between

GenAI and AI is vague, making interpretations of related work difficult, since AI by definition covers a wide range of applications [17]. Furthermore, research related to the combination of AI and project management is challenging. According to Muller et al., in different studies, quality over quantity should be emphasised, bridge theory and practice, and provide future research guidance [18].

Various GenAI tools are frequently introduced to help in project management. As an example, Project Management Institute (PMI) has published the GenAI-powered assistant tool called PMI Infinity for project professionals [19]. It is targeted to provide intelligent guidance to current and impact PMI's best practices that enhance project outcomes. It contains a repository of more than 14,000 content items, curated by PMI members and global project management experts. Like OpenAI's ChatGPT, it provides conversations through the chat interface [20]. PMI has also conducted the second global study that includes 500 project professionals, each representing their organisation, collecting data from 18 industry sectors in 12 countries that state the situation in 2024 of how AI is adopted in project management related work [21].

In addition to PMI's Infinity [19], Dam et al. have introduced a framework through which GenAI technologies can be used to facilitate agile project management [22]. In general, different GenAI generated content mechanisms have been studied. Zuge et al. have studied that GenAI-created content has the potential to revolutionise engineering management by improving decision making, optimising resource allocation, and streamlined workflows across various phases of the life cycle, including demand analysis, design, implementation, testing, and evaluation [23]. Continuing, Barcaui et al. argues that when comparing AI-generated and human-created project plans, each exhibited distinct strengths and weaknesses, suggesting that a collaborative approach could be the most effective [24]. According to Patel, by automating routine tasks, optimising resource allocation, supporting critical decisions, and enabling precise forecasting, GenAI has the potential to fundamentally reshape project management practices [25]. Continuing, Bushuyev et al. argue that AI shapes project management by improving decision-making, optimising resource use, and automating routine work through predictive analytics, real-time risk evaluation, and data-driven insights that support more accurate managerial decisions [26].

Today, GenAI solutions may lack the capabilities needed to effectively help in complex project management challenges. Although GenAI tools can help project managers control and monitor projects, the weaknesses identified in existing models highlight the continued importance of human knowledge and judgement in the evaluation of results [27]. Moreover, general-purpose Large Language Models (LLMs), although not explicitly trained in project management, already possess the ability to address diverse challenges in the field and demonstrate knowledge that approaches professional certification standards [28]. According to Odeh, the growing pace of AI adoption and its role in driving innovation will require the project management profession to adapt and evolve its methods [29]. Using AI-generated insights and automated workflows, project managers can handle complex situations with greater adaptability and anticipation. Whether selecting and prioritising initiatives, refining processes, or strengthening decision-making, AI elevates the entire project management life cycle, and in particular the administrative workload [30].

GenAI has already been applied in IT project management for some years. Relevant empirical research is still in the early phase and there is much room for future research. However, it is essential to study empirically current IT project management practices and how GenAI can transform work through field interviews to understand the current state and future state of using AI in daily project management work. Despite conceptual

literature, empirical research on actual GenAI usage practices among IT project managers remains limited. Most of the existing work is conceptual, theoretical, or tool-focused rather than practice-focused. By directly interviewing project managers, they can tell more about how they use GenAI.

3. Method

The research is conducted qualitatively based on interviews and applying the qualitative descriptive design [7, 31] to obtain direct low-inference descriptions of the experiences and perspectives of the participants without imposing a theoretical interpretation. Qualitative descriptive research is well suited to explore how people understand and engage with a phenomenon in practice [31, 32], and it prioritises staying close to the data provided by participants [6].

3.1. Participants and recruitment

The interview invitations were distributed electronically on LinkedIn and using the Finnish Project Management Association email list (Projektiammattilaiset ry). The invitation was aimed at people with substantive knowledge of the research topic while ensuring representation in various project management roles that participants (Px) had at the time of the interviews. Twelve IT project management professionals ($n = 12$) accepted the invitation and were recruited (Table 1). This approach can be interpreted as convenience sampling [33].

Table 1. Demographics of the participants in interview order

Participant	Current role
P1	Program manager
P2	Project manager
P3	Project manager
P4	Project management office
P5	Agile coach
P6	Project manager
P7	Project manager
P8	Program manager
P9	Agile coach
P10	Program manager
P11	Program manager
P12	Project manager

3.2. Data collection

Data were collected through semi-structured interviews conducted via Microsoft Teams video conferencing, following a predefined interview guide (Appendix A). The video recordings were transcribed and saved as Microsoft Word files. The interviews covered three main areas relevant to this study: (1) professional background, (2) current use of GenAI-assisted tools in project work, and (3) views on the future role of AI in project management. Additional questions on challenges in project management and the use of traditional project

management tools were asked during interviews, but were excluded from this study because they were not considered relevant to support the answers to the research questions.

3.3. Data analysis

To follow qualitative research [31] data analysis used a mixed inductive and deductive thematic approach [34], combining inductive content analysis [35] with a deductive organisational framework derived from the interview guide and research questions. This approach is consistent with qualitative descriptive methodology, which does not require purely emergent categories, but aims to present findings in a clear and organised manner close to participants' own accounts [6, 7]. The analysis was carried out in three phases.

Phase 1: Transcription and preparation

The interviews were carefully transcribed and saved as Microsoft Word documents. The Finnish-language transcripts were read multiple times to ensure accuracy and translated into English using Microsoft Word's integrated translation function. The English versions were reviewed and corrected through repeated reading and verification as the spoken language is not necessarily precise for machine-based transcription.

Phase 2: Coding

The transcripts were imported into the Atlas.ti qualitative data analysis software. Inductive open coding was performed using Atlas.ti's AI-assisted coding feature [32], generating 424 initial codes. These codes were reviewed against the original transcripts to verify accuracy. The interview guide provided a deductive framework organised around three grouped domains:

1. Professional background and experience (context for understanding adoption).
2. Current GenAI usage practises and applications (to support RQ1).
3. Future expectations for AI in project management (to support RQ2).

Phase 3: Forming themes

Within each domain, inductively generated codes were grouped into themes based on recurring practises in participant responses. For example, descriptions of current GenAI use (RQ1) were organised into themes (and 3) such as "Communication and Documentation," "Learning and Research," "Planning and Strategy," and "Workshop and Meeting Support". These themes were not predetermined but emerged from the data.

Generative AI tools (ChatGPT and Claude) were used to assist in organising groups into tabular summaries aligned with the research questions. All AI-generated outputs were treated as preliminary drafts and systematically verified against the original transcripts, with corrections made by hand where necessary. The interpretation was maintained at a low-inference descriptive level throughout, consistent with qualitative descriptive methodology [6, 7].

4. Results

This section presents the findings of the interview organised into categories emerging from the interview guide (Appendix A), open questions, and topics raised by participants (Pn). The categories cover: participant background and proficiency (4.1), current use of GenAI for project management tasks (4.2), adopted specific GenAI tools (4.3) and expectations regarding future envisions of GenAI (4.4). Each category is addressed in subsections corresponding to Tables 2–5. The answers about Non-AI tools used in the project work were ignored as they did not provide information supporting the research questions (Appendix A, topics 2, 3).

4.1. Background and proficiency of the participants

Participants demonstrate IT project management experience (5+ years) in various projects, domains, and industries. Table 2 presents their work experience in three dimensions: industry sectors, project content areas, and project management methodologies.

Industry experience spans telecommunications, banking, cyber security, healthcare, insurance, energy, postal services, mechanics, games, and education in various unnamed companies. All participants reported on their experience with the software development project. Additional project content areas include cloud technologies, IT system upgrades, ERP implementation, IT systems integration, organisational processes, and testing.

All participants have experience with waterfall methodology, with substantial representation of agile, hybrid approaches, and SAFe (Table 2). This diversity in methodologies,

Table 2. Work experience of the participants

Type\Participant	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
Industry												
Telecommunication	✓	–	✓	–	–	–	–	–	–	–	✓	✓
Gaming	✓	–	–	–	–	–	–	–	–	–	–	✓
Cyber security	✓	–	✓	–	–	–	–	–	–	–	✓	✓
Banking	–	–	–	–	–	–	✓	✓	–	✓	✓	✓
Health Care	–	✓	–	–	–	✓	–	–	–	✓	–	–
Insurance	–	–	–	–	–	✓	–	–	–	✓	✓	–
Education	–	–	–	–	✓	–	–	–	–	–	–	–
Energy	–	–	✓	–	–	✓	–	✓	–	–	–	–
Mechanics	–	–	–	–	–	–	✓	–	✓	–	–	–
Postal	–	–	–	–	–	–	✓	✓	✓	–	–	✓
Project content												
Software Development	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
ERP implementation	–	–	–	–	–	✓	–	–	–	✓	✓	–
Cloud Technologies	✓	–	–	–	–	✓	✓	–	–	✓	✓	✓
Systems Integration	–	–	–	–	–	✓	–	–	–	✓	–	✓
Organisation Processes	–	–	–	✓	–	–	–	✓	–	–	–	–
Systems Upgrade	✓	–	–	–	–	–	✓	–	–	–	✓	✓
Testing	–	–	–	–	✓	–	✓	–	–	–	–	–
Project management method												
Waterfall	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Agile	–	–	✓	✓	✓	–	✓	✓	✓	✓	✓	✓
Hybrid	✓	–	✓	✓	–	✓	✓	✓	✓	✓	✓	✓
SAFe	–	–	✓	–	–	✓	–	✓	–	✓	✓	✓
Mentions in the interview	8	3	8	5	4	10	10	8	6	11	12	11

industries, and project types establishes the participant group as experienced IT project managers with diverse professional backgrounds.

4.2. Project management related tasks where participants use GenAI tools

Table 3 presents activities in which participants have used GenAI tools, categorised under six themes: Communication and Documentation, Learning and Research, Planning and Strategy, Technical and Specialised, Translation and Communication, and Workshop and Meeting Support.

Table 3. Themes where participants indicated they relied on GenAI tools, alphabetical order

Theme\Participant	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
Communication and Documentation												
Creating presentations	✓	–	–	–	–	✓	–	–	–	–	–	✓
Creating templates	–	–	–	–	–	✓	✓	✓	–	–	–	–
Documentation writing	–	–	–	–	–	–	✓	✓	–	–	–	–
Improving text quality	–	–	–	–	✓	–	–	✓	–	–	–	✓
Meeting transcription/notes	–	–	–	–	–	–	✓	–	✓	–	–	–
Message/email drafting	–	–	–	✓	✓	–	–	–	–	–	–	–
Learning and Research												
Information summarisation	–	✓	–	–	✓	–	–	✓	–	–	–	✓
Learning new topics	–	✓	–	–	–	–	✓	–	–	–	–	–
Literature search	–	✓	–	–	–	–	–	–	–	–	–	✓
Technical term explanation	–	–	–	–	–	✓	–	–	–	–	–	–
Planning and Strategy												
Best practices guidance	–	–	✓	–	–	✓	–	–	–	–	–	–
Brainstorming	✓	–	–	✓	–	–	–	–	–	–	–	–
Problem-solving	✓	–	–	–	–	–	✓	–	–	–	✓	–
Project planning	–	–	–	–	–	–	–	✓	–	–	✓	–
Risk identification	–	–	–	–	–	–	–	–	–	–	✓	–
Technical and Specialised												
Building automation flows	–	–	–	✓	–	–	–	–	–	–	–	–
Creating Process instructions	–	–	–	–	–	–	–	✓	–	–	–	–
Cross-checking/QA	–	–	–	–	–	–	–	✓	–	–	–	–
Image generation	–	–	–	–	✓	–	–	–	–	–	–	–
Retrieving company data	–	–	–	–	–	✓	–	–	–	–	–	–
Translation and Communication												
Simplifying technical info	–	–	–	–	–	✓	–	–	–	–	–	–
Translation	–	–	–	–	–	✓	–	–	✓	–	–	–
Workshop and Meeting Support												
Creating action points	–	–	–	–	–	–	–	–	✓	–	–	–
Processing workshop results	–	–	–	–	–	–	–	✓	✓	–	–	–
Workshop facilitation ideas	–	–	–	–	–	✓	–	–	–	–	–	–

Communication and documentation. Three participants (P1, P6, P12) use GenAI to create presentations, particularly for client communications and stakeholder participation. Template creation (P6, P7, P8) includes generating alternatives for analyses, document structures, and Wiki documentation pages. Documentation writing (P7, P8) ranges from document templates to systematic project process instructions, with P8 demonstrating the most extensive usage by creating work cards, checklists, and implementation documentation. Text quality improvement (P5, P8, P12) involves a refinement of tone, correcting syntax, and synthesising older content into well-structured versions while maintaining business-critical details. Meeting transcription and note-taking (P7, P9) use Microsoft tools

to ease administrative burden, although with variable quality depending on the complexity of the meeting. Message and email crafting (P4, P5) serves both mass communications and overcoming writer's block, though outputs require substantial editing.

Learning and research. Information summarisation (P2, P5, P8, P12) accelerates searches through materials, categorises feedback, and synthesises documentation. Learning new topics (P2, P7) helps participants gain a basic understanding before stakeholder participation and efficiently locate relevant resources. The literature search (P2, P12) supports the finding of Finnish and international references. The technical term explanation (P6) enables non-technical project managers to understand technical concepts and assess the severity of the issue for effective team communication.

Planning and strategy. Best practices guide (P3, P6) provides examples and alternatives for professional judgement. Brainstorming and ideation (P1, P4) supports daily concept development and serves as a primary use case. Project planning (P8, P11) assists with work phase structuring and migration project frameworks, though P11's test with Gemini has not been operationalised. Problem-solving (P1, P7, P11) ranges from addressing client challenges to understanding system implementations, with P11 finding experimental results promising but not yet implemented. The identification of risks (P11) demonstrated the potential to identify the risks of typical migration projects in the tests.

Technical and specialised. Building automation workflows (P4) represents the most advanced usage, using ChatGPT API, Flowise, n8n, and Python interfaces to create automated workflows through experimental agent-driven development. Creating process instructions (P8) involves systematically writing value-stream mapping guidance and task checklists since January 20, 2025. Cross-checking and quality assurance (P8) help identify missing elements and overlaps in documentation, although it requires mandatory human verification. Image generation (P5) using Microsoft Copilot yields mixed results that require multiple attempts. Retrieving company data (P6) leverages Microsoft Copilot's internal network access capabilities.

Translation and communication. Simplifying technical information (P6) helps translate complex topics for management audiences. Translation (P6, P9) supports fast multilingual communication needs, with P9 reporting one instance of AI quality exceeding professional translation services.

Workshop and meeting support. Creating action points (P9) from meeting transcripts sometimes produces superior results to manual creation, although it requires verification. Processing workshop results (P8, P9) using AI algorithms in Whiteboard and Miro tools categorises ideas and significantly reduces processing time from 30–60 minutes. Workshop facilitation ideas (P6) provide suggestions for weekly workshops requiring new approaches.

4.3. Specific GenAI tools used by the participants

Table 4 presents the adoption of the specific GenAI tool. ChatGPT ($n = 8$) and Microsoft Copilot ($n = 7$) dominate, with Copilot accessed predominantly through organisational licences. Google Gemini represents both operational and experimental usage. Claude, OpenAI API, and Ollama appear exclusively among technically sophisticated users that require programmatic access or local deployment for data privacy. Specialised automation tools, Flowise, n8n, and Miro AI, are limited to advanced users with specific automation or platform-embedded needs. P4 demonstrates the most sophisticated tool ecosystem, using Flowise and n8n to create visual workflow automation with ChatGPT API integration.

Table 4. GenAI tools adopted by participants in alphabetic order

GenAI Tool \Participant	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
ChatGPT	✓	✓	✓	✓	✓	✓	✓	–	–	–	–	✓
Microsoft Copilot	–	✓	–	–	✓	✓	✓	✓	✓	–	–	✓
Google Gemini	–	–	–	–	✓	–	–	–	–	–	✓	✓
Claude	–	–	–	✓	–	–	–	–	–	–	–	✓
OpenAI API	–	–	–	✓	–	–	–	–	–	–	–	✓
Ollama	–	–	–	✓	–	–	–	–	–	–	–	✓
Flowise	–	–	–	✓	–	–	–	–	–	–	–	–
n8n	–	–	–	✓	–	–	–	–	–	–	–	✓
Miro AI	–	–	–	–	–	–	–	–	✓	–	–	–

The adoption of the specific GenAI tool reveals a clear distinction between main tools for general project management tasks and specialised tools that require technical expertise for advanced automation workflows.

4.4. The future envisions of AI in project management

Participants welcome the future use of AI in project management, emphasising the automation of routine tasks and project assistance as the most beneficial applications. However, they offered few futuristic ideas beyond the current capabilities. Table 5 presents future

Table 5. Future themes envisions by participant (alphabetically ordered)

Future AI theme \Participant	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
Administrative Automation												
Automated reminders	–	–	–	–	–	✓	–	–	–	–	✓	–
Calendar/meeting management aut.	–	–	–	–	–	✓	–	–	✓	✓	✓	–
Reporting automation	–	–	–	–	–	–	✓	–	–	✓	✓	–
Tool setup automation	–	–	–	–	–	–	✓	–	–	–	–	–
AI Agents												
Agent orchestration	–	–	–	–	–	–	–	–	–	–	✓	✓
Autonomous AI agents	✓	✓	–	–	–	–	–	✓	–	✓	✓	✓
Human Primacy and Concerns												
AI as assistant not replacement	–	–	–	–	–	✓	–	✓	–	–	✓	✓
Concerns about skill erosion	–	–	–	–	✓	–	–	–	–	–	–	–
Environmental impact concerns	–	–	–	–	–	–	✓	–	✓	–	–	–
Humans retain decision control	–	–	✓	✓	–	✓	✓	✓	–	–	✓	–
Intelligent Monitoring												
Dependency tracking and mgmt	–	–	✓	–	–	–	✓	–	–	–	✓	–
Deviation and problem alerts	–	–	–	–	–	✓	–	–	–	–	✓	✓
Real-time progress analysis	–	–	✓	✓	–	✓	–	–	–	–	–	–
Risk analysis	–	–	✓	–	–	–	–	–	–	–	✓	–
Quality and Trust												
Higher quality outputs	–	–	–	–	✓	–	–	–	✓	–	–	–
Privacy and data security	–	–	–	✓	–	–	–	–	–	–	–	–
Specialised or domain specific AI	–	–	–	–	✓	–	–	–	–	–	–	–
Trustworthiness (no hallucinations)	–	–	–	✓	–	–	–	–	–	✓	–	–
Strategic Support												
Best practices discovery	–	–	–	–	–	–	–	–	–	✓	–	–
Change impact analysis	–	–	–	–	–	–	–	–	–	–	✓	–
Prediction capabilities	–	–	–	–	–	–	✓	–	–	–	–	✓
Project planning from historical data	–	✓	–	–	–	–	–	–	–	–	✓	✓

AI envisions categorised into six themes: Administrative Automation, AI Agents, Human Primacy and Concerns, Intelligent Monitoring, Quality and Trust, and Strategic Support.

4.4.1. Administrative automation

Automated reminders (P6, P11) would free project managers from repetitive stakeholder follow-ups about agendas, time sheets, and approaching deadlines for work P6 describes as consuming disproportionate time. P11 envisions AI as a project assistant that manages reminders to steering groups and partners when managing multiple concurrent projects.

Calendar and meeting management automation (P6, P9, P10, P11) emerged as the most anticipated capability. Participants seek AI to handle scheduling logistics, coordinate between time zones and organisations, and identify optimal meeting slots for tasks that consume significant professional time. P10 wants an assistant who could manage global team coordination automatically. P11 envisions flight-booking-like functionality suggesting optimal times within flexible windows and automatically adjusting for conflicts.

Report automation (P7, P10, P11) addresses frustration with manual reporting burdens. P10 identifies reporting as the primary pain point, envisions AI automatically retrieving data from various systems, compiling reports without manual entry, and generating summaries to allow focus on problem resolution rather than data compilation. P11 seeks AI not only to automate existing processes, but also to suggest process improvements for complex reporting challenges. The automation of tool configuration (P7) specifically targets the Atlassian Jira configuration, which P7 describes as consistently difficult work involving numerous tasks and configurations, representing a clear opportunity for automation.

4.4.2. AI agents

Agent orchestration (P11, P12) envisions a technical architecture in which duplicable single-task agents could be orchestrated through integration services, with AI dynamically composing processes utilising agent collections based on specific requests to represent sophisticated distributed AI architecture thinking.

Autonomous AI agents (P1, P2, P8, P10, P11, P12) represent the most frequently mentioned advanced functionality. Use cases include requirements management agents monitoring specification updates (P1), knowledge base navigation for workforce planning (P2), workshop facilitation agents preparing exercises based on feedback (P8), backlog management agents providing real-time planning guidance (P8), complex task handling beyond current capabilities (P10), and agents for architectural design, coding, migration planning, and execution (P11). However, P10 frames current AI as a "twelve-year-old colleague who makes mistakes," indicating reliability concerns constrain expanded agent usage.

4.4.3. Human primacy and concerns

AI as assistant, not replacement (P6, P8, P11, P12) emphasises supporting rather than substituting professional judgement. P6 describes the envisioned AI as a project assistant, while P11 consistently frames AI as handling standard reminders and supervision according to lifecycle plans, but maintaining human control.

Concerns about skill erosion (P5) identify over-reliance as threatening employee skill development, particularly for beginners who have not developed experiential understanding.

P5 warns that AI-generated outputs prevent building relationships with the work products necessary for professional competence.

Environmental impact concerns (P7, P9) express anxiety about AI's ecological costs. P7 emphasises that environmental issues are significant yet absent from business AI discussions, questioning how company carbon footprints change with increased AI usage. Both participants view environmental sustainability as a legitimate evaluation criterion that is inadequately addressed in the current discourse.

Humans must retain decision control (P3, P4, P6, P7, P8, P11) represents majority consensus. P3 emphasises that project managers remain essential for group leadership and motivation for work that no machine can perform. P4 frames AI as "bad master, good servant." P6 and P8 particularly emphasise the irreplaceable human in facilitation, coaching, and team motivation, asking whether computers can encourage teams or motivate in ways humans can. P11 notes that even with AI architectural solutions, human verification remains mandatory for error detection.

4.4.4. Intelligent monitoring

Dependency tracking and management (P3, P7, P11) addresses visibility challenges. P7 identifies dependency management as particularly difficult for external project managers. P11 provides the most sophisticated vision: AI that analyses the impacts of changes on dependencies with automatic propagation and resource allocation recommendations.

Deviation and problem alerts (P6, P11, P12) seek active alerting rather than manual monitoring. P11 envisions sophisticated alert capabilities that monitor risk realisation signs and automatically detect when dependencies cannot be completed on schedule, providing actionable coordination recommendations beyond simple notifications.

Real-time progress analysis (P3, P4, P6) seeks automated situational awareness. P3 envisions AI collecting multi-source data, analysing progress against baselines, and providing status analyses with improvement recommendations. P4 envisions automated views that allow the focus to be on problem resolution rather than data compilation. P6 seeks AI management of all data with automated deviation alerting.

Risk analysis (P3, P11) ranges from improvement recommendations (P3) to comprehensive vision based on P11's experimental Gemini test, which successfully identified typical migration project risks. P11 envisions an AI trained on historical project data that provides risk assessments based on project characteristics.

4.4.5. Quality and trust

Higher quality outputs (P5, P9) reflect dissatisfaction with current results requiring extensive correction and editing. P5 seeks specialisation that allows more direct usage of output, while P9 describes situations where AI produces low-quality results that require abandonment in favour of manual work for negating time-saving benefits.

Privacy and data security (P4) motivates local AI installations for sensitive data, with P4 expressing hope for solutions that allow AI to use without data exposure to foreign commercial entities, representing a sophisticated awareness of data sovereignty issues.

Specialised or domain-specific AI (P5) argues that current generic AI does not serve professional work adequately. P5 illustrates with project management examples where generic LLMs produce approximately 70% usable content with 30% irrelevance, concluding

that AI specialists trained in agile methods, project management and organisational behaviour would produce substantially better quality.

Trustworthiness (P4, P10) emerges as a fundamental requirement. P4 prioritises trust above the features, identifying two dimensions: avoiding hallucinations and ensuring data privacy. P10 explicitly states the inability to trust AI for questions beyond existing knowledge, indicating that trust limitations prevent the use of AI for genuine knowledge acquisition versus information reformatting.

4.4.6. Strategic support

Best practice discovery (P10) expresses frustration that current AI cannot access paid professional knowledge databases, though concerns exist about information quality similar to web search results. Change impact analysis (P11) envisions AI analysing budget, schedule, and resource impacts of changes, providing strategic recommendations such as whether to add resources to delayed components or adjust dependent work schedules, with automatic communication to steering groups for representing sophisticated multi-dimensional impact modelling beyond current tool capabilities.

Prediction capabilities (P7) note that existing project management tool reports could be enhanced with AI-driven predictive analytics, though specific targets remain unelaborated. Project planning based on historical data (P2, P11, P12) recognises the value of accumulated organisational learning. P2 envisions AI creating basic project management rulebooks from existing research and practices. P11 provides the most comprehensive vision: systematic database development of project experience, plans, and outcomes that account for technology variations, enabling AI trained on historical data to generate planning templates based on comparable projects through structured parameters for accelerating planning with greater quality than traditional approaches.

5. Discussion

The results show that interviewed Finnish IT project manager consultants use GenAI tools today for a wide range of tasks, many of which are not necessarily directly related to project management, and tools are mainly chat based. In the future more assistive AI tools are visioned that can overtake time consuming and repetitive robot-like work. Moreover, the results reveal fragmented peripheral usage shaped primarily by organisational constraints rather than individual resistance.

5.1. Organisational constraints as primary adoption barrier

The scattered usage (Table 3) shows early adaptation to the use of GenAI tools. Furthermore, the tools are mostly generic GenAI tools and not project management specific (Table 4). The participants explicitly described the constraints for directing how GenAI can be used:

– **Approved tool lists:** Client organisations determine which tools are accessible. Microsoft Copilot dominates not because it is not necessarily superior, but because enterprise licences, sandboxed environment, and Microsoft 365 integration make it organisationally approved.

- **NDA restrictions:** Participants consistently noted that they use GenAI "like a web search"—no confidential information being shared. This fundamentally limits usage to peripheral tasks.
- **Security policies varying across clients:** As external consultants, participants navigate diverse policies ranging from complete GenAI bans to enterprise-wide Microsoft Copilot access. This structural variation indicates fragmentation of the GenAI tools that can be used.

5.2. Uneven GenAI usage reflects an early-stage and individualised adoption

The scattered checkmarks in Table 3 indicate that different GenAI assisted tasks can debatably mean that the participants use GenAI differently from each other. It should also be noted that the participants shared examples of their own GenAI assisted activities that came to mind at the time of the interviews, and not all possible activities were used. It can be assumed that, with increasing GenAI user experience, certain GenAI assisted activities may become more common or new ones may emerge.

The adoption of different GenAI tools by participants is uneven, as seen in Table 4. ChatGPT is reported to be the most widely used tool, suggesting strong familiarity and perceived usefulness among the group, but it may also indicate general public knowledge of the existence of ChatGPT. Microsoft Copilot appears as the second most common option, likely reflecting its integration into client corporate environments where participants already work with Microsoft 365 and client company information security policies affect by limiting usage of other tools. Several participants use a broad mix of tools, indicating a higher degree of experimentation or more advanced technical needs, while others rely on a single tool. Adoption of more specialised setups, such as using GenAI through the OpenAI API, Ollama [36], Flowise [37] and n8n [38], remains limited, indicating that automation-orientated or developer-focused GenAI capabilities have not become the norm within IT project management work.

5.3. AI as augmentation rather than a replacement in IT project management

Participants envision a future in which AI increasingly supports the administrative, analytical, and strategic components of project work. However, these expectations include concerns about trust, contextual understanding, and the preservation of human decision authority. Across participants, administrative automation including reminders, calendar coordination, reporting tasks, and tool configuration is identified as a domain where AI could deliver immediate value. The recurring emphasis on meeting scheduling and stakeholder follow-up indicates that inefficiencies in human coordination remain a central pain point, and participants anticipate that AI could alleviate these kinds of "monkey" work responsibilities.

Beyond administrative support, some participants articulated more advanced expectations involving autonomous AI agents. These visions ranged from relatively simple background assistance to sophisticated architectures involving orchestrated collections of specialised AI agents capable of analysing dependencies, executing migration tasks, or generating complex processes on demand. Such perspectives signal an emerging conceptual shift in which AI is not limited to assisting discrete tasks, but could operate as an integrated component or resource within project ecosystems. However, concerns about data confidentiality, hallucinations, and output quality persist, indicating that current AI capabilities

are not perceived as reliable enough either in the future or in the future. Moreover, doubts about skill erosion reflect concerns that over-reliance on AI may undermine professional expertise, particularly among junior IT project management practitioners.

Participants highlighted that even with substantial AI and automation, project leadership, stakeholder participation, and motivational work are inherently human activities. The notion of AI as a "project assistant," rather than a replacement, reflects a pragmatic orientation. Participants are willing to delegate structured and repetitive tasks, but resist the idea that AI could assume responsibilities that involve empathy, negotiation, or group facilitation as a human project manager.

Some participants also expressed expectations for AI-driven intelligent monitoring, including dependency analysis, deviation detection, and risk assessment. These visions highlight an aspiration for more proactive, predictive, and integrative project oversight that exceeds the reactive reporting capabilities of current tools.

Finally, expectations around strategic support indicate a vision for AI systems capable of synthesising historical project knowledge, offering best-practice recommendations, and modelling the downstream consequences of changes in scope or resources. However, concerns about information quality and source credibility indicate that increased access alone is insufficient.

In summary, these findings depict a future in which AI can play a substantial but bounded role in IT project management. Participants anticipate meaningful automation of repetitive, process-driven tasks and substantial augmentation of analytical and monitoring functions, while simultaneously drawing firm limits around decision authority and human-centric aspects of project work.

5.4. GenAI limitations in IT project management

Among the use of GenAI in various activities (Table 3) the participants identified several limitations of GenAI (Table 6). The inconsistent quality of the output requires human review to verify accuracy and correctness. Furthermore, GenAI struggles to fully grasp organisational context understanding. Hallucination remains a significant concern, as technology can produce inaccurate or fabricated information. Confidentiality concerns and environmental considerations were also mentioned.

Table 6. Identified limitations of GenAI use in project management

Limitation	Description
Inconsistent output quality	Participants reported that GenAI responses vary in accuracy and reliability, requiring human review to verify correctness and ensure the output is suitable for use.
Limited understanding of organisational context	GenAI struggles to interpret organisational structures, nuances, and internal practices, reducing its usefulness in context-dependent tasks.
Hallucinations	Several participants noted that GenAI can generate incorrect or fabricated information, posing risks when accuracy is critical.
Confidentiality concerns	Due to security and NDA requirements, participants avoid sharing confidential information with GenAI tools. GenAI is used similarly to public web search engines unless confidentiality can be guaranteed.
Environmental considerations	Some participants expressed concerns about the environmental impact of GenAI technologies, influencing how and when they use such tools.

5.5. Answers to research questions

Based on the discussion and qualitative analysis, we find the answers to the research questions as follows.

RQ1: How are IT project management professionals using GenAI tools today?

During the time, the IT project managers interviewed used GenAI primarily to discover knowledge and perform simple assistive tasks. They see GenAI as a helpful assistant whose outputs require refinement and verification. Client organisations where IT project management consultants work largely determine which tools are approved and available for use, thus shaping the adoption of GenAI usage. The primary GenAI usage activities are related to finding relevant information and best practices and content creation for presentations, reports, and communication materials. Moreover, quick translation needs for international projects, supporting meetings using transcriptions with relevant action point generation, and finally for brainstorming solutions seem to be practical assisted by GenAI.

RQ2: What vision do IT project management professionals have for the future use of AI?

In the future, many personal assistants, such as AI agents, are seen as important and can help with basic tasks that require less human thinking. However, control should be maintained by human IT project managers, and only assistant tasks that can be controlled and verified are assigned to AI assistants. The participants welcome agentic AI systems. They are perceived more as supportive assistants rather than autonomous peer IT project managers or instances that perform project management duties. There is also concern that training AI models on AI-generated content could lead to ”inbreeding,” potentially resulting in future AI systems that lack exposure to fresh, original data. Security and privacy concerns are viewed as significant risks in the adoption of AI.

6. Conclusions

This qualitative descriptive study documented GenAI usage examples and future expectations among 12 experienced Finnish IT project management consultants during August–September 2025. Through semi-structured interviews and hybrid thematic analysis, we examined how IT project managers currently use GenAI tools and what they envision for future AI applications in their work.

6.1. Theoretical implication

The findings indicate that the use of GenAI among the interviewed project managers follows a predominantly practical and situational logic rather than a framework-driven one. Tool usage is not systematically anchored in established project management methodologies, such as predictive, agile, or hybrid frameworks. Instead, GenAI appears to be appropriated in response to immediate cognitive or operational needs, suggesting that its role in project work is largely decoupled from formalised methodological structures. In general, the results highlight the need for project management theory to more explicitly address the informal and emerging use of GenAI technology.

6.2. Practical Implications

The findings suggest concrete implications for both IT project manager consultants and the organisations that employ them. The discussion reveals that GenAI adoption is shaped less by individual attitudes than by structural and organisational conditions, and consequently, the recommendations must address both levels. Table 7 (Page 17) identifies these challenges in targeted responses for two distinct stakeholder groups consulting project managers and their employers or client organisations structured around five key themes: organisational constraints and fragmented tool access, NDA and confidentiality restrictions, inconsistent output quality, limited understanding of organisational context, and uneven early-stage adoption. The recommendations reflect the finding that effective adoption of GenAI in IT project management consulting cannot be achieved through individual effort alone. Rather, it requires coordinated action in which practitioners continuously update a shared knowledge-base of GenAI tools and methods and disciplined working habits, while employers and client organisations provide the structural conditions by including clear policies, private deployment environments, reusable prompt resources, and knowledge-sharing forums that make responsible and consistent use of GenAI possible across diverse client engagements.

6.3. Limitations and validity

This qualitative descriptive study has several limitations that should inform interpretation and application of the findings. In qualitative descriptive research, the goal is not statistical generalisability, but transferability. The reader assesses whether the findings might apply to their contexts [6]. The detailed description of the characteristics of the participants (Tables 1, 2) and their contexts enable such transferability judgments.

With participants ($n = 12$), this study has achieved sufficient information power [39] to address exploratory research questions, given the narrow purpose of the study, the specific sample and the interview dialogue. However, fragmented usage examples (most activities mentioned by 1–3 participants) raise questions about whether a larger sample would reveal additional practices or stronger convergence practices. The project managers interviewed represent experienced professionals who can be assumed to be representative of a larger population of experts. Nevertheless, rare more specialised GenAI tool practices mentioned by single participants, for example, in Table 4, such as P4's Flowise usage, P11's Google Gemini usage may be more widespread than our sample suggests.

The interviews conducted in August–September 2025 capture a specific moment in the rapidly evolving landscape of generative AI technology. New tools, capabilities, organisational policies, and AI models emerge on a monthly basis. Specific tools (for example, the current dominance of ChatGPT and Microsoft Copilot) can quickly become outdated as others arrive. This temporal limitation is an inherent characteristic of the research on emerging technologies. The hybrid inductive-deductive analytical approach means that the findings are partially shaped by the structure of the interview guide. The five thematic domains (Section 3.3) were predetermined, while usage examples within those domains emerged inductively. Alternative interview designs (for example, fully open-ended narrative interviews) might have surfaced different themes.

The use of GenAI tools (ChatGPT, Claude) to help in transcript coding and categorisation (forming tables) represents both innovation and potential limitation. Although all AI-generated codes were manually verified and human in a loop modification required, the initial AI structure may have influenced the final coding schemes in subtle ways. This

Table 7. Recommendations for consulting project managers and their employers.

Challenge	Consulting project managers	Employers and client organisations
Organisational constraints and fragmented tool access	Maintain familiarity with multiple approved GenAI tools to remain productive across diverse client environments. Document which tools are permitted at each client engagement as part of onboarding.	Establish clear, written GenAI usage policies and approved tool lists at project start. Where possible, extend enterprise licences to external consultants to reduce fragmentation and enable consistent workflows.
NDA and confidentiality restrictions	Develop personal guidelines for classifying information before using GenAI, treating it as a public channel unless a private and confidential solution is confirmed. Use GenAI for structural, non-sensitive tasks such as templates, summaries of public information, and communication drafts.	Invest in private and confidential GenAI environments that allow consultants to work with project-specific data without breaching confidentiality agreements. Communicate data classification rules explicitly and early.
Inconsistent output quality	Use human-in-the-loop review and treat GenAI output as a first draft requiring critical evaluation rather than a final product. Build prompt refinement skills to improve output consistency over time.	Provide training on effective prompting and output evaluation. Include GenAI output quality as a topic in project quality assurance frameworks, rather than leaving verification entirely to individual discretion.
Limited understanding of organisational context	Compensate for GenAI's limited contextual awareness by supplying structured background information in prompts, such as project scope, stakeholder roles, organisational terminology, and relevant instructions. Maintain human ownership of context-dependent decisions and stakeholder communications.	Create reusable, organisation-specific prompt templates and context documents that consultants can use when engaging GenAI tools within a given client environment, reducing the burden of reinventing contextual groundwork at each new engagement.
Uneven and early-stage adoption	Proactively develop GenAI literacy beyond a single tool. Share effective use cases with colleagues to accelerate collective learning. Approach GenAI experimentation systematically by tracking which tasks benefit most from assistance.	Foster structured knowledge-sharing communities, for example internal practice groups or retrospectives, where project managers can exchange GenAI experiences. Recognise and reward experimentation to accelerate adoption beyond the current fragmented baseline.

transparency about AI-assisted analysis is methodologically important given the study's focus on GenAI adoption.

All interviews were conducted in Finnish and the transcripts were translated into English. Despite careful attention to preserving meaning, translation inevitably involves interpretation. Idiomatic expressions, cultural references, and nuanced meanings may not translate perfectly (for example, using spoken language). The interview data are based on participants' self-reported practices and recollections, which may differ from actual usage due to the following:

- memorable or recent GenAI interactions may be over-represented relative to routine usage,
- participants may over-report experimentation with innovative tools or under-report basic usage and
- unconscious habits or random usage may not be fully reported.

Despite these limitations, the study makes valuable contributions by providing systematic empirical documentation of GenAI adoption usage examples among experienced IT project management professionals during the year 2025. The qualitative descriptive approach [6, 7] offers accessible and practically orientated findings that can inform both practitioners who experiment with GenAI tools and researchers who design future studies. The documented diversity of GenAI tools, usage practices, adoption drivers, and future visions provides a foundation for understanding this emerging practice area.

6.4. Future research

This study demonstrates interest in AI assistants or agents, which opens compelling research opportunities to explore which aspects of IT project management work can be automated through cooperative AI-human collaboration. The interview data indicate areas that support further investigation in AI-assisted project management. As project management as a term is quite abstract, there are several possibilities to study. The scope and schedule of any project play a vital role, which opens up interesting research possibilities.

- Agent-assisted scope creep detection, where one can evaluate how effectively an autonomous AI agent can detect and flag scope creep in IT projects compared to PM-only monitoring, under varying project complexity conditions.
- Under what conditions does deploying an AI scheduling agent reduce project schedule variance and when might it introduce new sources of uncertainty.
- Examine human–AI collaboration dynamics to form a framework for future IT project management.

These research directions are essential for understanding the evolving nature of IT project management in organisations that are using GenAI or are beginning to use it to assist with project management tasks.

CRediT authorship contribution statement

Kari Sainio: Data curation, methodology, investigation, writing – original draft.
Petri Kettunen: Review, supervision.
Pekka Abrahamsson: Supervision.

Declaration of competing interest

The first author declares that he has no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

This research was performed within individual contribution without external funding.

Declaration of AI assistance in the writing process

For textual data analytics, ChatGPT by OpenAI and Claude by Anthropic have been used to process anonymised interview transcription data for coding and categorising purposes. During the preparation of this work, the first author used a generative AI tool called Writefull to improve the readability and English language. After using this tool, the first author reviewed and edited the content as needed and assumes full responsibility for the content of the publication.

References

[1] V. Holzmann, D. Zitter, and S. Peshkess, "The expectations of project managers from artificial intelligence: A Delphi study," *Project Management Journal*, Vol. 53, No. 5, 2022, pp. 438–455. [Online]. https://doi.org/10.1177/87569728211061779

[2] M.I. Hashfi and T. Raharjo, "Exploring the challenges and impacts of artificial intelligence implementation in project management: A systematic literature review," *International Journal of Advanced Computer Science and Applications*, Vol. 14, No. 9, 2023, place: West Yorkshire, United Kingdom. [Online]. https://www.proquest.com/docview/2883174147/abstract/F859DC41A1CA4252PQ/1

[3] G. Dhruva, I. Shettigar, S. Parthasarthy, and V.M. Sapna, "Agile project management using large language models," in *5th International Conference on Innovative Trends in Information Technology (ICITIIT)*, 2024, pp. 1–6. [Online]. https://ieeexplore.ieee.org/document/10580873

[4] A.M. Felicetti, A. Cimino, A. Mazzoleni, and S. Ammirato, "Artificial intelligence and project management: An empirical investigation on the appropriation of generative chatbots by project managers," *Journal of Innovation and Knowledge*, Vol. 9, No. 3, 2024, p. 100545. [Online]. https://www.sciencedirect.com/science/article/pii/S2444569X24000842

[5] I. Taboada, A. Daneshpajouh, N. Toledo, and T. de Vass, "Artificial intelligence enabled project management: A systematic literature review," *Applied Sciences (Switzerland)*, Vol. 13, No. 8, 2023, number: 8.

[6] M. Sandelowski, "Whatever happened to qualitative description?" *Research in Nursing and Health*, Vol. 23, No. 4, 2000, pp. 334–340. [Online]. https://onlinelibrary.wiley.com/doi/abs/10.1002/1098-240X%28200008%2923%3A4%3C334%3A%3AAID-NUR9%3E3.0.CO%3B2-G

[7] M. Sandelowski, "What's in a name? Qualitative description revisited," *Research in Nursing and Health*, Vol. 33, No. 1, 2010, pp. 77–84. [Online]. https://onlinelibrary.wiley.com/doi/pdf/10.1002/nur.20362

[8] "Commentary AI – 2023 – Alex Singla – Desktop," 2023. [Online]. http://ceros.mckinsey.com/commentary-ai-2023-alex-singla-desktop

[9] A.A. Tabassi, D.J. Bryde, E.M. Kamal, R. Michaelides, and J. Dowson, "Challenges for project management in the 21st century," in *The European Proceedings of Multidisciplinary Sciences*, 2019, p. 631–641. [Online]. https://www.researchgate.net/publication/330688957_Challenges_For_Project_Management_In_The_21st_Century

[10] T.O.A. Lehtinen, M.V. Mäntylä, J. Vanhanen, J. Itkonen, and C. Lassenius, "Perceived causes of software project failures – An analysis of their relationships," *Information and Software Technology*, Vol. 56, No. 6, 2014, pp. 623–643. [Online]. https://www.sciencedirect.com/science/article/pii/S0950584914000263

[11] A. Zeitoun and H. Kerzner, "Project management pain points and a path forward," *PM World Journal*, Vol. 2021, No. Vol. X, Issue X – October, 2021.

[12] "PMBOK® Guide," 2021. [Online]. https://www.pmi.org/pmbok-guide-standards/foundational/pmbok

[13] J. Varajão, H. Silva, and M. Pejic-Bach, "Key competences of information systems project managers," *International Journal of Information Technology Project Management*, Vol. 10, No. 3, 2019, pp. 73–90.
[14] J.K. Das, I. Elegbe, L. Coffie, R. Khadka, L. Chen et al., "AI-powered IT project management: Analyzing the effectiveness of advanced project management tools to ensure project efficiency," in *SoutheastCon*, 2025, pp. 1554–1559. [Online]. https://ieeexplore.ieee.org/document/10971718/
[15] B. Hettrich, N. Krings, and A. Kock, "Bridging the expertise gap: The role of generative AI in supporting project planning tasks for novices and professionals," *Creativity and Innovation Management*, Vol. 34, No. 4, 2025, pp. 789–805.
[16] V. Aramali, N. Cho, F. Pande, M.K.S. Al-Mhdawi, U. Ojiako et al., "Generative AI in project management: Impacts on corporate values, employee perceptions, and organizational practices," *Project Leadership and Society*, Vol. 6, 2025, p. 100191. [Online]. https://www.sciencedirect.com/science/article/pii/S266672152500016X
[17] C. Stryker and E. Kavlakoglu, "What is artificial intelligence (AI)?" 2024. [Online]. https://www.ibm.com/think/topics/artificial-intelligence
[18] R. Müller, G. Locatelli, V. Holzmann, M. Nilsson, and T. Sagay, "Artificial intelligence and project management: Empirical overview, state of the art, and guidelines for future research," *Project Management Journal*, Vol. 55, No. 1, 2024, pp. 9–15. [Online]. https://doi.org/10.1177/87569728231225198
[19] "PMI Infinity," 2024. [Online]. https://infinity.pmi.org/chat
[20] "ChatGPT (GPT-4)." [Online]. https://chat.openai.com/?model=gpt-4
[21] PMI.org, "Pushing the limits: Transforming project management with generative AI innovation," PMI.org, Tech. Rep., 2024. [Online]. https://www.pmi.org/learning/thought-leadership/transforming-project-management-with-generative-ai
[22] H. Dam, T. Tran, J. Grundy, A. Ghose, and Y. Kamei, "Towards effective AI-powered agile project management," 2019, pp. 41–44.
[23] Z. Yu and Y. Gong, "ChatGPT, AI-generated content, and engineering management," *Frontiers of Engineering Management*, Vol. 11, No. 1, 2024, pp. 159–166. [Online]. https://doi.org/10.1007/s42524-023-0289-6
[24] A. Barcaui and A. Monat, "Who is better in project planning? Generative artificial intelligence or project managers?" *Project Leadership and Society*, Vol. 4, 2023, p. 100101. [Online]. https://www.sciencedirect.com/science/article/pii/S2666721523000224
[25] D. Patel, "Revolutionizing project management with generative AI," 2023. [Online]. https://www.researchgate.net/publication/371313985_Revolutionizing_Project_Management_with_Generative_AI
[26] S. Bushuyev, D. Bushuiev, V. Bushuieva, and O. Ilin, "Managing digital transformation projects under the influence of AI," *CEUR Workshop Proceedings*, Vol. 2025, No. Vol. 4023, 2025, pp. 59–70.
[27] J. Gil, J. Martínez Torres, and R. González-Crespo, "The application of artificial intelligence in project management research: A review," *International Journal of Interactive Multimedia and Artificial Intelligence*, Vol. 6, No. 6, 2021, p. 54. [Online]. https://www.ijimai.org/journal/sites/default/files/2021-05/ijimai_6_6_6.pdf
[28] S. Karnouskos, "The relevance of large language models for project management," *IEEE Open Journal of the Industrial Electronics Society*, Vol. 5, 2024, pp. 758–768.
[29] M. Odeh, "The role of artificial intelligence in project management," *IEEE Engineering Management Review*, Vol. 51, No. 4, 2023, pp. 20–22.
[30] L. Greslehner, "Adapting to the Era of GenAI: Evolution of project management practices," 2024. [Online]. https://digital.pwc.at/2024/05/13/adapting-to-the-era-of-genai-evolution-of-project-management-practices/
[31] U. Flick, *The SAGE Handbook of Qualitative Research Design*. SAGE Publications Ltd, 2022. [Online]. https://methods.sagepub.com/hnbk/edvol/the-sage-handbook-of-qualitative-research-design-srm/toc

[32] V. Clarke and V. Braun, "Thematic analysis," *The Journal of Positive Psychology*, Vol. 12, No. 3, 2017, pp. 297–298. [Online]. https://www.tandfonline.com/doi/full/10.1080/17439760.2016.1262613
[33] L.A. Palinkas, S.M. Horwitz, C.A. Green, J.P. Wisdom, N. Duan et al., "Purposeful sampling for qualitative data collection and analysis in mixed method implementation research," *Administration and Policy in Mental Health and Mental Health Services Research*, Vol. 42, No. 5, 2015, pp. 533–544. [Online]. http://link.springer.com/10.1007/s10488-013-0528-y
[34] J. Fereday and E. Muir-Cochrane, "Demonstrating rigor using thematic analysis: A hybrid approach of inductive and deductive coding and theme development," *International Journal of Qualitative Methods*, Vol. 5, No. 1, 2006, pp. 80–92. [Online]. https://doi.org/10.1177/160940690600500107
[35] H.F. Hsieh and S.E. Shannon, "Three approaches to qualitative content analysis," *Qualitative Health Research*, Vol. 15, No. 9, 2005, pp. 1277–1288. [Online]. https://doi.org/10.1177/1049732305276687
[36] "Ollama," 2024. [Online]. https://github.com/ollama/ollama
[37] FlowiseAI, Inc., "Build AI agents, visually," 2023. [Online]. https://flowiseai.com/
[38] n8n.io, "AI workflow automation tool." [Online]. https://n8n.io/
[39] K. Malterud, V.D. Siersma, and A.D. Guassora, "Sample size in qualitative interview studies: Guided by information power," *Qualitative Health Research*, Vol. 26, No. 13, 2016, pp. 1753–1760. [Online]. https://doi.org/10.1177/1049732315617444

Appendix A. Interview guide

The interview was conducted in Finnish and was translated by the first author. Semi-structured interview questions were targeted at IT project managers who worked on several and different IT related projects within several client organisations. The recruitment process inherently favoured participants who had used different project management tools, either GenAI-driven or traditional.

Topics and guiding questions:

1. Work experience in project management.
   - *"Could you generally tell me what kind of experience you have in project management?"*
   - *"Can you tell us a little more about the history, more precisely what kinds of projects there have been?"*
2. Challenges in project or programme management.
   - *"What do you think have been the biggest challenges when working as project or programme management?"*
3. Non-AI tools used in project work.
   - *"What kind of tools have you used in this kind of work?"*
   - *"Have you been able to choose the tools yourself, or has the organisation required certain tools?"*
4. Use of generative artificial intelligence at work.
   - *"How have you used ChatGPT-type or similar generative AI tools in your work, and for what purposes?"*
   - *"What kind of things do you ask from artificial intelligence, and where do you seek help from it?"*
5. Future envisions for AI.

   - *"What would you like AI to do in the future to make work easier?"*
6. Autonomous AI agents.
   - *"Have you come across autonomous AI agents, and do you know what these are?"*
7. AI's influence on future project manager's job role.
   - *"How do you see artificial intelligence possibly changing your job description in the future?"*
8. Threats or possibilities of AI use.
   - *"Do you see any threats or possibilities in the use of artificial intelligence now or in the future?"*
9. The final open question.
   - *"Do you have anything else that you would like to add or ask for this interview?"*

Authors and affiliations

Kari Sainio
e-mail: kari.sainio@tuni.fi
ORCID: https://orcid.org/0009-0008-7705-9525
Faculty of Computing and Electrical Engineering, Tampere University, Finland

Petri Kettunen
e-mail: petri.kettunen@helsinki.fi
ORCID: https://orcid.org/0000-0002-2928-5885
Department of Computer Science, University of Helsinki, Finland

Pekka Abrahamsson
e-mail: pekka.abrahamsson@tuni.fi
ORCID: https://orcid.org/0000-0002-4360-2226
Faculty of Information Technology and Communication Sciences, Tampere University, Finland

e-Informatica Software Engineering Journal
Volume 20, Issue 1, pages: 1–21
DOI: 10.37190/e-Inf260105
BibTeX

Index-Based Type-3 Clone Detection

Zdenek Tronicek*
*Corresponding author: zdenek.tronicek@oneonta.edu

Article info

Keywords:
clone detection
code clones
near-miss clones

Submitted: 14 Dec. 2025
Revised: 16 Feb. 2026
Accepted: 20 Apr. 2026
Available online: 23 Apr. 2026

Abstract

Context: Clone detection is a common task in software engineering. Type-3 clones are fragments of code that can be slightly different in structure.
Objective: The article presents a new algorithm for Type-3 clone detection, its open-source implementation called *DrDupLex3*, and novel open-source tools that can be used for the automated assessment of Type-3 clones and to prepare training sets for machine learning-based clone detectors.
Method: The algorithm for Type-3 clone detection builds upon the index of source code used in *DrDupLex*, the most accurate Type-2 clone detector to date.
Results: A comparison with three state-of-the-art clone detectors (*NiCad*, *CloneWorks*, and *SourcererCC*) shows that *DrDupLex3* is able to outperform them in precision, recall, and running time. It reported no false positives and found all clones reported by *NiCad*, *CloneWorks*, and *SourcererCC*.
Conclusions: The presented clone detector outperforms three state-of-the-art competitors in a scenario that can be easily repeated because it is based on tools for automated assessment of Type-3 clones.

1. Introduction

Code clones are a well-established research area in software engineering with applications in software maintenance, refactoring, vulnerability detection, program comprehension, evolution analysis, and plagiarism detection. One fundamental problem in this area is the clone detection problem, which is to identify all code clones in a given code base. Despite the simplicity of its definition, the problem is nontrivial and has been attracting the attention of researchers for more than two decades. This is partially because it is not a single problem, but rather a suite of problems, which differ in the definition of the clone. In this article, a clone is a set of code fragments that are similar to each other. Clones can be reported either as clone pairs (two code fragments) or clone classes (two or more code fragments). We typically distinguish four categories of clones [1–3]:

– Type-1 (exact) clones are code fragments that are identical or differ only in white spaces and comments.
– Type-2 (renamed) clones are fragments of code that are either Type-1 clones or differ in identifiers and literals. If identifiers are systematically renamed, the clone is called parameterized.

© 2026 The Authors. Published by Wrocław University of Science and Technology Publishing House.
This is an open access article under the CC BY-SA 4.0 license international.

– Type-3 (near-miss) clones are fragments of code that are either Type-2 clones or differ in code structure. They appear in the code base when we copy a code fragment and add, modify, or remove statements.
– Type-4 (semantic) clones are fragments of code that are semantically equivalent, i.e., they provide the same functionality. There is no requirement for syntactic similarity.

The first three types of clones are related because Type-2 clones involve Type-1 clones, and Type-3 clones involve Type-2 clones; however, Type-4 clones are different because for two code fragments to be a Type-4 clone, the code fragments do not need to be syntactically similar. This is why algorithms for identifying Type-4 clones are typically very different from algorithms for identifying Type-1, Type-2, and Type-3 clones.

Another parameter of clone detection is code granularity. Common values of granularity are "statements", "methods" (or "functions"), and "files". While other granularities are also possible, most application areas require one of these three.

There are dozens of approaches to clone detection described in the literature, and they can be classified by various criteria. One common approach is to classify them based on how they compare code fragments [4–7]:

– Text-based clone detectors compare the textual representation of code [8, 9].
– Token-based clone detectors translate code to a sequence of tokens and compare sequences of tokens [10, 11].
– Tree-based clone detectors represent code as a tree, such as an abstract syntax tree, and compare trees or their representation [12].
– Graph-based clone detectors translate code to a graph, such as a Program Dependence Graph, and compare graphs [13–15].
– Metrics-based clone detectors use code metrics to detect clones [16].
– Hybrid clone detection techniques combine two or more approaches mentioned above [17].

Commonly, clone detectors in the same class of this classification share the same traits. For example, token-based clone detectors typically scale better than graph-based ones because building a graph code representation requires more resources than tokenization.

Like many other areas of software engineering, clone detection was also affected by development in the area of artificial intelligence [18]. Multiple researchers have explored various machine learning-based tools capable of classifying code fragments as clones or nonclones. Although these tools are called "clone detectors", they do not detect clones like traditional clone detectors. They only answer the question "Are these two code fragments a clone?" and sometimes provide a similarity score. In practice, they can hardly replace traditional clone detectors, which detect all Type-3 clones that are within the specified similarity.

This article deals with Type-3 clone detection. Two code fragments are considered a Type-3 clone if their similarity is above some specified threshold. There are various ways in which the similarity of code fragments can be measured, and it is common that the algorithm used for computing similarity is bound to the clone detection method. For example, text-based clone detectors typically compute the longest common subsequence of sequences of lines, and *Deckard* (see section Related Work for details) computes the distance of characteristic vectors in Euclidean space. The diversity of algorithms used for computing code similarity in clone detectors makes a comparison of clone detectors challenging because it is not always clear how to transform one similarity into the other. In addition, code fragments that are similar in one similarity can be very dissimilar in a different similarity. One approach to overcome these difficulties is to divide clone detectors into categories based on how they compute code similarity and compare them only with

clone detectors in the same category. This article introduces a category of "clone detectors that measure code similarity by the edit distance of token sequences" and presents the best clone detector in this category. The edit distance (also known as the Levenshtein distance) is the minimum number of operations substitution, insertion, and deletion required to transform one sequence of tokens into the other. This distance is particularly intuitive for code fragments that are not very distant, such as when they differ only in a few operators. On the other hand, for code fragments that differ in whole lines, the edit distance that allows block operations may be a better choice.

The rest of this article is organized as follows: Section 2 presents the motivation for this research, Section 3 summarizes the scientific contribution of this article, Section 4 describes a novel algorithm for Type-3 clone detection, Section 5 introduces an open-source clone detector called *DrDupLex3*, Section 6 discusses comparison of *DrDupLex3* with state-of-the-art competitors, Section 7 introduces open-source tools for automated evaluation of clone detectors, Section 8 outlines conducted experiments, Section 9 addresses threats to validity, Section 10 states the main limitations of the presented approach, Section 11 briefly mentions the most important related work, and Section 12 concludes.

2. Motivation

The primary motivation for research on code clones stems from their applications in software engineering. Code clone detection facilitates various tasks critical to software quality and maintainability, including:

- Refactoring [19, 20]: By identifying code clones, developers can refactor them into reusable functions or methods, reducing code duplication and improving code maintainability.
- Code maintenance [21, 22]: Detecting clones helps ensure consistent updates across all code instances, minimizing the risk of introducing bugs due to inadvertent modifications in one location but not others.
- Vulnerable code detection [23–26]: Clone detection algorithms play a crucial role in identifying potentially vulnerable code in various areas, such as cryptocurrencies [27, 28], Android applications [29–31], and malware [32].
- License compliance [33]: Identifying inter-project code clones can help developers avoid potential license infringement issues, particularly when dealing with slightly modified code snippets.

While solutions to these tasks typically involve Type-2 clone detection, they can often benefit from Type-3 clone detection, too.

The diversity in applications highlights the need for recognizing clone detection as a multifaceted problem with multiple solutions, each offering different results. Categorizing clone detectors enables researchers to conduct more scientific comparisons and facilitates developers in selecting the most appropriate approach for a specific task.

3. Contribution

The scientific contribution of this article is a novel algorithm for detecting Type-3 clones and a suite of open-source tools for an automated evaluation of Type-3 clone detectors. The algorithm identifies all Type-3 clones within the codebase; because this comprehensive

detection is significantly more demanding than simple pairwise classification, machine learning-based clone classifiers are not direct competitors to this approach.

The algorithm leverages the index of the source code employed in *DrDupLex* [34], the most accurate Type-2 clone detector to date. This index, a trie data structure built on sequences of tokens, facilitates the identification of Type-1 and Type-2 clones and the elimination of candidates that are not within the specified edit distance. Unlike previous approaches, the index can be easily updated upon a change in source code, which makes the presented solution suitable for reporting clones "online", i.e., upon each change in source code. The implementation of the algorithm achieves 100% precision and 100% relative recall (no known clone was missed). The presented evaluation is automated and can be easily reproduced.

4. Algorithm

Although the algorithm description utilizes Java programming language terminology, the underlying concept applies to other programming languages as well. The description references parsing for syntactic unit identification; however, full parsing is not required. Any parsing capable of detecting syntactic unit boundaries is sufficient. The algorithm can operate on diverse syntactic units, such as methods or statements, and their combinations. It consists of five steps (see also Figure 1):

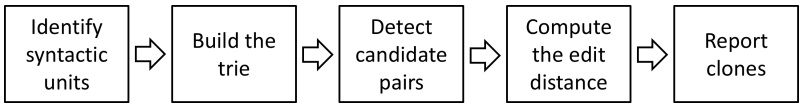


Figure 1. The clone detection has five steps: 1. Identify syntactic units and translate them into sequences of tokens; 2. Build the trie on sequences of tokens; 3. Group the sequences of tokens by length and identify pairs of sequences that can be within the specified distance; 4. Compute the edit distance for each candidate pair; 5. Report the pairs that are within the specified distance.

1. Identify syntactic units (this typically involves parsing) and translate them into sequences of tokens; for example, the statement "b = a + 1" is translated into the sequence "ID = ID + INT".
2. Build a trie on these sequences (see Figure 2 for an example); this is the same trie as is used to detect Type-2 clones in *DrDupLex*; Type-2 clones have the same sequences of tokens and thus they end in the same node of the trie.
3. Group the sequences of tokens by length and identify pairs of sequences that can be within the specified distance; for example, sequences of lengths 20 and 23 can be within

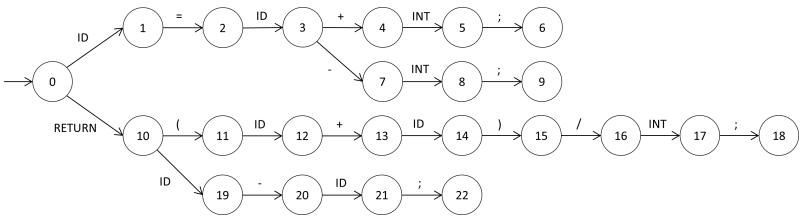


Figure 2. The trie for statements "b = a + 1 ;", "c = a – 1 ;", "y = x + 2 ;", "return (a + b) / 2 ;", "return a – b ;", and "return x – y ;".

the distance 3, but sequences of lengths 20 and 24 cannot be within that distance because 4 operations are needed just to make the lengths of the sequences equal.

4. Compute the edit distance for each candidate pair; the edit distance is computed between two sequences of tokens; *DrDupLex3* uses the algorithm "dynamic programming" [35], which is terminated as soon as it is clear that the sequences are not a clone; multiple threads are employed to compute the distances concurrently.
5. Report pairs that are within the specified distance.

The trie constructed in step 2 facilitates the identification of syntactic units for which the edit distance needs to be computed. As an example, let's consider the trie in Figure 2 and the maximum distance 1. Since "b = a + 1 ;" and "y = x + 2 ;" are a Type-2 clone, as well as "return a – b ;" and "return x – y ;", we get only four sequences of tokens:

– ID = ID + INT ;
– ID = ID – INT ;
– RETURN (ID + ID) / INT ;
– RETURN ID – ID ;

They form six pairs for which we need to compute the edit distance. When we group the sequences by length, we get one sequence with 5 tokens, two sequences with 6 tokens, and one sequence with 9 tokens. Pairs whose length difference exceeds the maximum distance can be eliminated because the length difference is the lower bound for the edit distance. So, we need to compute the edit distance between three pairs only:

– "ID = ID + INT ;" and "ID = ID – INT ;"
– "ID = ID + INT ;" and "RETURN ID – ID ;"
– "ID = ID – INT ;" and "RETURN ID – ID ;"

This is substantially less than the 10 pairs required if we omit to identify Type-2 clones and only cluster the methods by length.

The presented trie may seem similar to the suffix tree used in *CCFinder* [10] (see Related Work for details); however, there are two significant differences: (i) The suffix tree in *CCFinder* is built on the file level (the files are translated into sequences of tokens and the suffix tree is built on the concatenated sequences of tokens), which means that *CCFinder* does not know where syntactic units begin and end. This is probably why it does not report Type-3 clones. (ii) The trie can be modified upon a change in the source code. We can delete an original sequence of tokens and add a new sequence of tokens in time linear in the length of this sequence (see [34] for details). In contrast, a suffix tree requires rebuilding from scratch. This makes the presented algorithm suitable for reporting clones upon each change in source code. For example, an Integrated Development Environment can report clones while the developer writes code.

5. Implementation

The algorithm is implemented for the Java programming language in an open-source tool called *DrDupLex3*[1]. Like *DrDupLex*, it leverages JavaParser[2] to parse source code and builds the index, which is either plain or compressed, either in memory or on secondary storage, and either method-level or statement-level. *DrDupLex3* uses the naïve algorithm and blind renaming (see [34] for explanation of these terms). The edit distance between two token

[1] https://github.com/tronicek/DrDupLex3
[2] https://javaparser.org

sequences $a = a_1a_2 \dots a_m$ and $b = b_1b_2 \dots b_n$ is computed by "dynamic programming", which utilizes a matrix D defined as follows: $D_{i,0} = i$, for $0 \leq i \leq m$, $D_{0,j} = j$, for $1 \leq j \leq n$, and

$$D_{i,j} = \begin{cases} D_{i-1,j-1} & \text{if } a_i = b_j, \\ 1 + min(D_{i-1,j}, D_{i,j-1}, D_{i-1,j-1}) & \text{otherwise, for } 1 \leq i \leq m \text{ and } 1 \leq j \leq n. \end{cases}$$

When the matrix is calculated, the value of $D_{m,n}$ is the edit distance between a and b.

The run and output of *DrDupLex3* are controlled by a configuration file, which can contain the following parameters:

- *type* specifies the type of clones; accepted values are "2" (only Type-2 clones), "3" (only Type-3 clones that are not Type-2), and "2+3" (all Type-3 clones).
- *maxDistance* specifies the maximum edit distance between two code fragments; for example, if *maxDistance* is 3, only the code fragments with an edit distance of 3 or less are reported as a Type-3 clone.
- *level* specifies the granularity of the index; accepted values are "method" and "statement".
- *compressed* specifies whether the index is compressed or not; accepted values are "true" and "false".
- *persistent* specifies whether the index is built in main memory or on secondary storage; accepted values are "true" and "false". The persistent index is implemented using memory-mapped files and can be much larger than the nonpersistent index because it never needs to be fully loaded in memory.
- *minSize* specifies the minimum number of lines; for example, if *minSize* is 5, the code fragment must have at least 5 lines to be reported.
- *ignoreUnaryAtLiterals* specifies how the unary plus and minus are treated; accepted values are "true" and "false". If true, "return 1" and "return -1" are considered a Type-2 clone.
- *ignoreAnnotations* specifies whether annotations in code are taken into account; accepted values are "true" and "false".
- *treatNullAsLiteral* specifies how "null" is treated; accepted values are "true" and "false". If true, "return null" and "return 0" are considered a Type-2 clone.
- *treatSuperThisAsIdentifier* specifies whether "super" and "this" are treated as identifiers; accepted values are "true" and "false". If true, "this()" and "print()" are considered a Type-2 clone, as well as "this.size=1" and "p.length=2".
- *threads* specifies the number of concurrent threads employed to compute edit distances between code fragments.
- *batchFileSize* specifies how many files are processed before the index in memory is merged with the persistent index. A higher value means better throughput, but also higher memory usage.

Parameters *ignoreUnaryAtLiterals*, *ignoreAnnotations*, *treatNullAsLiteral*, and *treatSuperThisAsIdentifier* were added to facilitate comparing with other tools. For example, if a tool ignores annotations, we can configure *DrDupLex3* to ignore them as well.

6. Evaluation

Search algorithms are typically evaluated in terms of precision and recall. Precision is defined as the fraction of retrieved occurrences that are relevant:

$$precision = \frac{|\{relevant\} \cap \{retrieved\}|}{|\{retrieved\}|}$$

Recall is defined as the fraction of relevant occurrences that are retrieved:

$$recall = \frac{|\{relevant\} \cap \{retrieved\}|}{|\{relevant\}|}$$

Two types of errors can occur: false positives and false negatives. A false positive arises when an identified code fragment is not a clone, and a false negative occurs when a genuine clone remains undetected. Both false positives and false negatives are undesirable and should be eliminated.

I considered the following clone detectors for comparing with *DrDupLex3* (see section Related Work for details): *CCAligner*, *CloneWorks*, *Code2Img*, *Deckard*, *NiCad*, *NIL*, *Siamese*, *SourcererCC*, *StoneDetector*, *TACC*, and *Tamer*. However, none of them compute similarity in the same way as *DrDupLex3*:

- *CCAligner* [36] computes asymmetric Dice similarity between code windows;
- *CloneWorks* [37] computes Jaccard similarity between code fragments (but does not report it);
- *Code2Img* [38] computes Jaccard similarity of vectors that encode images created based on the adjacency matrix of the normalized abstract syntax tree;
- *Deckard* [39] computes similarity using the Euclidean distance of vectors;
- *NiCad* [40] computes similarity from the longest common subsequence of two sequences of code lines;
- *NIL* [41] computes similarity from the longest common subsequence of two token sequences;
- *Siamese* [42] computes similarity based on n-grams;
- *SourcererCC* [43] computes Jaccard similarity between code fragments (but does not report it);
- *StoneDetector* [44] computes similarity from the textual representations of paths in dominator trees;
- *TACC* [45] computes Jaccard similarity of vectors;
- *Tamer* [46] computes similarity based on the corresponding subtrees in abstract syntax trees.

How can we compare *DrDupLex3* with a tool that computes a different similarity? Let's say we select a benchmark project and let the tool identify all clones with a specified similarity threshold. Then we let *DrDupLex3* identify all clones with the equivalent edit distance and compare the outputs. This can work if we can transform the tool's similarity into the edit distance. Unfortunately, no such transformation is available for the similarities mentioned above. Another approach can be based on estimates. Let's say tool A identifies all clones with a similarity threshold s, and from s we are able to compute d such that all these clones have the edit distance at most d. Then we let *DrDupLex3* identify all clones with the maximum distance d and check that it reported all clones found by tool A. And then the opposite way: We let *DrDupLex3* identify all clones within a specified distance and then check that tool A found them all. This can work if we are able to estimate the edit distance based on a given similarity threshold.

As an example, let's consider the similarity measured by the percentage of identical lines. Let ed be the edit distance of two code fragments, $lines$ the number of lines of the larger fragment, and $lcssim$ the percentage of identical lines (i.e., $lcssim$ is the ratio of the number of identical lines and $lines$), and let's try to estimate ed for a given value of $lcssim$. If there is at least one different line ($lcssim < 100\%$), ed can be any value ≥ 1 because a line can contain any number of changes. Now let's estimate $lcssim$ for a given value of ed. If code changes are distributed so that there is no more than one change per line (provided that $ed \leq lines$), $lcssim$ is minimal, and if all code changes are on one line, $lcssim$ is maximal. Thus

$$\frac{\max(0, lines - ed)}{lines} \leq lcssim \leq \frac{lines - 1}{lines}.$$

This can be used to make estimates in some particular cases. For example, if the edit distance of two code fragments is 1, only one line can be different, and if $lines$ is at least 5, the similarity of the code fragments is at least $4/5 = 80\%$.

As another example, let's consider the Jaccard similarity, which calculates similarity between two sequences of tokens as the ratio of the number of shared tokens and the number of tokens in the longer sequence. For example, the Jaccard similarity of expressions "x + y" and "x + y + z" is 3/5 and of expressions "x – 1" and "1 – x" is 1. Let f_1, f_2 be two sequences of tokens, ed the edit distance of f_1 and f_2, $jsim$ the Jaccard similarity of f_1 and f_2, $shared$ the number of shared tokens between f_1 and f_2, and $maxlen$ the maximum of $|f_1|$ and $|f_2|$, and let's estimate ed for a given value of $jsim$. From the equation $jsim = shared/maxlen$ and the estimate $maxlen - shared \leq ed \leq maxlen$, we get

$$maxlen(1 - jsim) \leq ed \leq maxlen.$$

Now let's estimate $jsim$ for a given ed. For $ed = 1$, $jsim = (maxlen - 1)/maxlen$, and for $ed \geq 2$,

$$\frac{maxlen - ed}{maxlen} \leq jsim \leq 1.$$

The upper bound is tight, for example, when all changes are substitutions, such as for sequences $t_1\ t_2\ t_3\ t_4\ t_5$ and $t_5\ t_4\ t_3\ t_2\ t_1$ (we substitute t_5 for t_1, t_4 for t_2, and so on).

NIL may look like a good candidate for comparing because it works with tokens; however, it does not do any lexical analysis and uses sequences of input characters as tokens. For example, a statement "a = (int) (b + 0.5);" is converted into a sequence "a = (int) (b + 0.5) ;" with no information that "a", "int", and "b" are identifiers and "0.5" is a double literal. A consequence is that *NIL* does not recognize Type-2 clones that are not Type-1 clones. It reports only Type-1 clones and a specific subset of Type-3 clones (those created by inserting statements into a Type-1 clone). In other words, *NIL* detects different types of clones than *DrDupLex3* and so a direct comparison of these tools would be uninformative.

Other problems exist with the clone detectors mentioned above: *CCAligner* hangs up and does not provide any output, *Deckard* works only on small code bases, *Code2Img*, *StoneDetector*, and *Tamer* produce an output that does not contain enough information for identifying code fragments, *Siamese* compiles and runs but does not work, and *TACC* does not compile.

Based on these observations, I decided to continue with *NiCad* (version 6.2), *CloneWorks* (version 0.3), and *SourcererCC* (version on GitHub with the latest commit on May 17,

```
double root1(int a, int b, int c) {
return(-b+Math.sqrt(b*b-4*a*c))/(2*a);
}
```

```
double root2(int a, int b, int c) {
return(-b-Math.sqrt(b*b-4*a*c))/(2*a);
}
```

Figure 3. Two methods that are assessed very differently by *DrDupLex3* and *NiCad*. For *DrDupLex3*, the methods differ in one token, and so it reports a clone with edit distance 1. For *NiCad*, the methods share two lines, but the lines "}" are not counted, and so it reports a clone with similarity 50%.

```
public String getBaseTypeName() {
  debugCodeCall("getBaseTypeName");
  checkClosed();
  return "NULL";
}
```

```
public int getMaxFieldSize() {
  debugCodeCall("getMaxFieldSize", 10);
  checkClosed();
  return 0;
}
```

Figure 4. Two methods that are assessed very differently by *DrDupLex3* and *NiCad*. For *DrDupLex3*, the methods differ in two tokens, and so *DrDupLex3* reports a clone with edit distance 2. For *NiCad*, the first lines are different because it does not recognize "int" as an identifier, and the fourth lines are either different or equal, depending on the "abstract" parameter. Thus, *NiCad* reports a clone with similarity either 25% (with abstract=none) or 50% (with abstract=literal).

```
public int gcd(int a, int b) {
    while (a != b)
        if (a < b)
            b -= a;
        else
            a -= b;
    return a;
}
```

```
public int gcd (int a, int b) {
    while (a != b) if (a < b) b -= a;
    else a -= b;
    return a;
}
```

Figure 5. A fragment of code (on the left) and its formatting by *NiCad* (on the right).

2022) and show they do not report some of the clones reported by *DrDupLex3*. In addition to differences in similarity, the tools differ in how the size of clones is determined: *NiCad* reports pretty-printed lines after removing comments, *CloneWorks* and *SourcererCC* report no lines, and *DrDupLex3* reports real lines.

Since *DrDupLex* can outperform *NiCad*, *CloneWorks*, and *SourcererCC* in precision and recall when searching for Type-2 clones [34] and *DrDupLex3* reports the same Type-2 clones as *DrDupLex*, I focused on Type-3 clones that are not Type-2 clones. *NiCad* uses the longest common subsequence algorithm to compute similarity and *DrDupLex3* computes the edit distance between sequences of tokens, which may result in significant differences. As an example, Figure 3 and 4 show code fragments that are very similar for *DrDupLex3* and only moderately similar for *NiCad*. In Figure 3, *NiCad* considers the return statements different and disregards the fact that they are different in only one character. In Figure 4, *NiCad* does not recognize "int" as an identifier, which results in differences in the first lines of code. Since the second lines are also different and the lines "}" are not counted, the similarity of code fragments is either 25% (if *NiCad* runs with parameter abstract=none) or 50% (if *NiCad* runs with parameter abstract=literal).

CloneWorks and *SourcererCC* use Jaccard similarity to measure similarity of code fragments, which is sometimes very different from similarity based on edit distance. As an example, Figure 6 shows code fragments that are a Type-2 clone for *CloneWorks* and *SourcererCC* and Type-3 clone for *DrDupLex3*.

```
int poly1(int x) {
  return x * x - x - 1;
}
```

```
int poly2(int x) {
  return 1 - x - x * x;
}
```

Figure 6. Two methods that are assessed very differently by *DrDupLex3* and *CloneWorks* and *SourcererCC*. *DrDupLex3* reports a clone with edit distance 4. *CloneWorks* and *SourcererCC* use Jaccard similarity and report a clone with similarity 1.

A convenient approach to comparing with *NiCad* would be to modify *DrDupLex3* so that it reports the same similarity as *NiCad*; however, this is not easy to achieve because of how *NiCad* preprocesses source code before computing the longest common subsequence: it removes comments and formats source code by a pretty printer that is sometimes hard to reproduce (see Figure 5).

7. Tools

The evaluation was done on BigCloneBench [47] (the reduced version of IJaDataset 2.0), a well-established benchmark for clone detection. BigCloneBench consists of a large-scale source code repository and a clone database; however, only the source code repository is utilized in this evaluation. The rationale for excluding the clone database is that it has been shown to contain less than 3.4% of the actual Type-1 and Type-2 clones [34]. Furthermore, BigCloneBench's Type-3 clones use *NiCad* similarity.

I designed and implemented the following tools in open-source projects *EvalTool*[3] and *CloneDistance*[4] (they all work with an input XML file that contains clones):

- *FilterDistance* selects clones based on the XML attribute *distance*; the minimum and maximum distance must be specified.
- *FilterLines* selects clones based on the number of real lines of code; the minimum and maximum number of lines must be specified.
- *FilterSimilarity* selects clones based on the XML attribute *similarity*; the minimum and maximum similarity must be specified.
- *EditDistance* computes the edit distance of code fragments and stores it in the XML attribute *distance*.

I also modified the open-source project *CloneChecker*[5] [34] so that it can check Type-3 clones. When it runs with parameter type=3, it classifies clones as Type-1, Type-2, Type-3, "not a clone", or invalid. Classification "invalid" is used when the code fragments do not parse. If the code fragments parse but their distance differs from the value in the XML attribute *distance*, the clone is classified as "not a clone". In addition to these tools, the evaluation leverages the tools that were already described in [34]:

- *Counter* counts clones in the input XML file.
- *Diff* subtracts two XML files with clones and reports those in the first file and not in the second file.
- *Separator* separates clones in the input XML file into clone pairs. For example, a clone of A, B, and C is separated into clone pairs {A, B}, {B, C}, and {A, C}.
- *Sourcer* extracts code fragments from a code base and inserts them into the input XML file. The code fragments can then be checked visually or by *CloneChecker*.

[3] https://github.com/tronicek/EvalTool
[4] https://github.com/tronicek/CloneDistance
[5] https://github.com/tronicek/CloneChecker

8. Experiments

To evaluate precision, I ran *NiCad* with parameters rename=blind, abstract=literal (to treat all literals equal), threshold=0.01, and minsize=20, added code fragments to the output XML file (by *Sourcer*), computed distances between code fragments (by *EditDistance*), and classified the clones (by *CloneChecker*). Then I repeated that for *CloneWorks* with parameters rename-blind, abstract literal (to treat all literals equal), min-similarity 0.99, and min-lines 20. Next, I ran *SourcererCC* with parameters MIN_TOKENS=40 and threshold=9.9, filtered clones shorter than 20 lines (by *FilterLines*), added code fragments to the output XML file, computed distances between code fragments, and classified the clones. Finally, I ran *DrDupLex3* with parameters type=2+3, level=method, maxDistance=1, and minSize=20, separated clones into clone pairs (by *Separator*), added code fragments to the output XML file, and classified the clones (see Figure 7 and Table 1).

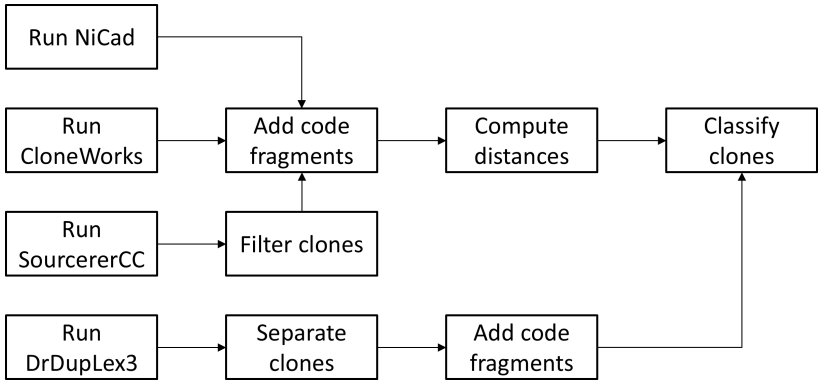


Figure 7. The flowchart of how precision was evaluated. There is a path for each clone detector. For example, the path for *NiCad* is "Run *NiCad*, Add code fragments, Compute distances, Classify clones". The following tools were used: *Sourcer* to add code fragments, *EditDistance* to compute distances between code fragments, *CloneChecker* to classify clones, *FilterLines* to filter clones, and *Separator* to separate clones into clone pairs.

Table 1. The clone pairs reported in BigCloneBench by *NiCad* (rename=blind, abstract=literal, threshold=0.01, minsize=20), *CloneWorks* (rename-blind, abstract literal, min-similarity 0.99, min-lines 20), *SourcererCC* (MIN_TOKENS=40, threshold=9.9), and *DrDupLex3* (type=2+3, level=method, maxDistance=1, minSize=20) and their classification by *CloneChecker*. The precision is computed based on this classification. *NiCad* seemingly reported more Type-1 clones than *DrDupLex3* because it ignores annotations and uses pretty-printed lines. For instance, two methods that differ only in annotations are reported as a Type-1 clone by *NiCad*. *SourcererCC* reported more Type-3 clones than the other tools, but only 409 of them had distance=1.

Tool	Reported	Type 1	Type 2	Type 3	Invalid	Precision
NiCad	258,845	226,503	31,836	435	71	99.97%
CloneWorks	258,870	224,216	30,949	3,633	72	99.97%
SourcererCC	235,737	213,847	2,142	7,200	12,548	94.68%
DrDupLex3	267,653	224,496	38,367	4,790	0	100%

To evaluate Type-3 precision, I ran *NiCad* with parameters rename=blind, abstract=literal, threshold=0.05, and minsize=20, filtered out Type-2 clones (by *FilterSimilarity*), added code fragments to the output XML file, computed distances between code fragments, and classified the clones. The clones classified as Type-2 were incorrectly reported with

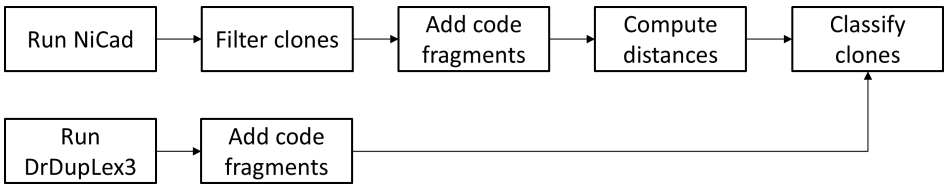


Figure 8. The flowchart of how Type-3 precision was evaluated. There is a path for each clone detector. The path for *NiCad* is "Run *NiCad*, Filter clones, Add code fragments, Compute distances, Classify clones" and the path for *DrDupLex3* is "Run *DrDupLex3*, Add code fragments, Classify clones". The following tools were used: *FilterSimilarity* to filter clones, *Sourcer* to add code fragments, *EditDistance* to compute distances between code fragments, and *CloneChecker* to classify clones

Table 2. The clone pairs reported as Type-3 clone pairs in BigCloneBench by *NiCad* (rename=blind, abstract=literal, threshold=0.05, minsize=20) and *DrDupLex3* (type=3, level=method, maxDistance=4, minSize=20) and their classification by *CloneChecker*. Type-2 clones were filtered out by *FilterSimilarity*. *CloneWorks* and *SourcererCC* are not involved because they do not report similarity.

Tool	Reported	Type 1	Type 2	Type 3	Invalid	Type-3 precision
NiCad	20,032	0	284	19,744	4	98.56%
DrDupLex3	23,659	0	0	23,659	0	100%

a similarity of less than 100. Then I ran *DrDupLex3* with parameters type=3, level=method, maxDistance=4, and minSize=20, added code fragments to the output XML file, and classified the clones (see Figure 8 and Table 2).

To evaluate recall, I ran *NiCad* with parameters rename=blind, abstract=literal, threshold=0.10, and minsize=20, added code fragments to the output XML file, computed distances between code fragments (this eliminated invalid clones), and filtered out clones with a distance greater than 1. Then I ran *DrDupLex3* with parameters type=2+3, level=method, maxDistance=1, minSize=6, and ignoreAnnotations=true (because *NiCad* ignores annotations), subtracted the output from *NiCad*'s output (by *Diff*), and got 13 clones. All 13 clones were methods with less than 15 lines of code that were pretty-printed on more than 20 lines. Then I ran *DrDupLex3* with parameters type=2+3, level=method, maxDistance=1, and minSize=40, separated the clones into pairs, ran *NiCad* with parameters rename=blind, abstract=literal, threshold=0.20, and minsize=10, subtracted the

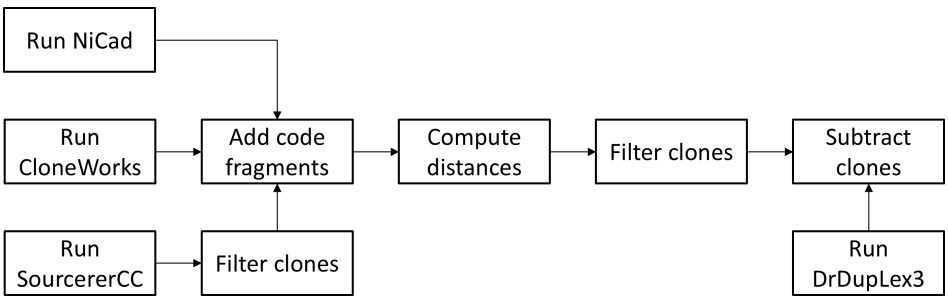


Figure 9. The flowchart describing how the output of *DrDupLex3* was subtracted from the output of *NiCad*, *CloneWorks*, and *SourcererCC* (the results are in Table 3 and 4). Each clone detector has its own path. For example, the path for *NiCad* is "Run *NiCad*, Add code fragments, Compute distances, Filter clones, Subtract clones detected by *DrDupLex3*". The following tools were used: *Sourcer* to add code fragments, *EditDistance* to compute distances between code fragments, *FilterDistance* to filter clones, and *Diff* to subtract clones.

output from *DrDupLex3*'s output, added code fragments to the output XML file, and classified the clones (see Figure 9 and 10).

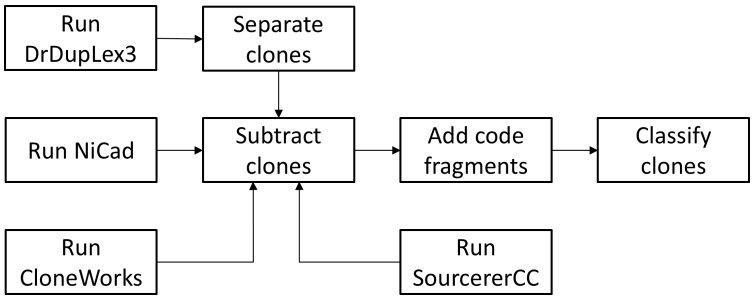


Figure 10. The flowchart describing how the output of *NiCad*, *CloneWorks*, and *SourcererCC* was subtracted from the output of *DrDupLex3* (the results are in Table 3 and 4). Each clone detector has its own path. For example, the path for *NiCad* is "Run *NiCad*, Subtract clones from the output of *DrDupLex3*, Add code fragments, Classify clones". The following tools were used: *Separator* to separate clones into clone pairs, *Diff* to subtract clones, *Sourcer* to add code fragments, and *CloneChecker* to classify clones.

Then I ran *CloneWorks* with parameters rename-blind, abstract literal, min-similarity 0.90, and min-lines 20, added code fragments to the output XML file, computed distances between code fragments (this eliminated invalid clones), filtered out clones with a distance greater than 1, subtracted the output of *DrDupLex3* with parameters type=2+3, level=method, maxDistance=1, minSize=6, and ignoreAnnotations=true (because *CloneWorks* ignores annotations), and got an empty XML file. Then I ran *CloneWorks* with parameters

Table 3. The output of tool B was subtracted from the output of tool A. The output of *NiCad*, *CloneWorks*, and *SourcererCC* in the column "Tool A" was filtered by *FilterDistance* and *FilterLines*. The column "Difference" is the number of clones with distance=1 reported by tool A and missed by tool B. These values were used to calculate recall in Table 4.

Tool A	Tool B	Difference
NiCad, rename=blind, abstract=literal, threshold=0.10, minsize=20 *FilterDistance*	*DrDupLex3*, type=2+3, level=method, maxDistance=1, minSize=6, ignoreAnnotations=true	13
DrDupLex3, type=2+3, level=method, maxDistance=1, minSize=40	*NiCad*, rename=blind, abstract=literal, threshold=0.20, minsize=10	101
CloneWorks, rename-blind, abstract literal, min-similarity 0.90, min-lines 20, *FilterDistance*	*DrDupLex3*, type=2+3, level=method, maxDistance=1, minSize=6, ignoreAnnotations=true	0
DrDupLex3, type=2+3, level=method, maxDistance=1, minSize=40	*CloneWorks*, rename-blind, abstract literal, min-similarity 0.90, min-lines 10	159
SourcererCC, MIN_TOKENS=40, threshold=9, *FilterLines*, *FilterDistance*	*DrDupLex3*, type=2+3, level=method, maxDistance=1, minSize=6, ignoreAnnotations=true	0
DrDupLex3, type=2+3, level=method, maxDistance=1, minSize=40	*SourcererCC*, MIN_TOKENS=40, threshold=8	8,435

Table 4. The clone pairs reported in BigCloneBench by *DrDupLex3* (type=2+3, level=method, maxDistance=1, minSize=40) and missed by *NiCad* (rename=blind, abstract=literal, threshold=0.20, minsize=10), *CloneWorks* (rename-blind, abstract literal, min-similarity 0.80, min-lines 10), and *SourcererCC* (MIN_TOKENS=40, threshold=8) (see Table 3) and their classification by *CloneChecker*. The values in the columns "Recall" and "Type-3 Recall" are relative recalls calculated as the ratio of clones reported by the tool to clones reported by *DrDupLex3* (it reported 67,688 clones: 61,340 Type-1 clones, 3,379 Type-2 clones, and 2,969 Type-3 clones).

Tool	Missed	Type 1	Type 2	Type 3	Recall	Type-3 recall
NiCad	101	88	11	2	99.85%	99.93%
CloneWorks	159	119	37	3	99.77%	99.90%
SourcererCC	8,435	5,623	133	2,679	87.54%	9.77%

rename-blind, abstract literal, min-similarity 0.90, and min-lines 10, subtracted the output from the output of *DrDupLex3* with parameters type=2+3, level=method, maxDistance=1, and minSize=40, added code fragments to the output XML file, and classified the clones.

Then I ran *SourcererCC* with parameters MIN_TOKENS=40 and threshold=9, filtered out clones shorter than 20 lines, added code fragments to the output XML file, computed distances between code fragments (this eliminated invalid clones), filtered out clones with a distance greater than 1, subtracted the output of *DrDupLex3* with parameters type=2+3, level=method, maxDistance=1, minSize=6, and ignoreAnnotations=true (because *SourcererCC* ignores annotations), and got an empty XML file. Then I ran *SourcererCC* with parameters MIN_TOKENS=40 and threshold=8, subtracted the output from the output of *DrDupLex3* with parameters type=2+3, level=method, maxDistance=1, and minSize=40, added code fragments to the output XML file, and classified the clones.

To compare the running time and consumed memory, I used a virtual machine with CPU Intel Xeon Silver 4310 at 2.10 GHz, 16 GB of memory, 64-bit Oracle Linux, and Oracle Java 17.0.9. I measured the running time and memory consumption by prefixing each command with "/usr/bin/time -v". Table 5 contains the elapsed (wall clock) time and the maximum resident set size as it was reported by the command ''/usr/bin/time -v''.

Table 5. Running time and peak memory usage of *NiCad* (rename=blind, abstract=literal, threshold=0.01, minsize=20), *CloneWorks* (rename-blind, abstract literal, min-similarity 0.99, min-lines 20), *SourcererCC* (MIN_TOKENS=75, threshold=9.9), and *DrDupLex3* (type=2+3, level=method, maxDistance=1, minSize=20) for processing BigCloneBench. The values in the columns "Time" and "Memory" are arithmetic averages of three measurements. The time is in minutes and seconds.

Tool	Reported clones	Time	Memory
NiCad	258,845	35:06	214 MB
CloneWorks	258,870	13:37	2,472 MB
SourcererCC	262,433	48:28	4,220 MB
DrDupLex3	267,653	10:12	1,669 MB

9. Threats to validity

The major threat to the validity of the presented evaluation is hidden in the configuration parameters. Since *NiCad* and *DrDupLex3* report different lines (*NiCad* reports pretty-printed lines and *DrDupLex3* reports real lines), *NiCad* can miss a clone because the code fragments have fewer than the minimum number of pretty-printed lines. To mitigate

this concern, I manually inspected all Type-3 clones missed by *NiCad* and verified that all of them should have been reported.

Another threat is the limited set of clone detectors used in evaluation. Although *NiCad*, *CloneWorks*, and *SourcererCC* represent state-of-the-art clone detectors, a comprehensive evaluation should involve more than three tools. Unfortunately, it turned out that out of eleven considered clone detectors, only five can be readily used, and out of these five, one tool is unable to process BigCloneBench, and one tool detects only a specific subset of Type-3 clones.

10. Limitations

The presented approach has three key limitations: (i) It requires a robust parser to identify syntactic units; (ii) The source code must be syntactically correct so that it can be parsed; (iii) It computes the edit distance for each pair of code fragments individually, which may be slow for large projects and big distances. An efficient algorithm for computing the edit distances in a set of strings would be preferable, but unfortunately, no such algorithm is available in the literature to date. Since computing edit distances can be parallelized, *DrDupLex3* employs multiple threads.

The algorithm uses the edit distance to measure code similarity. For a different code similarity, the algorithm may become less efficient. For example, for the normalized edit distance, i.e., the edit distance divided by the total number of tokens, the similarity needs to be computed for all candidate pairs because no pairs can be eliminated based on the difference in token counts. The trie is still useful, though, because it identifies Type-2 clones, which can be eliminated from candidate pairs.

The presented implementation detects clones only in the Java source code. While parsing of other programming languages is conceptually similar (a lexer and parser are typically employed), no other languages have been tried.

11. Related work

Over the past three decades, a substantial body of research has emerged in the area of clone detection. Numerous publications have described various techniques for identifying syntactic (Type-1, Type-2, and Type-3) and semantic (Type-4) clones; however, not all syntactic clone detection techniques are capable of detecting Type-3 clones.

Baxter et al. [12] described a method for detecting exact and near-miss clones that used abstract syntax trees and hashing. The similarity of code fragments was computed from the number of shared and different nodes in abstract syntax trees.

Kamiya, Kusumoto, and Inoue [10] presented *CCFinder*, which converted source code into sequences of tokens, concatenated these sequences, built a suffix tree, and used it to identify Type-2 clones. It does not support Type-3 clones, though.

Li et al. [11] presented *CP-Miner*, the tool that uses data mining techniques to efficiently identify copy-pasted code in a large code base. It converts source code into a sequence of tokens and searches for frequent subsequences.

Jiang et al. [39] presented an approach based on characteristic vectors in Euclidean space and its implementation for Java and C called *Deckard*. It parses source code, captures the syntactic information in numerical vectors, and clusters the vectors based on their

proximity. Unfortunately, the method does not scale well, and the tool is unable to process BigCloneBench.

Kraft, Bonds, and Smith [48] described an approach for detecting clones that span multiple languages and its implementation for the .NET Framework.

Roy and Cordy [40] presented *NiCad*, the accurate clone detector based on parsing and pretty printing. It formats source code, renames identifiers, and computes the longest common subsequence of sequences of lines.

Uddin et al. [49] researched the effectiveness of *simhash*, the fingerprint-based data similarity measurement technique, for clone detection.

Sajnani et al. [43] presented *SourcererCC*, the token-based clone detection tool, which uses an optimized inverted index and filtering heuristics to achieve exceptional scalability.

Svajlenko and Roy [37] presented *CloneWorks*, the fast and scalable clone detector, particularly suitable for large-scale near-miss clone detection. Unfortunately, it does not report the similarity of clones.

Saini et al. [50] focused on Type-3 clones with a similarity of less than 70% and presented *Oreo*, the tool built on machine learning, information retrieval, and software metrics, which can detect these clones. However, it is unclear how similarity is computed.

Wang et al. [36] presented *CCAligner*, the tool that uses a sliding window as a basic unit of matching and demonstrated that it was the best-performing large-gap clone detection tool at that time. Unfortunately, the project provided on GitHub does not work.

Ragkhitwetsagul and Krinke [42] introduced a novel clone detection technique that incorporates multiple code representations, query reduction, and a customized ranking function. They implemented the technique in a tool called *Siamese* and demonstrated that the tool scales to hundreds of millions of code lines. The implementation builds on *Elasticsearch*, the open-source distributed full-text search engine. Unfortunately, the project provided on GitHub does not work. I contacted the authors, but they were not able to help me.

Nakagawa, Higo, and Kusumoto [41] presented *NIL*, the clone detector that uses n-grams and an inverted index. It is particularly suitable for large-variance clones, which arise by inserting or deleting many statements in a copied fragment of code.

Schäfer, Amme, and Heinze [51] described a tool called *Stubber*, which compiles source code into bytecode without dependencies. It can be used for clone detection on the bytecode level.

Amme, Heinze, and Schäfer [44] described a new approach to clone detection based on control flow analysis and dominator trees and presented its implementation named *StoneDetector*. It parses source code and builds abstract syntax trees, transforms them into control flow graphs and dominator trees, encodes the dominator trees' paths into textual representations, and computes the similarity of these textual representations. The output is a sequence of lines, one line for each reported clone. The line consists of eight values: folder 1, file name 1, start line 1, end line 1, folder 2, file name 2, start line 2, and end line 2. Unfortunately, the folder is not a full path to the file, but only the parent folder, which is not sufficient for identifying the file.

Wang et al. [45] compared 12 clone detection algorithms and proposed a new tool called *TACC*, which is reportedly superior to *CCAligner*, *SourcererCC*, *NiCad*, *NIL*, and *Siamese*. Unfortunately, the source code in its GitHub repository does not compile. In addition, it seems that the output of the detector is just pairs of method identifiers with no additional information, which is insufficient for identifying code fragments.

Hu et al. [46] introduced *Tamer*, a tree-based clone detector that reportedly outperforms ten state-of-the-art clone detectors. Unfortunately, its GitHub repository contains only three Java files, and it looks more like a proof of concept than a real clone detector. In

addition, the sample output in the repository does not provide enough information to identify code fragments.

Wang et al. [52] presented *CCStokener*, a fast and accurate clone detection tool with the semantic token, and described an experiment in which *CCStokener* achieved the best recall in detecting moderate Type-3 clones. Unfortunately, its output does not provide enough information to identify code fragments.

Hu et al. [38] presented *Code2Img*, a tree-based code clone detector, which converts the abstract syntax trees into images, transforms each image into a one-dimensional vector, and calculates the Jaccard similarity of these vectors. The authors claim that *Code2Img* outperforms eight state-of-the-art clone detectors; however, the result is difficult to verify because the tool output consists of text files that contain only integers, which is insufficient for identifying code fragments.

Lavoie and Merlo [53] described an original method for constructing clone oracles based on the Levenshtein metric. It would be interesting to check their oracles with *CloneChecker*, but unfortunately, they do not provide it for download.

Keivanloo, Zhang, and Zou [33] proposed a threshold-free method for Type-3 clone classification. The method is based on K-means clustering.

White et al. [54] presented a deep learning-based approach to clone classification that can classify all types of clones.

Li et al. [55] described *CCLearner*, a neural-network classifier, which uses supervised learning, and trained it on a part of BigCloneBench.

Gao et al. [56] described a tree embedding technique for clone classification and its implementation called *TECCD*. The advantage of this technique is that it does not require a training set.

Several survey and comparison papers provide an overview of the state-of-the-art tools for clone detection at the time of their publication.

Rattan, Bhatia, and Singh [4] published a systematic literature review based on 213 papers selected out of 2039 papers published in 11 journals and 37 conferences and workshops.

Ragkhitwetsagul, Krinke, and Clark [57] evaluated 30 code similarity detection techniques and tools using five experimental scenarios for Java source code.

Walker, Cerny, and Song [58] published a study on open-source clone detection tools and benchmarks.

Zhang and Sakurai [26] provided a survey of clone detection from a security perspective. They describe applications of code clones in vulnerable code detection.

Zakeri-Nasrabadi et al. [59] published a systematic literature review on code similarity measurement and evaluation techniques. They investigated 80 tools working mainly on Java, C, and C++.

Kaur and Rattan [18] wrote a systematic review on using machine learning in clone classification.

12. Conclusion

The article described a new algorithm for Type-3 clone detection and its open-source implementation called *DrDupLex3*. The evaluation revealed that six out of eleven state-of-the-art clone detectors (*CCAligner*, *Code2Img*, *StoneDetector*, *Tamer*, *Siamese*, *TACC*) cannot be readily used, and one (*Deckard*) is not able to process BigCloneBench. The comparison of

DrDupLex3 with *NiCad*, *CloneWorks*, and *SourcererCC* demonstrated that *DrDupLex3* detects more clones than these tools. *DrDupLex3* builds on the index structure used in *DrDupLex*, the most accurate Type-2 clone detector to date. When the index structure is modified upon a change in source code, only the distances of modified pairs need to be computed, which makes the presented approach suitable for detecting clones "online". Furthermore, the article described the tools for automated evaluation of Type-3 clone detectors, which can facilitate the preparation of training sets for machine learning-based clone classifiers. For example, the tools can prepare a training set of clones and nonclones.

CRediT authorship contribution statement

Zdenek Tronicek: Conceptualization, Methodology, Software, Validation, Investigation, Writing – Original Draft Preparation, Writing – Review & Editing.

Declaration of competing interest

The author declares no competing interests.

Funding

The research has no funding sources.

References

[1] C.K. Roy and J.R. Cordy, "A survey on software clone detection research," *Queen's School of Computing TR*, Vol. 541, No. 115, 2007, pp. 64–68.

[2] S. Bellon, R. Koschke, G. Antoniol, J. Krinke, and E. Merlo, "Comparison and evaluation of clone detection tools," *IEEE Transactions on Software Engineering*, Vol. 33, No. 9, 2007, pp. 577–591.

[3] C.K. Roy, J.R. Cordy, and R. Koschke, "Comparison and evaluation of code clone detection techniques and tools: A qualitative approach," *Science of Computer Programming*, Vol. 74, No. 7, 2009, pp. 470–495.

[4] D. Rattan, R. Bhatia, and M. Singh, "Software clone detection: A systematic review," *Information and Software Technology*, Vol. 55, No. 7, 2013, pp. 1165–1199.

[5] A. Sheneamer and J. Kalita, "A survey of software clone detection techniques," *International Journal of Computer Applications*, Vol. 137, No. 10, 2016, pp. 1–21.

[6] Q.U. Ain, W.H. Butt, M.W. Anwar, F. Azam, and B. Maqbool, "A systematic review on code clone detection," *IEEE Access*, Vol. 7, 2019, pp. 86 121–86 144.

[7] G. Shobha, A. Rana, V. Kansal, and S. Tanwar, "Code clone detection – A systematic review," *Emerging Technologies in Data Mining and Information Security: Proceedings of IEMIS 2020, Volume 2*, 2021, pp. 645–655.

[8] B.S. Baker, "Parameterized duplication in strings: Algorithms and an application to software maintenance," *SIAM Journal on Computing*, Vol. 26, No. 5, 1997, pp. 1343–1362.

[9] J.R. Cordy and C.K. Roy, "The NiCad clone detector," in *19th International Conference on Program Comprehension*. IEEE, 2011, pp. 219–220.

[10] T. Kamiya, S. Kusumoto, and K. Inoue, "CCFinder: A multilinguistic token-based code clone detection system for large scale source code," *IEEE Transactions on Software Engineering*, Vol. 28, No. 7, 2002, pp. 654–670.
[11] Z. Li, S. Lu, S. Myagmar, and Y. Zhou, "CP-Miner: Finding copy-paste and related bugs in large-scale software code," *IEEE Transactions on Software Engineering*, Vol. 32, No. 3, 2006, pp. 176–192.
[12] I.D. Baxter, A. Yahin, L. Moura, M. Sant'Anna, and L. Bier, "Clone detection using abstract syntax trees," in *Proceedings of the International Conference on Software Maintenance*, 1998, pp. 368–377.
[13] J. Krinke, "Identifying similar code with program dependence graphs," in *Proceedings Eighth Working Conference on Reverse Engineering*, 2001, pp. 301–309.
[14] R. Komondoor and S. Horwitz, "Using slicing to identify duplication in source code," in *International Static Analysis Symposium*, 2001, pp. 40–56.
[15] C. Liu, C. Chen, J. Han, and P.S. Yu, "GPLAG: Detection of software plagiarism by program dependence graph analysis," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 872–881.
[16] J. Mayrand, C. Leblanc, and E. Merlo, "Experiment on the automatic detection of function clones in a software system using metrics," in *Proceedings of International Conference on Software Maintenance*, Vol. 96, 1996, p. 244.
[17] E. Kodhai and S. Kanmani, "Method-level code clone detection through LWH (Light Weight Hybrid) approach," *Journal of Software Engineering Research and Development*, Vol. 2, 2014, pp. 1–29.
[18] M. Kaur and D. Rattan, "A systematic literature review on the use of machine learning in code clone research," *Computer Science Review*, Vol. 47, 2023, p. 100528.
[19] W. Wang and M.W. Godfrey, "Recommending clones for refactoring using design, context, and history," in *International Conference on Software Maintenance and Evolution*. IEEE, 2014, pp. 331–340.
[20] N. Tsantalis, D. Mazinanian, and G.P. Krishnan, "Assessing the refactorability of software clones," *IEEE Transactions on Software Engineering*, Vol. 41, No. 11, 2015, pp. 1055–1090.
[21] E. Juergens, F. Deissenboeck, B. Hummel, and S. Wagner, "Do code clones matter?" in *31st International Conference on Software Engineering*, 2009, pp. 485–495.
[22] J. Jang, A. Agrawal, and D. Brumley, "ReDeBug: Finding unpatched code clones in entire OS distributions," in *Symposium on Security and Privacy*. IEEE, 2012, pp. 48–62.
[23] H. Li, H. Kwon, J. Kwon, and H. Lee, "CLORIFI: Software vulnerability discovery using code clone verification," *Concurrency and Computation: Practice and Experience*, Vol. 28, No. 6, 2016, pp. 1900–1917.
[24] S. Kim, S. Woo, H. Lee, and H. Oh, "Vuddy: A scalable approach for vulnerable code clone discovery," in *Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 595–614.
[25] B. Bowman and H.H. Huang, "VGRAPH: A robust vulnerable code clone detection system using code property triplets," in *European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2020, pp. 53–69.
[26] H. Zhang and K. Sakurai, "A survey of software clone detection from security perspective," *IEEE Access*, Vol. 9, 2021, pp. 48 157–48 173.
[27] N. He, L. Wu, H. Wang, Y. Guo, and X. Jiang, "Characterizing code clones in the Ethereum smart contract ecosystem," in *Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers 24*. Springer, 2020, pp. 654–675.
[28] Q. Hum, W.J. Tan, S.Y. Tey, L. Lenus, I. Homoliak et al., "Coinwatch: A clone-based approach for detecting vulnerabilities in cryptocurrencies," in *International Conference on Blockchain (Blockchain)*. IEEE, 2020, pp. 17–25.
[29] J. Crussell, C. Gibler, and H. Chen, "Attack of the clones: Detecting cloned applications on Android markets," in *Computer Security–ESORICS 2012: 17th European Symposium on Research in Computer Security, Pisa, Italy, September 10-12, 2012. Proceedings 17*. Springer, 2012, pp. 37–54.

[30] K. Chen, P. Liu, and Y. Zhang, "Achieving accuracy and scalability simultaneously in detecting application clones on Android markets," in *Proceedings of the 36th International Conference on Software Engineering*, 2014, pp. 175–186.
[31] J. Akram, Z. Shi, M. Mumtaz, and P. Luo, "Droidcc: A scalable clone detection approach for Android applications to detect similarity at source code level," in *42nd Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 1. IEEE, 2018, pp. 100–105.
[32] M.R. Farhadi, B.C. Fung, P. Charland, and M. Debbabi, "Binclone: Detecting code clones in malware," in *Eighth International Conference on Software Security and Reliability (SERE)*. IEEE, 2014, pp. 78–87.
[33] I. Keivanloo, F. Zhang, and Y. Zou, "Threshold-free code clone detection for a large-scale heterogeneous Java repository," in *22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. IEEE, 2015, pp. 201–210.
[34] Z. Tronicek, "Indexing source code and clone detection," *Information and Software Technology*, Vol. 144, 2022, p. 106805.
[35] R.A. Wagner and M.J. Fischer, "The string-to-string correction problem," *Journal of the ACM*, Vol. 21, No. 1, 1974, pp. 168–173.
[36] P. Wang, J. Svajlenko, Y. Wu, Y. Xu, and C.K. Roy, "CCAligner: A token based large-gap clone detector," in *Proceedings of the 40th International Conference on Software Engineering*, 2018, pp. 1066–1077.
[37] J. Svajlenko and C.K. Roy, "Fast and flexible large-scale clone detection with CloneWorks," in *IEEE/ACM 39th International Conference on Software Engineering (Companion Volume)*, 2017, pp. 27–30.
[38] Y. Hu, Y. Fang, Y. Sun, Y. Jia, Y. Wu et al., "Code2Img: Tree-based image transformation for scalable code clone detection," *IEEE Transactions on Software Engineering*, 2023.
[39] L. Jiang, G. Misherghi, Z. Su, and S. Glondu, "Deckard: Scalable and accurate tree-based detection of code clones," in *29th International Conference on Software Engineering*, 2007, pp. 96–105.
[40] C.K. Roy and J.R. Cordy, "NICAD: Accurate detection of near-miss intentional clones using flexible pretty-printing and code normalization," in *16th IEEE International Conference on Program Comprehension*, 2008, pp. 172–181.
[41] T. Nakagawa, Y. Higo, and S. Kusumoto, "NIL: Large-scale detection of large-variance clones," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2021, pp. 830–841.
[42] C. Ragkhitwetsagul and J. Krinke, "Siamese: Scalable and incremental code clone search via multiple code representations," *Empirical Software Engineering*, Vol. 24, No. 4, 2019, pp. 2236–2284.
[43] H. Sajnani, V. Saini, J. Svajlenko, C.K. Roy, and C.V. Lopes, "SourcererCC: Scaling code clone detection to big-code," in *Proceedings of the 38th International Conference on Software Engineering*, 2016, pp. 1157–1168.
[44] W. Amme, T.S. Heinze, and A. Schäfer, "You look so different: Finding structural clones and subclones in Java source code," in *IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2021, pp. 70–80.
[45] Y. Wang, Y. Ye, Y. Wu, W. Zhang, Y. Xue et al., "Comparison and evaluation of clone detection techniques with different code representations," in *IEEE/ACM 45th International Conference on Software Engineering*. IEEE, 2023, pp. 332–344.
[46] T. Hu, Z. Xu, Y. Fang, Y. Wu, B. Yuan et al., "Fine-grained code clone detection with block-based splitting of abstract syntax tree," in *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2023, pp. 89–100.
[47] J. Svajlenko and C.K. Roy, "Evaluating clone detection tools with BigCloneBench," in *International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2015, pp. 131–140.
[48] N.A. Kraft, B.W. Bonds, and R.K. Smith, "Cross-language clone detection," in *20th International Conference on Software Engineering & Knowledge Engineering*, 2008, pp. 54–59.

[49] M.S. Uddin, C.K. Roy, K.A. Schneider, and A. Hindle, "On the effectiveness of Simhash for detecting near-miss clones in large scale software systems," in *18th Working Conference on Reverse Engineering*. IEEE, 2011, pp. 13–22.
[50] V. Saini, F. Farmahinifarahani, Y. Lu, P. Baldi, and C.V. Lopes, "Oreo: Detection of clones in the twilight zone," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2018, pp. 354–365.
[51] A. Schäfer, W. Amme, and T.S. Heinze, "Stubber: Compiling source code into bytecode without dependencies for Java code clone detection," in *15th International Workshop on Software Clones (IWSC)*. IEEE, 2021, pp. 29–35.
[52] W. Wang, Z. Deng, Y. Xue, and Y. Xu, "CCStokener: Fast yet accurate code clone detection with semantic token," *Journal of Systems and Software*, Vol. 199, 2023, p. 111618.
[53] T. Lavoie and E. Merlo, "Automated type-3 clone oracle using Levenshtein metric," in *Proceedings of the 5th International Workshop on Software Clones*, 2011, pp. 34–40.
[54] M. White, M. Tufano, C. Vendome, and D. Poshyvanyk, "Deep learning code fragments for code clone detection," in *Proceedings of the 31st IEEE/ACM international conference on automated software engineering*, 2016, pp. 87–98.
[55] L. Li, H. Feng, W. Zhuang, N. Meng, and B. Ryder, "CCLearner: A deep learning-based clone detection approach," in *International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2017, pp. 249–260.
[56] Y. Gao, Z. Wang, S. Liu, L. Yang, W. Sang et al., "TECCD: A tree embedding approach for code clone detection," in *International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2019, pp. 145–156.
[57] C. Ragkhitwetsagul, J. Krinke, and D. Clark, "A comparison of code similarity analysers," *Empirical Software Engineering*, Vol. 23, 2018, pp. 2464–2519.
[58] A. Walker, T. Cerny, and E. Song, "Open-source tools and benchmarks for code-clone detection: past, present, and future trends," *ACM SIGAPP Applied Computing Review*, Vol. 19, No. 4, 2020, pp. 28–39.
[59] M. Zakeri-Nasrabadi, S. Parsa, M. Ramezani, C. Roy, and M. Ekhtiarzadeh, "A systematic literature review on source code similarity measurement and clone detection: Techniques, applications, and challenges," *Journal of Systems and Software*, 2023, p. 111796.

Authors and affiliations

Zdenek Tronicek
e-mail: zdenek.tronicek@oneonta.edu
ORCID: https://orcid.org/0000-0002-2737-1317
State University of New York, Oneonta, NY, USA

e-Informatica Software Engineering Journal
Volume 20, Issue 1, pages: 1–38
DOI: 10.37190/e-Inf260106
BibTeX

From Draft to Standard: An Expert-Validated Catalog of API Gateway Software Metrics

Eder dos Santos*, Sandra Casas
*Corresponding author: esantos@uarg.unpa.edu.ar

Article info

Dataset link: http://hdl.handle.net/11336/264879

Keywords:
software quality
software measurement and metrics
internet software systems development
empirical and experimental studies in software engineering

Submitted: 17 Feb. 2026
Revised: 6 May 2026
Accepted: 6 May 2026
Available online: 12 May 2026

Abstract

Context: In the fast-growing API Economy scenario, API Gateways became an essential tool for microservice reliability and scalability, yet lack standardized quality models and measures to assess these tools in an objective fashion.
Objective: This study quantitatively and qualitatively assessed the accuracy and relevance of a set of 59 metrics, gathered from prior analysis of 68 mainstream API Gateway software offerings.
Method: An expert judgment evaluation study was conducted with seven domain specialists ($N = 7$) from both academia and industry, who assessed each metric's accuracy and relevance using 5-point Likert scales. A multi-statistical method was performed to assess ratings, inter-observer consistency and reliability.
Results: Although ratings were generally neutral to positive, statistical analysis revealed low consensus across experts. A subset of 24 core metrics with strong agreement and high scores was identified, alongside moderate- and low-agreement sets. These items cover key API management concerns, including traffic monitoring, error handling, latency, and resource usage.
Conclusions: By combining practical definitions, expert judgment, and a tiered classification mechanism, this work established a broad and comprehensive set of metrics that will help to bridge the gap between metric adoption in real-world platforms and their theoretical grounding in software quality models.

1. Introduction

Problem statement

As organizations increasingly adopted microservices architectures in their software ecosystems, the role of API Gateways became pivotal in ensuring system performance, scalability, reliability, security and maintainability [1–3].

While existing literature addresses various aspects of API Management software quality such as performance, security, and usability [4,5], there is a notable absence of standardized, validated frameworks to assess the quality of API Gateways [6]. This gap impedes the development of consistent evaluation methodologies and benchmarks for API Gateway implementations.

© 2026 The Authors. Published by Wrocław University of Science and Technology Publishing House.
This is an open access article under the CC BY-SA 4.0 license international.

Study context

In contemporary software engineering, Application Programming Interfaces (API) serve as fundamental building blocks, enabling interoperability and integration across diverse systems and platforms [1, 7]. The proliferation of digital services has underscored the necessity for effective API management, with the API Gateway emerging as a critical component in orchestrating and securing API traffic [5].

The growing adoption of APIs has led to the proliferation of various API Gateway products. Current industry "state of the market" reports indicate that there are 65+ API Gateway offerings available nowadays[1,2,3].

In such a competitive landscape, it is essential to understand how software quality can be defined, measured and assessed, and what these products provide to satisfy the stakeholders' requirements. However, there is a fragmented research landscape characterized by technical implementations often lacking grounding in established software quality models, and a notable absence of standardized, validated frameworks for API Gateway quality assessment [6].

To address this gap comprehensively, our research has pursued a multi-phase strategy integrating multiple evidence sources:

1. Understanding the research landscape [6]: identified quality attributes, measurement and evaluation methods, and the absence of formal frameworks in API management research.
2. Understanding practitioners' priorities [3]: a practitioner-oriented survey explored how professionals perceive API management quality requirements and mapped these to ISO/IEC 25010 quality attributes [8], revealing that functional suitability, performance efficiency, reliability, and security are consistently prioritized.
3. Capturing operational reality [9]: the analysis of 68 mainstream API Gateway platforms documented the operational metrics actually deployed in production systems.
4. Expert validation (current study): Rigorous evaluation of practice-based metrics by domain experts to establish a validated, standardized foundation.

Research objective

The main goal of this work was to quantitatively and qualitatively validate the accuracy and relevance of a preliminary set of 59 API Gateway software metrics gathered from a previous analysis of current mainstream API Gateway software offerings, as detailed in [9]. The validation was carried out using an expert judgment approach, following the guidelines proposed in [10, 11].

Contribution statement

This study contributes to the software engineering and API management literature by providing a validated and empirically grounded set of API Gateway quality metrics. Main contributions include:

– The empirical evaluation of accuracy and relevance of a large set of API Gateway software metrics, performed by a cohort of seven internationally recognized specialists from both academia and industry.

[1] https://www.gartner.com/en/documents/4829031
[2] https://www.postman.com/state-of-api/
[3] https://apilandscape.apiscene.io/

– A structured statistical analysis of the agreement level among specialists, based on inter-observer reliability and internal consistency, according to [12].
– A tier-based classification scheme based on expert ratings and statistical agreement between the specialists.
– A subset of 24 core metrics with strong agreement and high scores that can support standardized quality assessment in the API management domain, which can be establish a ready-to-use artifact to measure, assess and improve API Gateway software quality.

By combining systematic artifact analysis, expert validation, and quantitative rigor, this research effort addresses the absence of standardized quality measures for API Gateways and offers a robust foundation for benchmarking, product evaluation, and integration with established software quality frameworks. Ultimately, it represents a significant step toward defining a tailored API Gateway software product quality model, serving as a foundation for more consistent assessment, informed decision-making, and future standardization efforts in the domain.

Research significance

This work addresses a critical gap in API management research and practice at a pivotal moment in the field's evolution. The significance of this work operates on multiple levels:

Theoretical contribution: By validating operational metrics through expert judgment and grounding them in established software quality frameworks, this research bridges the gap between ad-hoc industry practices and formal quality assessment methodologies. The resulting catalog provides the empirical foundation necessary for developing a comprehensive API Gateway quality model aligned with existing standards such as [13].

Practical impact: The 24 validated core metrics offer immediately deployable quality indicators for organizations managing API infrastructure. Unlike existing platform-specific metrics, this validated set provides a vendor-neutral, expert-endorsed foundation for benchmarking, monitoring, and evaluating API Gateway implementations across diverse organizational contexts.

Methodological implication: The four-step validation strategy – integrating systematic literature analysis, practitioner survey data, operational platform analysis, and expert judgment – establishes a rigorous methodological template for validating practice-derived software metrics. This approach demonstrates how industry practices can be systematically elevated to theoretically grounded standards.

Timing and context: With 65+ API Gateway products currently available and the API Economy projected to grow substantially, the absence of standardized quality assessment frameworks represents a significant impediment to informed decision-making, systematic improvement, and vendor comparison. This work proposes a validated measurement foundation essential for the field's maturation from fragmented implementations to standardized best practices.

1.1. Paper structure

The remainder of this paper is organized as follows. Section 2 explains the background knowledge. Section 3 reviews related work on API management and API Gateway quality, providing context for the research. Section 4 outlines the research methodology, detailing the sampling strategy, survey design, and multi-statistical approach used to validate expert ratings. In Section 5, we present the key findings from expert ratings and statistical analysis,

and include a tier-based classification of API Gateway measures. Section 6 discusses the implications of these findings, while Section 7 addresses the potential threats to the validity of the study. Finally, Section 8 concludes the paper and offers directions for future research.

2. Background

2.1. API, API management and API Gateways

In recent years, the distribution models for information systems have been moving towards XaaS [14] paradigms. This shift entailed organizations providing their digital assets as services to customers [15]. In general, such services are supported by API that typically adhere to REST [16] principles.

API present both a business side and a technical side [17]: they determine how the organizations want to use their assets to deliver value within the organization and to external third parties, and they are "a technical answer to a business problem" that allow to implement a set of requirements that govern how applications can interact and exchange data. In this sense, the term "API Economy" [18] reflects the ongoing trend across industries towards innovative approaches to enhancing their business frameworks.

The API Economy scenario exerts additional pressure in developing, deploying, and maintaining information systems. Since APIs have gained a critical aspect [19], organizations need to proactively address the risks of failure by enhancing API Management [5, 17] capabilities and managing their APIs through API Management platforms.

API Management Platforms provide the basic capabilities to create, analyze, and manage APIs in a secure and scalable environment such as providing helpful documentation, controlling access to the API, as well as monitoring and analyzing its usage [5]. At the core of digital integration strategies, these platforms facilitate the creation, analysis, and governance of APIs within a secure and flexible framework, streamlining API management processes [1, 5, 20].

At the core of API Management products lies the API Gateway. The API Gateway acts as a centrally managed, API-oriented control service that encapsulates, exposes, secures, and manages back-end data and services as RESTful APIs, which provides a framework for establishing a facade on top of these services [21]. API Gateway's main features may include security, traffic management, service and data mapping, caching, load balancing, error reporting, and performance monitoring, among others. API Gateways provide a broad set of metrics that enables the production of business-value reports and user-friendly dashboards aiming at helping to understand who uses available APIs and how they are utilized [5].

2.2. API management capabilities and practices

In general, API Management Platform features are generally characterized in the form of practices and capabilities.

According to [5], API Management capabilities represent the features that enables developers to create, analyze, audit, log, analyze, and manage the lifecycle of API in a secure, reliable and flexible environment. Suggested capabilities are: developer enablement for API; API lifecycle management; secure, reliable and flexible communications; and API auditing, logging and analytics.

Additionally, a systematic mapping study (SMS) by [22] identified 114 API Management practices and 39 capabilities. In their framework, a practice was defined as "any activity with the explicit goal of improving, encouraging, or managing the usage of an API", while a capability referred to "the ability to achieve a specific API Management objective through the coordinated execution of two or more interrelated practices". Commonly reported capabilities included authentication, monitoring, security, analytics, catalog and documentation, API publication and deployment, monetization, and version management.

2.3. Software product quality and metrics

There is a wide range of standards applicable to various areas of knowledge in Software Engineering, with a common objective related to quality. In general, software quality can be appreciated from two perspectives: the first is based on the ISO 9001 standard [23], which focuses on quality policy across the entire organization. The terminology of this standard may be unfamiliar to software professionals, and quality management auditors may not be familiar with the software engineering vocabulary [10, 24]; the second seeks to decompose software quality into a set of desired characteristics of a software product and select measures and evaluation methods to determine if the desired level of those characteristics has been achieved. From this perspective, over time, various quality models have been developed, such as FURPS, Boehm, Dromey, and McCall, among others [25].

The terminology for quality attributes of software products differs from one model to another; each model has a different number of hierarchical levels, as well as a distinct number of characteristics [25]. A milestone in the definition of quality standards was the publication of the ISO/IEC 9126:1991 standard [26], which was subsequently replaced by the ISO/IEC 9126:2001 [27] and ISO/IEC 14598:1999 [28] standards. Currently, the international standards series ISO/IEC 25000 [29], known as SQuaRE (Software Product Quality Requirements and Evaluation), unifies and replaces the ISO/IEC 9126 and ISO/IEC 14598 series. SQuaRE is considered the current de facto standard series for software product quality, addressing aspects related to quality management, requirements, software and data quality models, as well as measurement and evaluation processes for quality.

The ISO/IEC SQuaRE series is structured into several divisions. Within the scope of this paper, the relevant divisions are outlined as follows:

The Quality Model Division (2501n) provides models for system and software product quality, quality in use, and data quality. The general software product quality model is described in [13].

The Quality Measurement Division (2502n) includes a system and software product quality measurement reference model, definitions of quality measures, and practical guidance for their implementation. The framework for quality measurement is illustrated in [30].

The Quality Evaluation Division (2504n) offers requirements, recommendations, and guidelines for system and software product evaluation. An overview of the evaluation process is presented in [31].

The concept of measurement was introduced by [32] as "the assignment of numerals to objects or events according to rules". This foundational definition highlights that measurement is not merely about assigning numbers, but doing so systematically, in a way that reflects the nature of the attributes being observed. The scale of measurement used – such as nominal, ordinal, interval, or ratio – determines the types of analysis or comparisons that are meaningful.

In software engineering, measurement is essential for understanding, evaluating, and improving software products and processes [33]. Accordingly, metrics are proposed to quantify specific quality attributes and support decision-making.

The representational theory of measurement formalizes the relationship between observed attributes and the numerical values used to represent them. As noted by several authors [12, 33, 34], all measurement begins with the observer's intuition and subjective interpretation, which must then be systematically refined into reliable, objective measures.

2.4. Expert judgment in software engineering

In software engineering, judgment studies leverage structured expert evaluations to assess constructs that are difficult to measure directly, by gathering empirical data from a group of participants who are asked to judge or rate behaviors, to discuss a given topic of interest. As stated by [10], "the Judgment Study is a compromise between achieving a high level of precision and generalizability of findings".

Judgment studies rely on systematic sampling and involve experts, appropriately informed to respond to a certain question. These studies commonly employ formal elicitation methods – like Delphi [35] panels, focus groups [36], or evaluation studies [10] – combined with inter-rater agreement metrics [12] to ensure reliability and validation of subjective assessments.

According to [33], validating a software metric is the process of ensuring that the measure is a proper characterization of a claimed attribute by showing that its representation condition is satisfied.

From a theoretical standpoint, the accuracy of a measure depends on the definition of the measurement, as stated by [33]. However, measures may not be likely to be accurate, either because the definition is imprecise or ambiguous, or because it depends on the subjective judgment of the person doing the measuring.

3. Related work

The quality of API management has been increasingly examined from both academic and industry perspectives, yet it remains loosely defined and lacking formal standardization. This section critically examines existing research across four dimensions: the API management quality research landscape, quality modeling approaches, validation methodologies in the API domain, and the specific metric validation gap that motivates this study.

3.1. API management quality research landscape

A recent SMS [6] identified a fragmented landscape of research problems and evaluation methods in the API management domain. The study analyzed contributions across multiple venues and revealed that most work is categorized as technical implementations or specific solutions [37], often without grounding in established software quality models. Among the most commonly addressed quality attributes were performance efficiency, reliability, functional suitability, and security. The study also highlighted that research generally did not explicitly utilize formal quality models. However, it underscored that common meta-model elements, such as features and metrics, were widely employed as evaluation mechanisms.

This fragmentation reflects a broader pattern in software engineering research where, as noted by [37], solution-oriented contributions often provide limited generalizability without grounding in theoretical frameworks. The API management domain exemplifies this pattern – considerable implementation innovation exists, alongside sparse validation of underlying quality constructs. While researchers and practitioners recognize that performance, reliability, and security matter, the absence of standardized measurement frameworks prevents systematic comparison, benchmarking, or cumulative knowledge building. This gap between recognized importance and measurement standardization represents a critical impediment to the field's maturation.

3.2. Quality modeling approaches for APIs

Several researchers have attempted to formalize API quality assessment through structured models. [38] proposed an API management maturity model (API-m-FAMM) from an organizational perspective, establishing a set of focus areas for capability development across different maturity levels. The model provides a framework for organizations to assess and improve their API management practices systematically. [39] introduced a web API quality meta-model and defined a quality model with three attributes – functionality, reliability, and usability – demonstrating quantitative evaluation in real-world applications through a case study approach. More recently, [40] presented a catalog of metrics to assess API usability, developed using the Goal-Question-Metric (GQM) [41] approach, which provides a structured method for deriving metrics from quality goals.

These contributions advance quality conceptualization in important ways. The maturity model [38] offers organizational guidance for capability improvement, while the meta-model [39] demonstrates feasibility of quantitative quality assessment. The GQM-based usability metrics [40] provide rigorous derivation from explicit quality goals. However, these works share common limitations relevant to the current study's focus. First, the maturity model addresses organizational capabilities rather than product quality metrics – it guides *how* to manage APIs but not *what* to measure in API products. Second, the meta-model's three-attribute scope provides limited coverage of the comprehensive quality concerns identified in [6], particularly overlooking operational aspects like resource utilization and system health. Third, while the usability metrics are rigorously derived, they address only one quality characteristic. Critically, none of these works specifically target API Gateway quality as a distinct domain, nor do they validate metrics against operational implementations in production systems. The current study addresses these gaps by focusing specifically on API Gateway infrastructure and validating metrics already deployed in real-world platforms.

3.3. Validation methodologies in API research

The use of surveys and empirical studies has been employed to characterize API practices and validate artifacts in the domain. [42] presented the outcomes of the first survey specifically targeting API developers in Argentina, identifying usage patterns and challenges faced by software developers when consuming web APIs within that regional context. This work provided valuable insights into practitioner experiences and common pain points in API consumption.

Building on artifact validation approaches, [40] employed surveys directed to both API consumers and developers to evaluate the usefulness of their proposed usability metrics

set. In a related study, [43] conducted a survey with 36 practitioners – including both experts and non-experts – to assess perceptions on the influence of various usability factors. Interestingly, their findings indicated that ratings did not vary significantly between experts and non-experts in the context of usability evaluation.

The finding that expert and non-expert ratings converged in usability assessment [43] provides an important methodological insight that contrasts with the current study's design. This convergence likely reflects the intuitive nature of usability concerns – factors like ease of use and clarity are readily assessed by diverse users. However, evaluating technical quality metrics requires different expertise: assessing metric *accuracy* (whether definitions are precise and unambiguous) and operational *feasibility* (whether metrics can be implemented reliably) demands deep domain knowledge. Our deliberate focus on recognized experts thus reflects distinct validation objectives appropriate for technical metric evaluation rather than user experience assessment.

3.4. Practitioner perspectives and requirements

Complementing academic research, practitioner-oriented studies have explored quality requirements from the perspective of those implementing and operating API management systems. [3] conducted a survey exploring how professionals in the field perceive the quality requirements of API management systems and mapped these traits to quality attributes established in the ISO/IEC 25010 standard [8]. Functional suitability, performance efficiency, reliability, and security were consistently highlighted as the most critical characteristics – reinforcing the overlap between academic research focus and practitioner needs, while simultaneously exposing the absence of systematic quality frameworks for assessment.

The remarkable convergence between academic research emphasis [6] and practitioner priorities [3] on performance, reliability, security, and functional suitability provides strong evidence for these attributes' centrality to API Gateway quality. This alignment validates the relevance of academic research directions. However, this consensus also reveals a critical gap: while agreement exists on *what quality attributes matter*, no standardized, validated frameworks exist for systematically *measuring* these concerns in API Gateway contexts. Practitioners must either rely on vendor-specific metrics that vary across platforms or develop ad-hoc measurement approaches that lack empirical validation. This measurement gap motivated our previous work in cataloging operational metrics [9] and drives the validation study reported in this paper.

3.5. The metric validation gap

Despite growing recognition of API management quality's importance, as evidenced by the studies reviewed above, standardized and empirically validated metrics for API Gateway assessment remain conspicuously absent. Existing research has identified quality attributes that matter [6], characterized practitioner priorities [3], and proposed quality models [38,39], yet none have established validated metrics specifically for API Gateway infrastructure.

Our previous work [9] began addressing this gap by systematically cataloging 59 metrics from 68 commercial and open-source API Gateway platforms. The catalog was developed following Design Science Research (DSR) principles [44] and organized according to established API management capability frameworks [5,22]. Quantitative analysis revealed that most metrics addressed latency, response time, error tracking, and traffic monitoring – operational concerns aligned with the practitioner perspectives discussed in [3]. This

work documented operational reality by capturing what practitioners actually measure in production systems. However, the catalog lacked external validation, particularly through systematic expert evaluation.

As emphasized by measurement theory [33], validating a software metric requires demonstrating that it properly characterizes the claimed attribute – ensuring that the representation condition is satisfied. Practice-derived metrics, while demonstrating operational feasibility, require expert validation to establish theoretical grounding and confirm accurate representation of quality constructs. Without such validation, the risk remains that operational metrics may be measuring convenient proxies rather than genuine quality attributes, or that metric definitions may be ambiguous or inconsistent across implementations. This validation gap – the absence of systematic expert assessment of practice-derived API Gateway metrics – directly motivated the current study.

3.6. Positioning the current work

This study uniquely addresses the identified gaps by integrating multiple evidence sources and validation approaches. Our work differs from existing research in several critical dimensions:

Domain specificity: Unlike general API quality models that address APIs broadly, this work specifically targets API Gateway infrastructure – the critical component that orchestrates, secures, and monitors API traffic in microservices architectures. API Gateways present distinct quality concerns related to traffic routing, load balancing, security enforcement, and operational monitoring that differ from application-level API quality.

Practice-grounded validation: Rather than proposing theoretically derived metrics, we validate measures already deployed in 68 production systems [9], ensuring practical feasibility and operational relevance. This approach acknowledges that metrics demonstrating real-world adoption likely address genuine operational concerns, though they require validation to establish theoretical soundness.

Rigorous expert judgment methodology: We apply established validation methodologies from software engineering research [10, 12], including purposive expert sampling, structured evaluation instruments, and multi-method statistical analysis of inter-rater agreement. This rigor distinguishes our validation effort from informal consensus or anecdotal acceptance.

Multi-source triangulation: The current validation study represents the fourth phase of a comprehensive research program that integrated: (1) systematic mapping of API management research [6], (2) practitioner survey of quality requirements [3], (3) systematic analysis of operational metrics across platforms [9], and (4) expert validation (current study). This four-step approach provides robust triangulation across literature, practitioner perspectives, operational reality, and expert judgment.

Actionable classification framework: Beyond binary validation (accept/reject), we provide a tier-based classification that translates mixed expert assessments into practical guidance. This framework identifies metrics ready for immediate adoption (Tier 1), those requiring refinement (Tier 2), and those needing substantial revision (Tier 3), supporting evidence-based decision-making by researchers and practitioners.

By systematically bridging the gap between operational practice and theoretical validation, this research establishes the empirical foundation necessary for developing standardized API Gateway quality assessment frameworks. The validated metric set addresses the fragmentation identified in [6], operationalizes the quality requirements articulated in [3],

and elevates the practice-derived metrics documented in [9] to theoretically grounded, expert-validated standards.

4. Method

Research questions

The overarching goal of this study is to establish a validated foundation for API Gateway quality assessment by evaluating the accuracy and relevance of 59 practice-derived metrics through expert judgment. This validation serves multiple purposes: (1) establishing consensus on metric definitions, (2) identifying a core set of reliable measures, (3) mapping validated metrics to quality concerns, and (4) providing a classification framework to guide adoption and future standardization efforts. To operationalize this goal systematically, four complementary research questions that address different facets of the validation process were formulated.

RQ1: *To what extent do domain experts agree on the accuracy and relevance of existing API Gateway quality metrics?*

Understanding the level of agreement among experts helps determine the perceived validity and practical applicability of the proposed metrics. Low consensus may indicate ambiguity, redundancy, or contextual variability that needs to be addressed before broader adoption.

RQ2: *Which metrics demonstrate strong consensus and high scores among experts, and can thus be considered core indicators for API Gateway quality evaluation?*

Identifying a prioritized set of validated metrics allows the construction of a focused, reliable foundation for API Gateway quality assessment that aligns with expert expectations and practical concerns.

RQ3: *What categories of API management capabilities are mostly represented by the validated metrics?*

Mapping validated metrics to broader quality concerns clarifies coverage, highlights gaps, and informs the development of a structured quality model tailored to the specific needs of API Gateway evaluation.

RQ4: *How can API Gateway quality metrics be classified based on expert ratings to guide their adoption, review, or exclusion?*

A systematic classification framework is needed to translate expert judgments into actionable categories that support decision-making. By defining clear criteria for grouping metrics, this classification can prioritize metrics for inclusion in quality models, identify those requiring further review, and exclude unreliable or irrelevant measures.

Collectively, these four research questions address complementary aspects of metric validation: RQ1 establishes the reliability baseline through consensus measurement; RQ2 identifies the validated core suitable for adoption; RQ3 ensures domain coverage by mapping to established capability frameworks; and RQ4 provides a classification mechanism that translates empirical findings into practical guidance. Answering all four RQs enables us to achieve the overarching goal of establishing a robust, expert-validated foundation for API Gateway quality assessment.

4.1. A practice-oriented approach

The deliberate focus on metrics from deployed systems introduced in this work was informed by findings from our prior SMS [6]. That study found that: most contributions were categorized as technical implementations or specific solutions, often without grounding in established software quality models; research generally did not explicitly utilize formal quality models; and a notable absence of standardized, validated frameworks for API Gateway quality assessment existed. Given this landscape, a practice-first approach was methodologically motivated by:

1. Complementary evidence: The SMS [6] captured what researchers propose; [3] revealed the practitioners perspective of quality requirements; and the platform analysis [9] captured what practitioners implement. Together, these provide comprehensive coverage of the domain.
2. Bridging the research-practice gap: validating operational metrics already deployed ensures to create a pathway for integrating industry practice with academic quality frameworks, addressing the gaps identified in [6].
3. Ecological validity: Metrics implemented in production platforms have demonstrated feasibility, measurability, and perceived utility by platform developers and users across diverse organizational contexts.
4. Operationalizability: Real-world metrics come with implicit or explicit measurement procedures, reducing ambiguity about measurement implementation. It helps to mitigate definitional inconsistencies.

4.2. Origin and development of the preliminary metric set

The 59 API Gateway software metrics evaluated in this study were systematically derived through a comprehensive analysis of mainstream API Gateway platforms, as detailed in our prior work [9]. This subsection describes the methodological approach used to ensure the breadth and representativeness of the preliminary metric catalog.

The metric extraction process analyzed 68 API Gateway software offerings, selected to represent the current market landscape comprehensively. The platform sample included commercial enterprise solutions, open-source platforms, cloud-native gateways, and hybrid and specialized solutions across various deployment models.

Platform selection was guided by three industry reports that catalog active API management solutions: Gartner Magic Quadrant for API Management,[1] Postman State of API Report,[2] and Apidays global API Landscape[3]. This triangulated approach ensured coverage of both market leaders and emerging platforms, representing diverse architectural patterns, target audiences, and operational contexts.

4.2.1. Metrics extraction and definition

Following the DSR [44] framework, the metric extraction process consisted of four systematic phases.

1. Documentation review: For each platform, official documentation was systematically reviewed, including product documentation and technical specifications, monitoring and observability guides, dashboard and reporting interface descriptions, API reference documentation, and administrative console specifications.

2. Metric identification: Measures explicitly exposed, tracked, or reported by each platform were cataloged. This included metrics available through built-in monitoring dashboards, exported telemetry data, and logging and analytics features.
3. Standardization: Metrics with similar semantic meaning but different naming conventions across platforms were consolidated.
4. Categorization: Metrics were organized according to established API management capability frameworks [5, 22]. This categorization ensured that the metric set was grounded in recognized API management concerns and aligned with both academic research and industry practice.

For each identified metric, a formal definition was developed including: a metric code and a human-readable, standardized identifier; a clear, operational description of what the metric measures; and an associated category, derived from a set of seven capability-based categories (API performance, API utilization, caching, errors, latency and response time, resource utilization, system health, and traffic monitoring).

4.3. Sampling technique and data manipulation

This work employed a survey methodology [11, 12, 45] aimed at validating a preliminary catalog of 59 API Gateway software quality metrics [9] through expert judgment [10]. The survey followed an exploratory approach based on the classification proposed by [46], which focuses on using surveys to gather insights and assess the quality characteristics of software management tools and practices. The objective of the survey was to evaluate the accuracy and relevance of each proposed metric for assessing API Gateway quality, as perceived by a select group of experts in the field.

This evaluation study [10] applied a purposive judgmental sampling scheme, specifically expert sampling [47,48]. Expert sampling is a type of purposive sampling where participants are intentionally selected for their recognized expertise in the field. This technique ensures that the feedback is credible, informed by deep subject knowledge, and directly relevant to the study's objectives.

Sample size: Seven experts participated in this validation study ($N = 7$). This sample size aligns with established practices for expert judgment studies in software engineering [46, 47], where small, purposively selected panels of 5–10 specialists are typical, providing a manageable yet diverse set of opinions.

The selection aimed to balance theoretical perspectives with industry experience, ensuring that the feedback would be rigorous and applicable to both academic research and real-world practices. Experts were selected based on the following criteria: (1) recognized contributions to API management research or practice (e.g., peer-reviewed publications, book authorship, or platform development leadership), (2) minimum of three years of professional experience with API technologies, and (3) current active engagement in API-related work.

The survey was implemented using an online questionnaire platform, facilitating efficient data capture and ensuring ease of access for the participants. The platform allowed participants to submit responses and additional comments, providing both quantitative and qualitative data. Then, survey responses were exported as CSV files.

Data preprocessing, statistical computation, and visualization, including the generation of bar plots and distribution charts, were performed using RStudio (version 2025.05.1+513) [49] and R (version 4.5.0) [50], leveraging the `ggplot2` package (version 3.5.2) [51] for

plotting. Figures were incorporated to enhance interpretability of expert rating distributions and consensus levels.

4.4. Survey design

The survey instrument was designed to validate a catalog of 59 API Gateway quality metrics proposed in earlier work [9]. It consisted of four primary sections: introduction, demographic profiling, metric evaluation, and final feedback.

Introduction: As suggested by [12], this section provided context for the study, including an overview of the research group, a summary of the previous work on the preliminary catalog, and the objectives of the current validation effort. It also informed participants the estimated completion time (approximately 15–20 minutes), and the intended use of the responses in refining the catalog.

Demographic information: This section gathered background data on participants, including name and email (for contact if needed), current professional sphere, primary role, years of experience, exposure to API technologies, and involvement in API-related communities. This information supported the verification of participant expertise and relevance to the API management domain.

Metric assessment: The core of the survey consisted of an evaluation of 59 metrics, summarized in Appendix A. For each metric, respondents were shown the metric's name, definition, and category based on API management capabilities, as outlined in [5, 22]. Participants rated each metric using a five-point Likert scale (1 = Strongly Disagree (SD) to 5 = Strongly Agree (SA)) across two dimensions: (i) *Accuracy* – the extent to which the metric definition was understandable, and (ii) *Relevance* – the perceived importance of the metric for assessing API Gateway quality. Radio buttons were used to encourage intentional responses and minimize accidental selections.

The question format is illustrated in Table 1 and an example question is provided in Table 2.

Table 1. Survey question structure

Metric Descriptor: Text. Contains the metric code and metric name. Definition: Text. Brief definition of the metric. Category: Text. Based on API Management capability.					
	Ratings				
Criteria	SD	D	NAD	A	SA
Accuracy: this metric is well-defined. Relevance: this metric is relevant.					

The complete survey instrument, including all instructions, demographic questions, and the 59 metric evaluation items, was implemented using Google Forms. A PDF export of the complete survey showing all questions, definitions, rating scales, and instructions exactly as presented to participants is publicly available (see Data Availability section).

Final feedback: The survey concluded with an open-ended question soliciting general comments, suggestions, or observations from participants. This provided an opportunity for qualitative insights regarding the catalog content, clarity of definitions, coverage breadth, and any potential omissions.

Table 2. Example metric evaluation question

Metric Descriptor: **AU.1.1 Active Inbound Data** Definition: The total volume of inbound data currently being processed by the API Gateway. Reflects the real-time data received from clients. Category: API Utilization Metrics (AU).					
	SD	D	NAD	A	SA
Accuracy: this metric is well-defined.					
Relevance: this metric is relevant.					

In total, each participant was asked to complete 126 responses. The survey was made available from May 31 to June 21, 2025, providing a three-week period for participation. This timeframe was selected to offer respondents sufficient flexibility and time to complete the questionnaire at their convenience.

4.5. Statistical analysis of expert ratings

To assess the inter-reliability and internal consistency of expert evaluations, a multi-method statistical validation strategy was adopted, following the principles outlined in [12, 52], as detailed as follows.

Descriptive analysis and visualization: A preliminary analysis was performed to examine the distribution of responses, identify potential outliers, and detect patterns of agreement or divergence between accuracy and relevance assessments. Visualization tools supported the initial interpretation of score dispersion and item-level trends.

Exact agreement count: For each metric, the number of identical responses across raters was computed, providing a direct measure of consensus. This allowed metrics with high or low rating uniformity to be identified.

Analysis of variance: A one-way ANOVA [53] was applied to determine whether significant differences existed in ratings across metrics. This tested the null hypothesis that all items were evaluated similarly. Where significant results were observed, Tukey's Honestly Significant Difference (HSD) [54] post-hoc test was used to identify which pairs of metrics differed significantly.

Kendall's coefficient of concordance (W) [55]: This non-parametric statistic was used to measure the overall level of agreement among raters. Given the ordinal nature of the Likert-scale data, Kendall's W provided an appropriate indicator of consensus across all experts for both accuracy and relevance dimensions.

Internal consistency: To assess whether the metric evaluations consistently reflected underlying constructs, Cronbach's alpha[56] was calculated. Values closer to 1.0 indicate high internal consistency.

Inter-rater reliability: Fleiss' Kappa [57] was computed to evaluate the extent of agreement beyond chance across multiple raters for categorical Likert-scale responses. This complemented Kendall's W by offering a more conservative measure that accounts for probability of random agreement.

The selected statistical techniques were chosen to comprehensively assess the reliability, consistency, and interpretability of the expert evaluations, particularly given the ordinal nature of the data and the small sample size typical of expert judgment studies.

4.6. Tier-based classification of metrics

To support selection of a refined and robust set of evaluation metrics, a tier-based classification scheme is proposed. It integrates both the central tendency and variability of expert ratings. The algorithm (see Algorithm 1) systematically categorizes each metric based on its mean and standard deviation values for both accuracy and relevance.

This approach balances two critical dimensions:

1. *Rating quality (mean)*: Higher average scores reflect stronger expert endorsement of the metric's effectiveness and relevance.
2. *Rating consistency (standard deviation)*: Lower variability indicates greater agreement among raters and stronger interpretative cohesion.

The algorithm was implemented in R and applied to each metric in the dataset, classifying metrics into one of three mutually exclusive tiers:

Tier 1 – Core Metrics: Metrics that meet stringent quality and consistency thresholds. These include metrics with exceptionally high relevance (mean ≥ 4.25), or those with above-average accuracy (≥ 3.66) and relevance (≥ 4.0), combined with low variability (SD ≤ 1.0 for accuracy and ≤ 0.89 for relevance). These metrics represent the most reliable and well-supported items.

Tier 2 – Moderate-agreement Metrics: Metrics that demonstrate generally positive evaluations but fall short of Tier 1 criteria. These items show acceptable accuracy and relevance with moderate variability and may be candidates for further refinement or contextual validation.

Tier 3 – Low-agreement Metrics: Metrics characterized by low perceived quality (mean < 3.0 for either dimension) or high disagreement among experts (SD > 1.2 for accuracy or relevance). These items are flagged as potentially unreliable or inconsistent and are less suitable for continued use without revision.

Algorithm 1. Tier-based classification scheme of metrics

for each row in `metrics_summary` **do**
  $a_\mu \leftarrow$ accuracy_mean
  $a_\sigma \leftarrow$ accuracy_sd
  $r_\mu \leftarrow$ relevance_mean
  $r_\sigma \leftarrow$ relevance_sd
  if $r_\mu \geq 4.25$ **then**
    assign Tier $\leftarrow$ "Tier 1: Core metrics"
  else if $a_\mu \geq 3.66$ **and** $r_\mu \geq 4.0$ **and** $a_\sigma \leq 1.0$ **and** $r_\sigma \leq 0.89$ **then**
    assign Tier $\leftarrow$ "Tier 1: Core metrics"
  else if $a_\mu < 3.0$ **or** $r_\mu < 3.0$ **or** $a_\sigma > 1.2$ **or** $r_\sigma > 1.2$ **then**
    assign Tier $\leftarrow$ "Tier 3: Low-agreement metrics"
  else
    assign Tier $\leftarrow$ "Tier 2: Moderate-agreement metrics"
  end if
end for

This tiered structure facilitates evidence-based metric selection by distinguishing high-confidence metrics from those requiring further scrutiny.

5. Findings

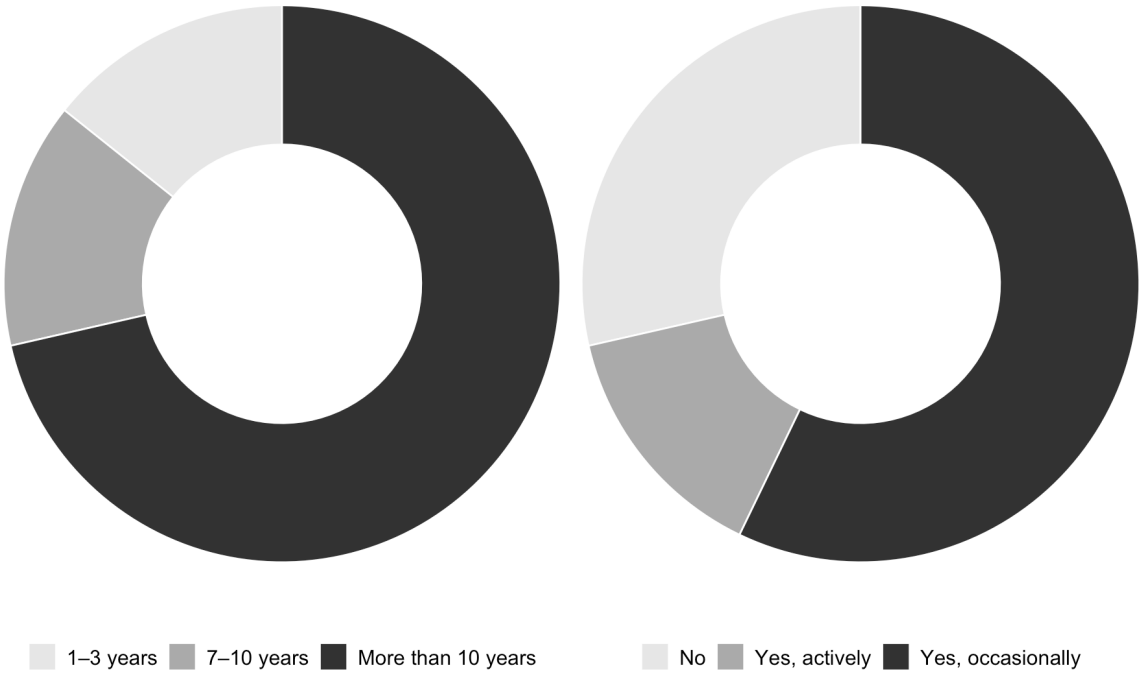
5.1. Population characterization

In total, seven participants took part in the study. The initial part of the survey aimed to profile the respondents, offering context for interpreting their perspectives. This characterization included institutional affiliation, professional roles, experience in the field, familiarity with relevant technologies, and engagement with the API-related community.

All participants met predefined expertise criteria, including recognized contributions to the API management domain, substantial professional experience with API technologies, and active participation in API development, research, or platform implementation. Professional affiliation, years of experience, and community involvement are shown in Figure 1 and summarized below.

Professional affiliation: Participants came from a mix of sectors including academia, private industry, and the public sector. The sample reflects a balanced representation between academic and corporate environments (three participants each), with more limited input from government organizations (one participant).

Years of experience: Participants demonstrated considerable API expertise across varying career stages. As illustrated in Figure 1a, five participants (71%) reported 10 or more years of API-related experience, one participant (14%) reported 7-10 years, and one participant (14%) reported 3 years of experience. This distribution indicates that expert evaluations were predominantly provided by senior professionals with extensive accumulated knowledge in the API domain, while still maintaining the minimum expertise threshold of 3 years for all participants.



(a) Years of API experience (b) Involvement in API communities

Figure 1. Professional background and community engagement of the respondents



(a) Primary roles

(b) API technologies and tools

Figure 2. Professional roles and API-related tools expertise of specialists

Community engagement: Figure 1b shows that most respondents have at least some involvement in API-related communities, events, or open-source projects, though active engagement is relatively rare. This suggests a group that is technically experienced, but more selectively engaged in community-oriented activities.

Professional roles: Respondents were asked to identify their primary roles in API-related work; multiple selections were allowed. Figure 2a shows a concentration in academic and technical leadership positions, as well as developer roles. This indicates that the surveyed group brings a combination of hands-on technical experience and strategic oversight.

API technologies and tools: Participants reported extensive hands-on experience with a wide range of API technologies. As depicted in Figure 2b, REST APIs, OpenAPI/Swagger, and Postman stood out as the most consistently used tools. The data also reflect notable use of legacy protocols like SOAP/XML and growing adoption of cloud-native and CI/CD-oriented practices.

Professional roles and API-related expertise can be seen next in Figure 2.

5.2. Expert ratings

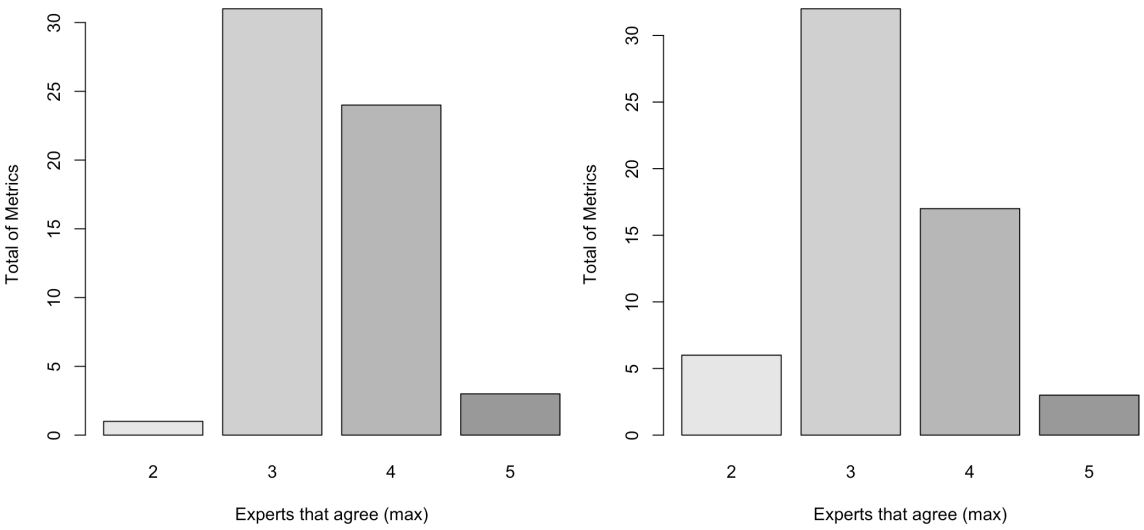
This section presents a comprehensive overview of expert ratings across the metrics based on two assessed criteria: accuracy and relevance.

Figure 3 displays the concordance per item. Each bar shows the maximum number of experts who assigned the same rating to a metric's accuracy (3a) and relevance (b).

Overall, expert agreement was moderate. The most common level of concordance ranged between three and four experts assigning the same rating for both accuracy and relevance.

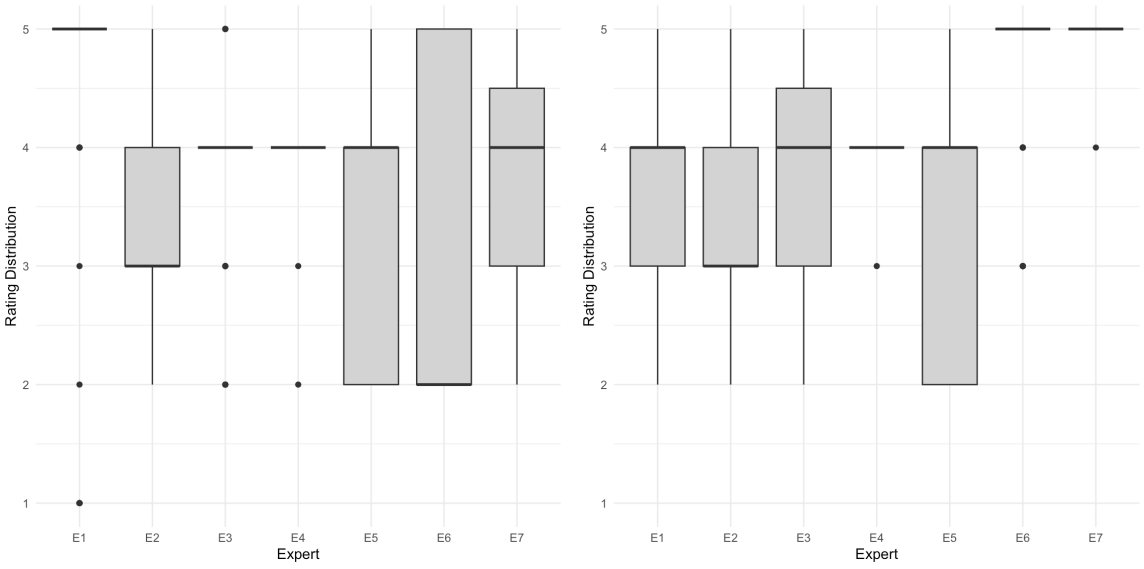
Figure 4 illustrates the distribution of ratings given by each expert. The rating distribution per expert showed no strong bias from any individual rater for both accuracy (4a) and relevance (4b) criteria. Some experts used the full Likert-scale range, while others gravitated toward middle and upper ratings, indicating slight variation in interpretive thresholds.

Finally, Figure 5 depicts how ratings were distributed across individual metrics. Metric-level rating distributions (subfigures 5a and 5b) revealed that experts generally evaluated the metrics favorably, particularly in terms of relevance (5b). The most common



(a) Metrics accuracy (b) Metrics relevance

Figure 3. Expert concordance per metric

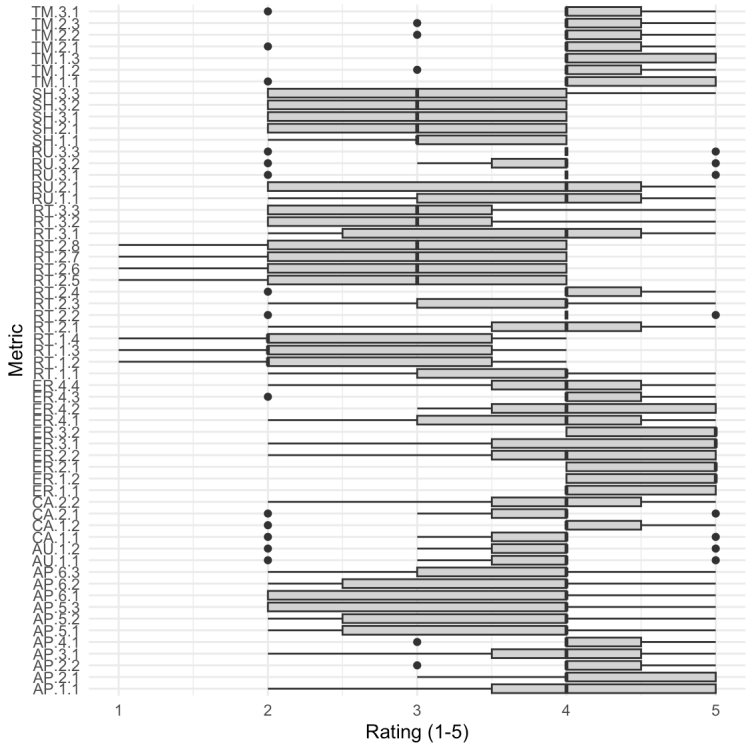


(a) Accuracy ratings for each expert (b) Relevance ratings for each expert

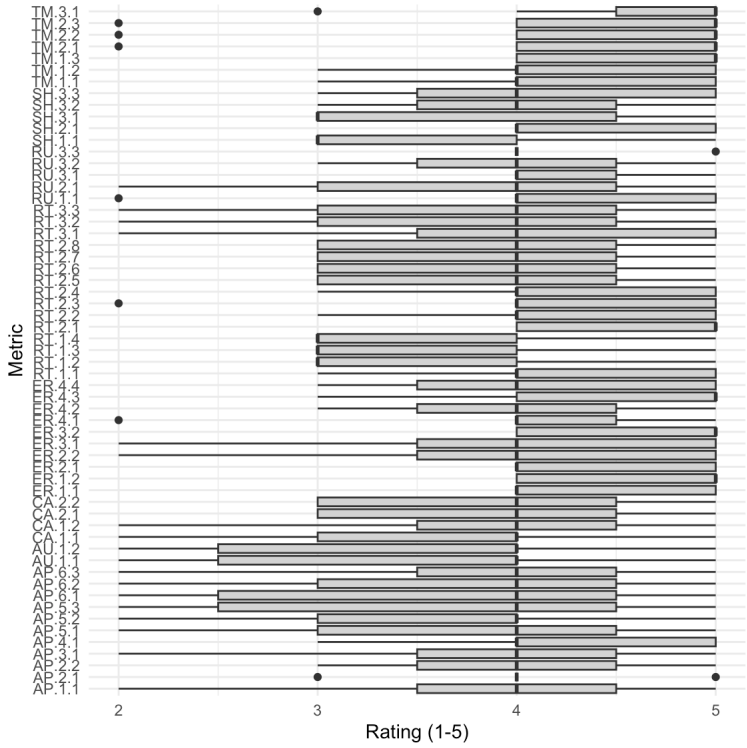
Figure 4. Overview of experts ratings' distribution

response was "Agree", followed by "Strongly Agree". No "Strongly Disagree" responses were given for relevance, but a few appeared under accuracy (5a), suggesting that while most metrics were seen as useful, some definitions may need refinement.

These observed patterns in expert behavior reinforced the need for further analysis of inter-rater consensus using statistical methods suggested in the literature. The analysis conducted in this work is described in the following section.



(a) Accuracy ratings distribution for each metric



(b) Relevance ratings distribution for each metric

Figure 5. Overview of metrics scores distribution

5.3. Statistical validation

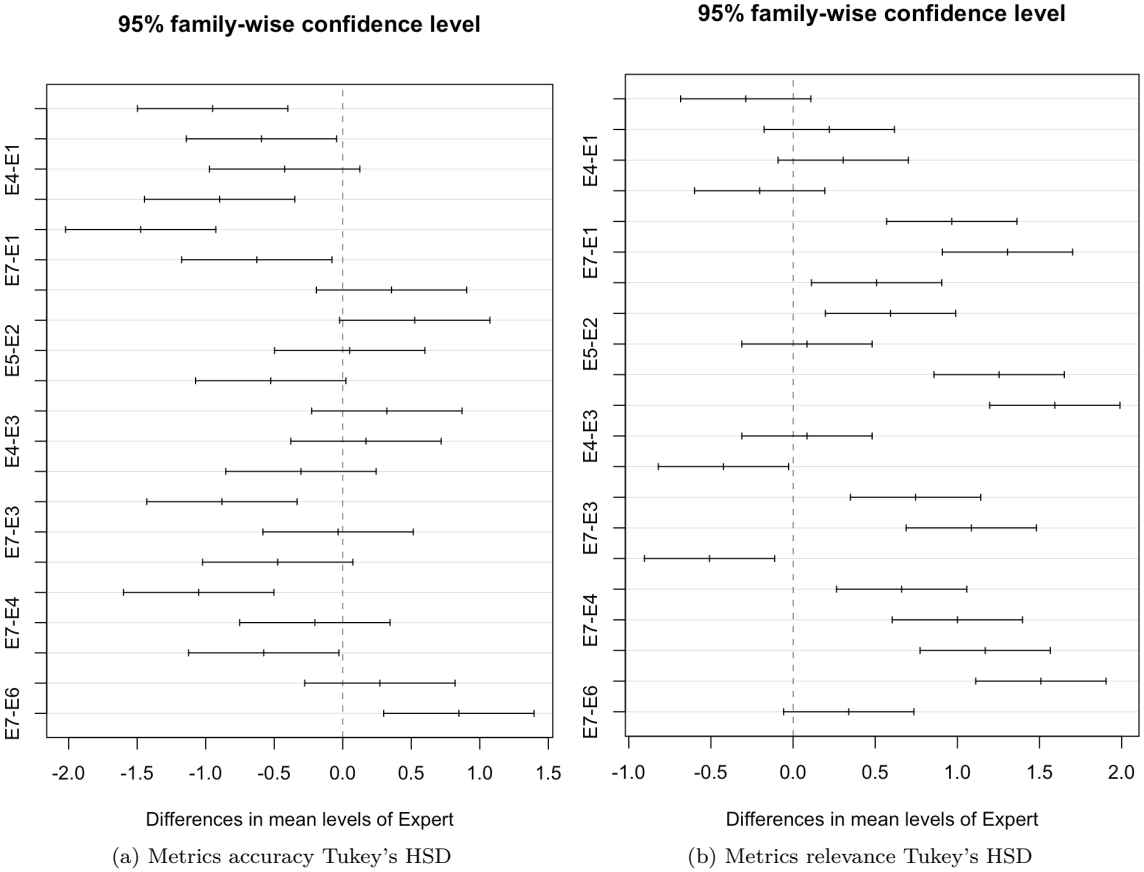
A multi-method statistical analysis was conducted to assess inter-rater agreement. Table 3 summarizes the main indicators. Descriptive statistics of metrics accuracy and relevance ratings can be seen in Appendices B and C, respectively.

Table 3. Summary of inter-rater agreement statistics over metrics

Statistic	Accuracy ratings	Relevance ratings
ANOVA [53]	$F(6, 406) = 12.49$, $p < 0.001$	$F(6, 406) = 39.9$, $p < 0.001$
Cronbach's Alpha [56]	$\alpha = 0.57$ (CI: 0.37–0.72)	$\alpha = 0.33$ (CI: 0.03–0.56)
Kendall's W [55]	$W = 0.0814$, $p = 0.997$	$W = 0.0661$, $p = 1.000$
Fleiss' Kappa [57]	$\kappa = -0.0393$, $p = 0.0221$	$\kappa = -0.0389$, $p = 0.0338$

The results revealed low to moderate levels of inter-rater agreement. Cronbach's alpha showed moderate internal consistency for accuracy but low reliability for relevance. Kendall's W and Fleiss' Kappa suggested low or even sub-chance agreement in both dimensions.

ANOVA tests confirmed statistically significant differences in expert scoring. This led to performing Tukey's HSD post hoc tests to identify specific differences in expert scoring behavior across metrics. Figure 6 summarizes the Appendix D as two subfigures, each corresponding to one evaluation criterion, namely accuracy and relevance.



(a) Metrics accuracy Tukey's HSD

(b) Metrics relevance Tukey's HSD

Figure 6. Tukey's HSD multiple comparisons of rating means between experts

Regarding accuracy (Figure 6a), significant differences were found between several expert pairs. Specifically, experts E2, E3, E5, E6 and E7 each differed significantly from E1 ($p < 0.05$), with E6–E1 showing the largest mean difference ($\Delta = -1.47$, $p < 0.0001$). Additionally, E6 differed significantly from E3, E4, and E5, and E7 differed from E6. These results suggest that E1 generally gave higher ratings than most others, with E6 showing the lowest relative values.

In contrast, the second Tukey test (6b) revealed that experts E6 and E7 had significantly higher scores than most others. E6 differed significantly from E1, E2, E3, and E4 ($p < 0.001$), and E7 differed from all groups except E6, with a maximum mean difference of 1.59 over E2. Notably, E5 showed significantly lower values than E3 and E4, but did not differ significantly from E1 or E2.

5.4. The core metrics set

Based on the expert assessment, Table 4 presents the definitions of 24 core metrics. Importantly, names and definitions were refined in response to feedback, as specialists remarked on the need for adopting consistent naming conventions and comprehensive descriptions.

Table 4. API Gateway core metrics set

Code	Human-readable name	Description
AP.2.1	Successful Response Rate	Percentage of API responses with HTTP 2xx status codes relative to the total number of API requests during a specified time period.
AP.2.2	Successful Request Count	Total number of API requests that resulted in successful responses (HTTP 2xx status codes) during a specified time period.
AP.4.1	Request Throughput	Number of API requests processed per second (or defined time unit) by the API Gateway.
ER.1.1	Total Error Count	Total number of failed API requests (HTTP 4xx and 5xx status codes) recorded during a specified time period.
ER.1.2	Overall Error Rate	Percentage of failed API requests (HTTP 4xx and 5xx status codes) relative to the total number of API requests during a specified time period.
ER.2.1	Client Error Count	Total number of client-side errors (HTTP 4xx status codes) recorded during a specified time period.
ER.3.2	Server Error Rate	Percentage of server-side errors (HTTP 5xx status codes) relative to the total number of API requests during a specified time period.
ER.4.2	Exception Count	Total number of API requests that encountered unexpected conditions or exceptions during processing, regardless of HTTP status code.
ER.4.3	Authentication Error Count	Total number of authentication-related failures during a specified time period, including invalid credentials, expired tokens, missing authentication headers, and authorization failures.
RT.1.1	Internal Processing Time	Time spent by the API Gateway processing requests internally, including routing, plugin execution, and request transformation.

Table 4 (cont.)

Code	Human-readable name	Description
RT.2.1	Total Response Time	Complete end-to-end response time from when the API Gateway receives a request until it returns a response to the client, including both gateway processing time and upstream service response time.
RT.2.2	Maximum Response Time	Highest observed response time for any API request during a specified time period.
RT.2.4	Response Time Standard Deviation	Statistical measure of response time variability, indicating the consistency of API performance.
RU.3.1	Average Memory Usage (Last Hour)	Average memory consumption by the API Gateway instance over the past hour, measured in bytes or percentage of available memory.
RU.3.2	Maximum Memory Usage	Peak memory utilization observed by the API Gateway instance during its operational period.
RU.3.3	Node Memory Usage	Memory consumption per individual API Gateway node in a distributed or clustered deployment.
SH.2.1	Upstream Health Status	Operational status indicator for upstream services and backend systems, typically represented as healthy, degraded, or unavailable.
TM.1.1	Total API Count	Total number of individual APIs deployed and actively managed within the API Gateway platform.
TM.1.2	Total Product Count	Total number of API products or bundles configured in the API Gateway, where each product may contain multiple APIs with associated usage plans, rate limits, and access policies.
TM.1.3	Total Request Count	Total number of API requests received and processed by the API Gateway during a specified time period, regardless of success or failure status.
TM.2.1	Total Data Volume	Combined volume of data transferred through the API Gateway during a specified time period, including both inbound requests and outbound responses.
TM.2.2	Inbound Data Volume	Total volume of data received by the API Gateway from clients during a specified time period, including request headers, body content, and metadata.
TM.2.3	Outbound Data Volume	Total volume of data transmitted from the API Gateway to clients during a specified time period, including response headers, body content, and metadata.
TM.3.1	Throttling Error Count	Number of API requests rejected due to exceeding configured rate limits, quota restrictions, or throttling policies.

6. Discussion

The need for a formal, expert-validated, metrics-based quality model for API Management has been emphasized by recent work [6, 9], which identified fragmentation in existing research and a lack of standardized quality evaluation frameworks. Our study directly addressed this gap by providing a set of metrics, empirically validated by expert judgment. This section interprets the findings from each research question, synthesizes cross-cutting themes, and discusses their practical, theoretical, and methodological implications.

6.1. Interpretation of research questions

6.1.1. RQ1: Expert agreement levels

RQ1 asked: **To what extent do domain experts agree on the accuracy and relevance of existing API Gateway quality metrics?**

The statistical analysis (Subsection 5.3, Table 3) revealed low to moderate inter-rater agreement. However, this apparent lack of consensus requires careful interpretation in light of the study's deliberate sampling strategy. The expert panel was intentionally composed of diverse profiles spanning academia and industry, ranging from emerging to highly senior professionals (3 to more than 10 years), and representing varied organizational contexts. This heterogeneity was a methodological design choice aimed at ensuring the validated metrics would be applicable across diverse API Gateway deployment scenarios rather than reflecting a narrow, context-specific perspective.

Given this intentional diversity, the observed inter-rater variability reflects two distinct phenomena. First, legitimate perspective diversity: Experts from different backgrounds may appropriately weight accuracy and relevance differently based on their operational contexts. For example, an enterprise architect managing cloud-native deployments may prioritize different metric characteristics than a researcher focusing on quality model formalization. The concordance analysis (Figure 3) supports this interpretation: most metrics had 3–4 experts assigning the same rating, indicating **moderate practical consensus within subgroups** despite low overall statistical coefficients.

Second, scale interpretation differences: Distribution analysis (Figures 4 and 5) showed that experts generally evaluated metrics favorably, particularly for relevance, where "Agree" and "Strongly agree" dominated responses. The ANOVA results ($F(6, 406) = 12.49$ and $F(6, 406) = 39.9$, $p < 0.001$) and Tukey's HSD post hoc tests (Figure 6) revealed that some experts (e.g., E6 and E7) consistently used higher ratings while maintaining similar relative rankings of metrics. This suggests differences in how individuals use Likert scales rather than fundamental disagreement about metric value.

Importantly, this variability should not be interpreted as validation failure. Rather, it demonstrates that the expert panel successfully captured the heterogeneity of real-world API Gateway stakeholders. The tier-based classification (RQ4) was specifically designed to handle this diversity by identifying metrics with robust agreement despite varied perspectives (Tier 1), while flagging those requiring context-specific refinement (Tiers 2 and 3).

6.1.2. RQ2: Core metrics identification

RQ2 asked: Which metrics demonstrate strong consensus and high scores among experts, and can thus be considered core indicators?

Despite moderate overall agreement, the tier-based classification algorithm (Algorithm 1) successfully identified **24 core metrics** (Tier 1) that met stringent quality and consistency thresholds. These metrics demonstrated either exceptionally high relevance (mean ≥ 4.25) or combined above-average accuracy (≥ 3.66) and relevance (≥ 4.0) with low variability (SD ≤ 1.0 for accuracy and ≤ 0.89 for relevance).

The core set spans all major capability categories: **Errors** (9 metrics), **Traffic monitoring** (7 metrics), **Latency and response time** (4 metrics), **Resource utilization** (3 metrics), and **System health** (1 metric). Notably, error-related metrics (ER.x codes)

received the highest combined accuracy and relevance scores, with metrics such as ER.1.2 (Overall Error Rate) and ER.2.1 (Client Error Count) achieving means above 4.4 with standard deviations below 0.54.

This finding directly addresses a practical need: **operational metrics that experts unanimously recognize as essential** can serve as a validated foundation for quality assessment frameworks. The 24 core metrics represent the intersection of theoretical importance and practical feasibility, as they are both expert-endorsed and already implemented in production systems [9].

6.1.3. RQ3: Capability coverage

RQ3 asked: What categories of API management capabilities are mostly represented by the validated metrics?

The core metrics set (Table 4) demonstrates **comprehensive coverage across key API Gateway concerns**, with particularly strong representation in areas aligned with practitioner priorities identified in [3]:

- **Error handling and reliability** (9 metrics): total error counts, error rates by type (client/server), authentication failures, and exceptions. This aligns with reliability being consistently prioritized by practitioners.
- **Traffic monitoring and functional suitability** (7 metrics): API counts, request volumes, data transfers, and throttling. These support functional suitability assessment by tracking what the gateway actually processes.
- **Performance efficiency** (4 metrics): response times, processing latencies, and performance variability. Directly addresses performance efficiency, another top practitioner priority.
- **Resource utilization** (3 metrics): memory usage patterns across deployment configurations. Essential for operational efficiency and scalability assessment.
- **System health** (1 metric): upstream service status monitoring.

Notably absent from Tier 1 are **Caching** and **API Utilization** metrics, which showed lower consensus. This gap may reflect variability in how organizations implement caching strategies or the context-dependent nature of utilization thresholds.

The coverage pattern validates the practice-oriented approach: validated metrics concentrate in areas where **both academic research [6] and practitioner concerns [3] converge** – namely, performance efficiency, reliability, and functional suitability.

6.1.4. RQ4: Classification framework

RQ4 asked: How can API Gateway quality metrics be classified based on expert ratings to guide their adoption, review, or exclusion?

The tier-based classification (Algorithm 1) successfully translated mixed expert assessments into **actionable categories**:

- **Tier 1 (24 metrics, 41%)**: ready for immediate adoption in quality models, dashboards, and SLA definitions. High scores with low variability indicate robust expert endorsement.
- **Tier 2 (moderate-agreement metrics)**: candidates for definitional refinement. These metrics address relevant concerns but require improved clarity or contextual framing.
- **Tier 3 (low-agreement metrics)**: flagged for revision or exclusion. Many involved percentile-based indicators (e.g., RT.1.2–RT.1.4: IPT percentiles) or compound constructs that experts found ambiguous.

This classification mechanism provides **decision support guidance** for multiple stakeholder groups: researchers can prioritize Tier 1 metrics in quality models; platform developers can focus implementation efforts on high-consensus measures; and practitioners gain clarity about which metrics warrant investment in monitoring infrastructure.

Importantly, the framework is **iterative**: Tier 2 and 3 metrics can be refined and re-evaluated, creating a pathway for continuous improvement of the metric catalog.

6.2. Synthesis and cross-cutting themes

Having interpreted each research question individually, several cross-cutting themes emerge that illuminate the broader validation challenge.

Rating asymmetry: in all RQs, relevance scores consistently exceeded accuracy scores (Figures 5a vs. 5b). This asymmetry reveals that experts agree on *what* should be measured, but disagree on *how* clearly it is defined. Multiple experts noted that "all metrics may be relevant," yet accuracy ratings varied significantly (Section 5.3). This suggests the catalog captured the right concepts but requires semantic refinement – a finding that directly motivates the implications discussed below.

Interpretive variability as diagnostic signal: The low agreement coefficients (RQ1) should not be interpreted as validation failure. Rather, they reflect the field's evolving maturity in API Gateway quality assessment. Given the novelty and diversity of deployment patterns, strong consensus on metric priorities does not yet exist. Expert divergence thus serves as a diagnostic indicator: it identifies metrics needing contextual framing (Tier 2) versus those with broad agreement (Tier 1).

Convergence with prior research: The capability coverage patterns (RQ3) and core metrics identified (RQ2) align remarkably with findings from our previous studies. The SMS [6] identified performance efficiency, reliability, and functional suitability as most-addressed attributes. The practitioners' survey [3] confirmed these as top priorities. The platform analysis [9] showed most implementations focus on throughput, latency, and error tracking. The current validation study reinforces this convergence: Tier 1 metrics concentrate precisely in these areas, providing strong triangulated evidence for their centrality to API Gateway quality.

In an effort to improve the preliminary catalog assessed in this paper, the evaluation process enabled the extraction of a high-confidence subset of metrics suitable for immediate adoption, while identifying others in need of refinement. The tier-based classification served as a constructive lens through which to interpret mixed expert assessments. Rather than enforcing a strict inclusion/exclusion boundary, the proposed structure enabled a more nuanced prioritization, by identifying metrics that are already robust (Tier 1), potentially useful but needing refinement (Tier 2), and those not yet ready for adoption (Tier 3).

Interestingly, many Tier 3 metrics involved percentile-based indicators or compound constructs. These are often seen in modern observability tools but may require contextual framing – if the system is considered "as a whole," as some experts noted – or improved phrasing to be uniformly understood by diverse evaluators. Lower scores may reflect this interpretive challenge rather than a lack of utility per se.

For Tier 2 metrics, refinement is warranted. This may include improving wording, providing contextual examples, or offering visualizations to support interpretation. Iterative validation, possibly with practitioner feedback or scenario-based calibration, could improve consensus.

From a practical standpoint, the Tier 1 catalog can serve as a ready-to-adopt artifact for API Gateway operators and tool developers, among other roles. These metrics offer validated indicators that align with expert expectations and observed best practices, supporting immediate integration into dashboards, alert systems, Service Level Indicators (SLI), and Service Level Agreement (SLA) implementation.

Although 59 proposed metrics were initially comprised, the refined Tier 1 set is still broader and more comprehensive than what most mainstream API Gateway products currently provide. According to the quantitative analysis performed in our previous work [9], most commercial and open-source API Gateway platforms expose between 8 and 20 metrics, typically focused on throughput, latency, and error tracking. This highlighted the added value of the current validation effort, which consolidates expert consensus while advancing beyond the limited operational visibility offered by typical platforms.

6.3. Practical and theoretical implications

The findings from RQ1–RQ4 give rise to several practical and theoretical implications for API Gateway quality assessment:

Semantic clarity as a design imperative: The rating asymmetry identified in RQ1 (high relevance, variable accuracy) directly demonstrates that ambiguity in metric definitions erodes consensus even when concepts are recognized as important. For example, percentile-based metrics (RT.1.2–RT.1.4, classified as Tier 3) received lower accuracy scores despite measuring conceptually relevant constructs. This empirical finding underscores that metric definitions must be unambiguous, operational, and supported by measurement context to achieve widespread adoption in diverse enterprise settings.

Diversity as validation strength: The moderate inter-rater agreement observed in this study should be contextualized within the deliberate expert selection strategy. High agreement among homogeneous experts would indicate narrow applicability, while moderate agreement among diverse experts suggests the validated metrics address concerns across varied organizational contexts. This heterogeneity strengthens the external validity of the core metrics set: the 24 Tier 1 metrics achieved consensus despite panel diversity, indicating they represent genuinely universal API Gateway quality concerns rather than context-specific preferences. Future validation studies should consider panel composition as a tunable parameter: homogeneous panels for context-specific metrics, heterogeneous panels for general-purpose frameworks.

This study also underscores the limitations of expert-based validation studies. While expert judgment is indispensable in the early stages of metric validation, its effectiveness improves substantially when combined with rater training, controlled scenarios, or complementary quantitative validation.

Furthermore, the variability observed here reflects the field's evolving maturity. Given the novelty and flexibility of API Gateway deployments, strong consensus on metric priorities may not yet exist. In this light, expert divergence is not a failure but a diagnostic signal: a call for contextualized metrics that adapt to organizational needs rather than a one-size-fits-all solution.

Another important consideration emerging from both prior literature and expert feedback is the role of metric metadata in ensuring understanding, re-utilization, integration, and automation. In this sense, as argued in [58], including such metadata can mitigate inconsistent or erroneous interpretations in metrics analysis.

6.4. Methodological implications

The methodological approach employed in this study – integrating multiple evidence sources through a practice-oriented validation strategy – yielded insights beyond the specific metrics validated:

The methodological approach introduced in this paper reflects a deliberate research strategy that integrates multiple evidence sources to address the API Gateway quality assessment gap comprehensively. The convergence across four complementary studies strengthens confidence in the resulting validated core metrics set:

- SMS [6] documented what research says about API management quality by identifying practices, capabilities, research methods, and outcomes, while revealing fragmentation and absent formal frameworks.
- Practitioners' survey [3] captured what users prioritize, revealing that functional suitability, performance efficiency, reliability, and security are consistently highlighted as most critical – reinforcing the overlap between academic focus and practitioner needs.
- Platform analysis [9] revealed what is actually measured in production environments across 68 diverse implementations, with most platforms exposing 8–20 metrics focused on throughput, latency, and error tracking.
- Expert validation (current study) determined what domain experts consider both accurate and relevant, yielding 24 high-confidence core metrics organized in seven capability-based categories.

The remarkable convergence on performance efficiency, reliability, and functional suitability across all four perspectives provides strong validity evidence. This demonstrates that our core metrics address concerns that are simultaneously theoretically recognized in academic research, prioritized by practitioners in operational contexts, implemented in real-world platforms, and endorsed by domain experts.

This four-way validation significantly strengthens confidence in the resulting catalog as a foundation for standardization and enables bidirectional knowledge transfer:

- **From practice to research:** The validated catalog informs future quality model development with proven, measurable constructs that have demonstrated feasibility in production environments.
- **From research to practice:** Expert validation (including academic perspectives) elevates ad-hoc industry metrics to theoretically informed standards, providing the rigor necessary for formal standardization efforts.

The practice-oriented approach for the validation phase proved methodologically sound for several reasons that became evident through the validation process:

First, experts evaluated operational metrics more consistently when those metrics reflected actual implementation patterns. Although inter-rater agreement was moderate overall (Section 5.3), experts recognized these metrics as addressing genuine operational concerns, as evidenced by generally high relevance scores (Figure 5b). Several experts noted that the metrics reflected "what we actually monitor in production," suggesting that grounding in real-world practice aided expert assessment.

Second, the variability in accuracy ratings (Section 5.3) revealed that even operationalized metrics benefit from definitional refinement. Had we included purely theoretical metrics without implementation precedent, agreement might have been even lower, as experts would lack shared reference points for interpretation. The practice-grounded approach provided a common foundation for evaluation.

Third, the finding that the proposed core metrics set is broader and more comprehensive than what most mainstream API Gateway products currently provide validates the synthesis approach: by aggregating metrics across 68 platforms, we identified a richer measurement space than any single platform exposes. This demonstrates the value of systematic analysis over reliance on any particular product's metric set.

7. Threats to validity

7.1. Construct validity

To mitigate ambiguity, all experts received a common set of definitions and instructions. The questionnaire was reviewed and refined by two authors prior to deployment. Accuracy and relevance were evaluated using standardized five-point Likert scales, and concise operational definitions were provided for both criteria. However, some metric descriptions may still have included terminology open to multiple interpretations, especially in the absence of domain-specific calibration or contextual examples.

No rater calibration session or shared interpretation workshop was conducted before or after the evaluation, potentially allowing diverging understandings of the criteria. Future replications should consider including a calibration phase to improve consistency.

Although expert feedback was based on clearly defined Likert scales (1 to 5), and criteria definitions were provided, the absence of fully specified metric metadata (e.g., scope, temporal constraints, or measurement conditions) may have affected how consistently experts interpreted some metrics – an issue previously highlighted [58] as critical for robust evaluation processes.

7.2. Internal validity

Experts were selected through formal invitations based on predefined criteria: authorship of widely cited research, leadership in API-related communities, and authorship of influential books. The deliberate inclusion of diverse profiles (academia, industry, public sector; varying experience levels; different organizational contexts) was a methodological design choice to mitigate selection bias and ensure broad applicability of validated metrics.

While this heterogeneity may contribute to lower statistical agreement coefficients, it strengthens the ecological validity of findings by capturing real-world stakeholder diversity. The trade-off between consensus (favoring homogeneous panels) and generalizability (favoring heterogeneous panels) was intentionally resolved in favor of the latter, consistent with the study's goal of developing general-purpose quality metrics rather than context-specific solutions.

In particular, the study deliberately excluded API Gateway developers to avoid evaluation bias tied to specific implementations. However, this exclusion may have omitted valuable operational insights.

Additionally, the evaluation was not anonymized – participants were asked to provide contact information for potential follow-up. Although identities were not considered during analysis, the lack of full anonymity could introduce minor social desirability bias. No such effects were observed, and rating distributions suggest a diversity of perspectives.

7.3. External validity

This study aimed to validate metrics applicable across varied API Gateway architectures, including cloud-native and enterprise-grade environments. The expert panel was intentionally diverse in academic and industry backgrounds, as well as in geographic representation, to increase the generalizability of results. Nonetheless, the inherent variability of API Gateway deployments and organizational contexts presents a challenge to universal applicability.

To enhance external validity, the metric set evaluated in this study was derived from real-world implementations analyzed in prior work [9]. This grounding in practical usage data complements the expert assessments and increases the likelihood that the validated metrics are broadly relevant.

7.4. Conclusion validity

The statistical analyses indicated low agreement among raters. These results reflect the known difficulty of achieving consensus in expert-based evaluations. Nonetheless, the adoption of a tiered classification algorithm allowed us to convert variability into actionable categorization – balancing mean scores with standard deviation to identify Core, Promising, and Low-Agreement metrics. This approach mitigated the threat posed by low agreement and provided a structured path for metric refinement and prioritization.

7.5. Study limitations

Beyond the methodological validity concerns discussed above, this study has several inherent limitations that should be considered when interpreting and applying the findings:

Sample size and generalizability: While seven experts represent an adequate sample for judgment studies, the relatively small panel limits statistical power and may not capture the full diversity of perspectives across the global API management community. The findings reflect expert consensus within this specific group and may not generalize to all organizational contexts or emerging API Gateway paradigms.

Temporal scope: The metric catalog was derived from platforms analyzed in 2024–2025. As API Gateway technologies and operational practices evolve, new metrics may emerge and current metrics may become obsolete or require redefinition. The validated set should be viewed as a snapshot of contemporary best practices rather than a permanent, unchanging standard.

Practical validation absence: This study validates metrics through expert judgment but does not include empirical testing in operational environments. The metrics' actual predictive validity – their ability to correlate with real system failures, performance issues, or user satisfaction – remains to be demonstrated through field studies.

Metric definition granularity: While the study validates metric concepts and relevance, it does not specify complete measurement procedures, including data collection methods, aggregation rules, threshold values, or temporal windows. Research and organizations implementing these metrics will need to operationalize these details based on their specific contexts.

Coverage gaps: The catalog focuses on metrics actually deployed in existing platforms, potentially overlooking theoretically important but not yet implemented quality aspects. Novel quality concerns emerging from new architectural patterns (e.g., service mesh integration, serverless gateways) may not be adequately represented.

These limitations suggest directions for future research, including larger-scale validation studies, longitudinal tracking of metric evolution, and field deployment studies.

8. Concluding remarks and future work

By combining practical definitions, expert judgment, and a tiered classification mechanism, this study proposes a validated and categorized set of API Gateway quality metrics, grounded in expert evaluation and aligned with both academic and practitioner perspectives. The resulting classification represents a significant advancement toward the standardization of observability-based quality assessment in the API Management field.

The statistical techniques employed during this research were chosen to comprehensively assess the inter-reliability, internal consistency, and interpretability of the expert evaluations, particularly given the ordinal nature of the data and the small sample size typical of expert judgment studies. In this sense, variability in expert judgment should not be viewed solely as a limitation, but as a reflection of the heterogeneity during expert judgment studies.

The insights provided by the expert panel reflected diverse perspectives from both academia and industry, which added credibility and depth to the findings. Importantly, results reinforced the need for semantic clarity and contextual adaptability in metric design. Experts acknowledged that relevance often depends on implementation scenarios.

While statistical methods guided the consensus analysis, this work also explored practical strategies to translate expert feedback into concrete catalog improvements: thus, metrics were classified under three different tiers, and a set of 24 core metrics was outlined. This validated subset of metrics contributes to bridge the gap between metric adoption in real-world platforms and their theoretical grounding in software quality models. The validation effort not only confirms the feasibility of metric-based assessment but also expands the operational scope beyond what is typically available in current API Gateway offerings in the industry.

Several research directions can build upon the findings of this study.

First, a key next step is to consolidate the validated metrics into a formal quality model tailored to API Gateway systems. This model should: (i) Offer a multi-perspective structure encompassing quantitative, operational, and conceptual views of quality. (ii) Include a complete and standardized metadata specification to describe each metric, including its definition, interpretation guidance, aggregation rules, thresholds, scope, and implementation considerations. (iii) Link metrics to higher-level software product quality attributes, enabling traceability between observed values and desired system properties.

Second, some Tier 2 and Tier 3 metrics may still offer potential value if reformulated. Further rounds of validation – potentially using Delphi methods, group interpretation workshops, or scenario-based rating – could improve definitions and agreement levels.

Third, the core metrics (Tier 1) set establishes a ready-to-use artifact which can be deployed into monitoring tools, dashboards, or observability platforms. Prototypes or reference implementations can test their effectiveness in conveying quality insights, guiding operations, and supporting decision making.

Finally, future studies should apply the validated metrics in real operational environments (e.g., enterprise or cloud-native gateways) to evaluate: (i) Their correlation with actual system behavior and failures. (ii) Their usefulness in incident diagnosis, trend analysis, or capacity planning. (iii) Their acceptance and interpretability by practitioners and teams.

CRediT authorship contribution statement

E. dos Santos: conceptualization, data curation, formal analysis, methodology, investigation, resources, software, validation, visualization, writing – original draft, writing – review and editing.
S. Casas: conceptualization, funding acquisition, investigation, methodology, project administration, resources, supervision, writing – review and editing.
All authors reviewed the results and approved the final version of the manuscript.

Declaration of competing interest

The authors declare that no competing interests exist.

Data and code availability

The dataset supporting this study is publicly available through the Argentinean National Council of Scientific and Technical Research (CONICET) digital repository and can be accessed from http://hdl.handle.net/11336/264879.
It is provided in two files which contain: (i) Anonymized data collected from expert evaluations of the proposed software metrics (all personally identifiable information has been removed to preserve respondent confidentiality). (ii) The list of metrics evaluated by the experts, including each metric's definition and its final classification according to the tier-based scheme described in this article.
Both files are provided in a machine-readable format and are intended to support reproducibility and further analysis.
Additionally, the complete survey instrument (PDF export from Google Forms) is publicly available through the Universidad Nacional de la Patagonia Austral institutional repository at https://drive.google.com/file/d/1Yuws6AUKg2LMkX0AdQZF3M0BSBDc2bgb .

Funding

This research was supported by the Universidad Nacional de la Patagonia Austral (Argentina) under research project 29/A531, titled Administracion de APIs Web: Ecosistemas y Plataformas.

References

[1] S. Preibisch, *API development: A practical guide for business implementation success*. New York, NY: Springer Science+Business Media, 2018.
[2] M. Waseem, P. Liang, M. Shahin, A. Di Salle, and G. Márquez, "Design, monitoring, and testing of microservices systems: The practitioners' perspective," *arXiv preprint arXiv:2108.03384*, 2021.
[3] E. dos Santos and S. Casas, "API management and SQuaRE: A comprehensive overview from the practitioners' standpoint," in *Argentine Congress of Computer Science*. Springer, 2023, pp. 137–150.

[4] D. Bermbach and E. Wittern, "Benchmarking web API quality – revisited," *Journal of Web Engineering*, Vol. 19, No. 5-6, 2020, pp. 603–646.
[5] B. De, *API Management: An Architect's Guide to Developing and Managing APIs for Your Organization*. Apress Berkeley, CA, 2023.
[6] E. dos Santos and S. Casas, "Unveiling quality in API management: A systematic mapping study," in *L Latin American Computer Conference (CLEI)*. IEEE, 2024, pp. 1–10.
[7] B. Iyer and M. Subramaniam, "The strategic value of APIs," *Harvard Business Review*, Vol. 1, No. 7, 2015, p. 2015.
[8] *ISO/IEC 25010:2011 – Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models*, International Organization for Standardization; International Electrotechnical Commission Std., 2011.
[9] E. dos Santos and S. Casas, "A catalog of API gateway metrics and its quantitative evaluation," *DYNA*, Vol. 92, No. 237, 2025, pp. 106–114.
[10] K.J. Stol and B. Fitzgerald, "The ABC of software engineering research," *ACM Transactions on Software Engineering and Methodology*, Vol. 27, No. 3, 2018. [Online]. https://doi.org/10.1145/3241743
[11] S.L. Pfleeger and B.A. Kitchenham, "Principles of survey research: part 1: turning lemons into lemonade," *ACM SIGSOFT Software Engineering Notes*, Vol. 26, No. 6, 2001, pp. 16–18.
[12] B. Kitchenham and S.L. Pfleeger, "Principles of survey research part 4: Questionnaire evaluation," *ACM SIGSOFT Software Engineering Notes*, Vol. 27, No. 3, 2002, pp. 20–23.
[13] *ISO/IEC 25010:2023 – Systems and Software Engineering – Systems and Software Quality Requirements and Evaluation (SQuaRE) – Product Quality Model*, International Organization for Standardization; International Electrotechnical Commission Std., 2023.
[14] Y. Duan, G. Fu, N. Zhou, X. Sun, N.C. Narendra et al., "Everything as a service (XaaS) on the cloud: Origins, current and future trends," in *8th International Conference on Cloud Computing*. IEEE, pp. 621–628. [Online]. http://ieeexplore.ieee.org/document/7214098/
[15] C. Fehling, F. Leymann, R. Retter, W. Schupeck, and P. Arbitter, *Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications*. Springer Vienna. [Online]. https://link.springer.com/10.1007/978-3-7091-1568-8
[16] R.T. Fielding, "Architectural styles and the design of network-based software architectures."
[17] S. Andreo and J. Bosch, "API management challenges in ecosystems," in *Software Business*, S. Hyrynsalmi, M. Suoranta, A. Nguyen-Duc, P. Tyrväinen, and P. Abrahamsson, Eds. Springer International Publishing, Vol. 370, pp. 86–93, series Title: Lecture Notes in Business Information Processing. [Online]. http://link.springer.com/10.1007/978-3-030-33742-1_8
[18] A. Brown, J. Fishenden, and M. Thompson, *API Economy, Ecosystems and Engagement Models*. Palgrave Macmillan UK, pp. 225–236. [Online]. http://link.springer.com/10.1057/9781137443649_13
[19] J. Bloch, "How to design a good API and why it matters," in *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*. ACM, pp. 506–507. [Online]. https://dl.acm.org/doi/10.1145/1176617.1176622
[20] A. Gamez-Diaz, P. Fernandez, and A. Ruiz-Cortés, "Governify for APIs: SLA-driven ecosystem for API governance," in *Proceedings of the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2019. Association for Computing Machinery, pp. 1120–1123, event-place: Tallinn, Estonia. [Online]. https://doi.org/10.1145/3338906.3341176
[21] M. Medjaoui, E. Wilde, R. Mitra, and M. Amundsen, *Continuous API management*. O'Reilly Media, Inc., 2021.
[22] M. Mathijssen, M. Overeem, and S. Jansen, "Identification of practices and capabilities in API management: A systematic literature review," *arXiv preprint arXiv:2006.10481*, 2020.
[23] *ISO 9001:2015 – Quality Management Systems – Requirements*, International Organization for Standardization Std., 2015.
[24] *ISO/IEC/IEEE 24765:2017 – Systems and Software Engineering – Vocabulary*, International Organization for Standardization; International Electrotechnical Commission; Institute of Electrical and Electronics Engineers Std., 2017.

[25] P. Nistala, K.V. Nori, and R. Reddy, "Software quality models: A systematic mapping study," in *International Conference on Software and System Processes (ICSSP)*. IEEE, 2019, pp. 125–134.
[26] *ISO/IEC 9126:1991 – Software Engineering – Product Quality*, International Organization for Standardization; International Electrotechnical Commission Std., 1991, iSO/IEC 9126:1991.
[27] *ISO/IEC 9126:2001 – Software Engineering – Product Quality*, International Organization for Standardization; International Electrotechnical Commission Std., 2001.
[28] *ISO/IEC 14598:1999 – Software Engineering – Product Evaluation*, International Organization for Standardization; International Electrotechnical Commission Std., 1999.
[29] *ISO/IEC 25000:2014 – Systems and Software Engineering – SQuaRE – Guide to SQuaRE*, International Organization for Standardization; International Electrotechnical Commission Std., 2014.
[30] *ISO/IEC 25023:2016 – Systems and Software Engineering – Measurement of System and Software Product Quality*, International Organization for Standardization; International Electrotechnical Commission Std., 2016.
[31] *ISO/IEC 25040:2024 – Systems and Software Engineering – Quality Evaluation Framework*, International Organization for Standardization; International Electrotechnical Commission Std., 2024.
[32] S.S. Stevens, "On the theory of scales of measurement," *Science*, Vol. 103, No. 2684, 1946, pp. 677–680.
[33] N. Fenton and J. Bieman, *Software metrics: A rigorous and practical approach*. CRC Press, 2014.
[34] G. Poels and G. Dedene, "DISTANCE: A framework for software measure construction," KU Leuven Departement Toegepaste Economische Wetenschappen, DTEW Research Report 9937, 1999.
[35] N. Dalkey and O. Helmer, "An experimental application of the Delphi method to the use of experts," *Management Science*, Vol. 9, No. 3, 1963, pp. 458–467.
[36] J. Kontio, J. Bragge, and L. Lehtola, "The focus group method as an empirical tool in software engineering," in *Guide to advanced empirical software engineering*. Springer, 2008, pp. 93–116.
[37] M. Shaw, "Writing good software engineering research papers," in *25th International Conference on Software Engineering, 2003. Proceedings.* Portland, OR, USA: IEEE, 2003, pp. 726–736.
[38] M. Overeem, M. Mathijssen, and S. Jansen, "API-m-FAMM: A focus area maturity model for API management," *Information and Software Technology*, Vol. 147, 2022, p. 106890.
[39] R. Yamamoto, K. Ohashi, M. Fukuyori, K. Kimura, A. Sekiguchi et al., "A quality model and its quantitative evaluation method for web APIs," in *25th Asia-Pacific Software Engineering Conference (APSEC)*, 2018, pp. 598–607.
[40] A. Machini and S. Casas, "A preliminary GQM model to evaluate web API usability," *Memorias de las JAIIO*, Vol. 10, No. 2, 2024, pp. 1–13.
[41] V.R. Basili, G. Caldiera, and H.D. Rombach, "The goal question metric approach," in *Encyclopedia of software engineering*, 1994.
[42] M. Constanzo, S.I. Casas, G.B. Vidal, and D. Cruz, "Usos y problemas de las APIs web en la República Argentina," No. 44, pp. 79–97. [Online]. https://rtyc.utn.edu.ar/index.php/rtyc/article/view/913
[43] A. Machini and S. Casas, "An empirical study on web API usability: The consumer-developer perspective," *Brazilian Journal of Technology*, Vol. 7, No. 4, 2024, p. e74474.
[44] K. Peffers, T. Tuunanen, M.A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of Management Information Systems*, Vol. 24, No. 3, 2007, pp. 45–77.
[45] B. Kitchenham and S.L. Pfleeger, "Principles of survey research part 6: Data analysis," *ACM SIGSOFT Software Engineering Notes*, Vol. 28, No. 2, 2003, pp. 24–27.
[46] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell et al., *Experimentation in Software Engineering*. Springer Berlin Heidelberg. [Online]. http://link.springer.com/10.1007/978-3-642-29044-2
[47] F.J. Fowler Jr, *Survey research methods*. Sage Publications, 2013.
[48] M. Kasunic, "Designing an effective survey," *Carnegie Mellon University*, 2005.

[49] RStudio, PBC, *RStudio: Integrated Development Environment for R*, 2025, version 2025.05.1+513. [Online]. https://posit.co/download/rstudio-desktop/
[50] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2024, version 4.5.0. [Online]. https://www.R-project.org/
[51] H. Wickham, *ggplot2: Elegant Graphics for Data Analysis*, 2023, version 3.5.2. [Online]. https://ggplot2.tidyverse.org
[52] R.C. Schmidt, "Managing delphi surveys using nonparametric statistical techniques," *Decision Sciences*, Vol. 28, No. 3, 1997, pp. 763–774. [Online]. https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-5915.1997.tb01330.x
[53] R.A. Fisher, "Statistical methods for research workers," in *Breakthroughs in statistics: Methodology and distribution*. Springer, 1970, pp. 66–70.
[54] J.W. Tukey, "Comparing individual means in the analysis of variance," *Biometrics*, 1949, pp. 99–114.
[55] M.G. Kendall and B.B. Smith, "The problem of m rankings," *The Annals of Mathematical Statistics*, Vol. 10, No. 3, 1939, pp. 275–287.
[56] L.J. Cronbach, "Coefficient alpha and the internal structure of tests," *Psychometrika*, Vol. 16, No. 3, 1951, pp. 297–334.
[57] J.L. Fleiss, "Measuring nominal scale agreement among many raters," *Psychological Bulletin*, Vol. 76, No. 5, 1971, pp. 378–382.
[58] P. Becker, L. Olsina, and M.F. Papa, "Especificación de métricas de mantenibilidad para mejorar código Java: Caso aplicado en un curso avanzado de ingeniería en sistemas," in *Actas del XXI Congreso Nacional de Ingeniería Informática/Sistemas de Información (CoNaIISI), Tucumán, Argentina*, 2023, pp. 1–15.

Appendix A. Preliminary catalog of metrics

This section presents the preliminary catalog of metrics validated in this work, derived from [9].

Category	Metric Code	Metric Name
API Performance (AP)	AP.1.1	Informational Response Rate
API Performance (AP)	AP.2.1	Successful Response Rate
API Performance (AP)	AP.2.2	Total Successful Requests
API Performance (AP)	AP.3.1	Redirection Response Rate
API Performance (AP)	AP.4.1	Requests Throughput
API Performance (AP)	AP.5.1	Average inbound data
API Performance (AP)	AP.5.2	Maximum inbound data
API Performance (AP)	AP.5.3	Minimum inbound data
API Performance (AP)	AP.6.1	Average outbound data
API Performance (AP)	AP.6.2	Maximum outbound data
API Performance (AP)	AP.6.3	Minimum outbound data
API Utilization (AU)	AU.1.1	Active inbound data
API Utilization (AU)	AU.1.2	Active outbound data
Caching (CA)	CA.1.1	Total Cache Hits
Caching (CA)	CA.1.2	Cache Hit Rate
Caching (CA)	CA.2.1	Total Cache Misses
Caching (CA)	CA.2.2	Cache Miss Rate
Errors (ER)	ER.1.1	Total Error Count
Errors (ER)	ER.1.2	Overall Error Rate
Errors (ER)	ER.2.1	Client Error Count
Errors (ER)	ER.2.2	Client Error Rate
Errors (ER)	ER.3.1	Server Error Count

Category	Metric Code	Metric Name
Errors (ER)	ER.3.2	Server Error Rate
Errors (ER)	ER.4.1	Total Other Errors
Errors (ER)	ER.4.2	Exception Count
Errors (ER)	ER.4.3	Total Authentication Errors
Errors (ER)	ER.4.4	Total Target Connection Errors
Latency and Response Time (RT)	RT.1.1	Internal Processing Time (IPT)
Latency and Response Time (RT)	RT.1.2	IPT (90th Percentile)
Latency and Response Time (RT)	RT.1.3	IPT (95th Percentile)
Latency and Response Time (RT)	RT.1.4	IPT (99th Percentile)
Latency and Response Time (RT)	RT.2.1	Total Response Time (TRT)
Latency and Response Time (RT)	RT.2.2	Maximum Response Time
Latency and Response Time (RT)	RT.2.3	Minimum Response Time.
Latency and Response Time (RT)	RT.2.4	Standard Deviation of Response Time
Latency and Response Time (RT)	RT.2.5	TRT (50th Percentile)
Latency and Response Time (RT)	RT.2.6	TRT (90th Percentile)
Latency and Response Time (RT)	RT.2.7	TRT (95th Percentile)
Latency and Response Time (RT)	RT.2.8	TRT (99th Percentile)
Latency and Response Time (RT)	RT.3.1	Upstream Response Time (UpRT)
Latency and Response Time (RT)	RT.3.2	UpRT (90th Percentile)
Latency and Response Time (RT)	RT.3.3	UpRT (95th Percentile)
Resource Utilization (RU)	RU.1.1	CPU Utilization Percentage
Resource Utilization (RU)	RU.2.1	Disk Utilization Percentage
Resource Utilization (RU)	RU.3.1	Memory Usage in the Last Hour
Resource Utilization (RU)	RU.3.2	Maximum Memory Utilization
Resource Utilization (RU)	RU.3.3	Memory Usage by Node
System Health (SH)	SH.1.1	Database Reachability Status
System Health (SH)	SH.2.1	Upstream Health Status
System Health (SH)	SH.3.1	Active Reading Connections
System Health (SH)	SH.3.2	Idle waiting connections
System Health (SH)	SH.3.3	Active writing connections
Traffic Monitoring (TM)	TM.1.1	Total APIs
Traffic Monitoring (TM)	TM.1.2	Total number of Products
Traffic Monitoring (TM)	TM.1.3	Total API Requests
Traffic Monitoring (TM)	TM.2.1	Total Data transferred
Traffic Monitoring (TM)	TM.2.2	Total Inbound data
Traffic Monitoring (TM)	TM.2.3	Total outbound data
Traffic Monitoring (TM)	TM.3.1	Throttling Errors

Appendix B. Descriptive statistics of metrics' accuracy ratings

metric	mean	median	sd	min	max
TM.1.1	4.142857	4	1.0690450	2	5
TM.1.2	4.142857	4	0.6900656	3	5
TM.1.3	4.428571	4	0.5345225	4	5
TM.2.1	4.000000	4	1.0000000	2	5
TM.2.2	4.142857	4	0.6900656	3	5
TM.2.3	4.142857	4	0.6900656	3	5
TM.3.1	4.000000	4	1.0000000	2	5
SH.1.1	3.285714	3	0.7559289	2	4
SH.2.1	3.000000	3	1.0000000	2	4
SH.3.1	3.000000	3	1.0000000	2	4
SH.3.2	3.000000	3	1.0000000	2	4

metric	mean	median	sd	min	max
SH.3.3	3.142857	3	1.2149858	2	5
CA.1.1	3.714286	4	0.9511897	2	5
CA.1.2	4.000000	4	1.0000000	2	5
CA.2.1	3.714286	4	0.9511897	2	5
CA.2.2	3.857143	4	1.0690450	2	5
AU.1.1	3.714286	4	0.9511897	2	5
AU.1.2	3.714286	4	0.9511897	2	5
RT.1.1	3.571429	4	1.1338934	2	5
RT.1.2	2.571429	2	1.1338934	1	4
RT.1.3	2.571429	2	1.1338934	1	4
RT.1.4	2.571429	2	1.1338934	1	4
RT.2.1	3.857143	4	1.0690450	2	5
RT.2.2	3.857143	4	0.8997354	2	5
RT.2.3	3.571429	4	1.1338934	2	5
RT.2.4	4.000000	4	1.0000000	2	5
RT.2.5	2.857143	3	1.2149858	1	4
RT.2.6	2.857143	3	1.2149858	1	4
RT.2.7	2.857143	3	1.2149858	1	4
RT.2.8	2.857143	3	1.2149858	1	4
RT.3.1	3.571429	4	1.2724180	2	5
RT.3.2	3.000000	3	1.1547005	2	5
RT.3.3	3.000000	3	1.1547005	2	5
RU.1.1	3.714286	4	1.2535663	2	5
RU.2.1	3.428571	4	1.3972763	2	5
RU.3.1	3.857143	4	0.8997354	2	5
RU.3.2	3.714286	4	0.9511897	2	5
RU.3.3	3.857143	4	0.8997354	2	5
AP.1.1	4.000000	4	1.1547005	2	5
AP.2.1	4.285714	4	0.7559289	3	5
AP.2.2	4.142857	4	0.6900656	3	5
AP.3.1	3.857143	4	1.0690450	2	5
AP.4.1	4.142857	4	0.6900656	3	5
AP.5.1	3.428571	4	1.1338934	2	5
AP.5.2	3.428571	4	1.1338934	2	5
AP.5.3	3.285714	4	1.2535663	2	5
AP.6.1	3.285714	4	1.2535663	2	5
AP.6.2	3.428571	4	1.1338934	2	5
AP.6.3	3.571429	4	1.1338934	2	5
ER.1.1	4.428571	4	0.5345225	4	5
ER.1.2	4.571429	5	0.5345225	4	5
ER.2.1	4.571429	5	0.5345225	4	5
ER.2.2	4.000000	4	1.1547005	2	5
ER.3.1	4.142857	5	1.2149858	2	5
ER.3.2	4.571429	5	0.5345225	4	5
ER.4.1	3.714286	4	1.2535663	2	5
ER.4.2	4.142857	4	0.8997354	3	5
ER.4.3	4.000000	4	1.0000000	2	5
ER.4.4	3.857143	4	1.0690450	2	5

Appendix C. Descriptive statistics of metrics' relevance ratings

metric	mean	median	sd	min	max
TM.1.1	4.285714	4	0.7559289	3	5
TM.1.2	4.285714	4	0.7559289	3	5
TM.1.3	4.571429	5	0.5345225	4	5
TM.2.1	4.285714	5	1.1126973	2	5
TM.2.2	4.285714	5	1.1126973	2	5
TM.2.3	4.285714	5	1.1126973	2	5
TM.3.1	4.571429	5	0.7867958	3	5
SH.1.1	3.571429	3	0.7867958	3	5
SH.2.1	4.428571	4	0.5345225	4	5
SH.3.1	3.714286	3	0.9511897	3	5
SH.3.2	4.000000	4	0.8164966	3	5
SH.3.3	4.142857	4	0.8997354	3	5
CA.1.1	3.571429	4	0.9759001	2	5
CA.1.2	3.857143	4	1.0690450	2	5
CA.2.1	3.857143	4	0.8997354	3	5
CA.2.2	3.857143	4	0.8997354	3	5
AU.1.1	3.428571	4	1.1338934	2	5
AU.1.2	3.428571	4	1.1338934	2	5
RT.1.1	4.285714	4	0.7559289	3	5
RT.1.2	3.571429	3	0.7867958	3	5
RT.1.3	3.571429	3	0.7867958	3	5
RT.1.4	3.571429	3	0.7867958	3	5
RT.2.1	4.571429	5	0.5345225	4	5
RT.2.2	4.285714	4	0.7559289	3	5
RT.2.3	4.142857	4	1.0690450	2	5
RT.2.4	4.285714	4	0.7559289	3	5
RT.2.5	3.857143	4	0.8997354	3	5
RT.2.6	3.857143	4	0.8997354	3	5
RT.2.7	3.857143	4	0.8997354	3	5
RT.2.8	3.857143	4	0.8997354	3	5
RT.3.1	4.000000	4	1.1547005	2	5
RT.3.2	3.714286	4	1.1126973	2	5
RT.3.3	3.714286	4	1.1126973	2	5
RU.1.1	4.142857	4	1.0690450	2	5
RU.2.1	3.714286	4	1.2535663	2	5
RU.3.1	4.285714	4	0.4879500	4	5
RU.3.2	4.000000	4	0.8164966	3	5
RU.3.3	4.142857	4	0.3779645	4	5
AP.1.1	3.857143	4	1.0690450	2	5
AP.2.1	4.000000	4	0.5773503	3	5
AP.2.2	4.000000	4	0.8164966	3	5
AP.3.1	3.857143	4	1.0690450	2	5
AP.4.1	4.285714	4	0.7559289	3	5
AP.5.1	3.714286	4	1.1126973	2	5
AP.5.2	3.571429	4	0.9759001	2	5
AP.5.3	3.571429	4	1.2724180	2	5
AP.6.1	3.571429	4	1.2724180	2	5
AP.6.2	3.714286	4	1.1126973	2	5
AP.6.3	3.857143	4	1.0690450	2	5
ER.1.1	4.428571	4	0.5345225	4	5
ER.1.2	4.571429	5	0.5345225	4	5
ER.2.1	4.428571	4	0.5345225	4	5
ER.2.2	4.000000	4	1.1547005	2	5

metric	mean	median	sd	min	max
ER.3.1	4.000000	4	1.1547005	2	5
ER.3.2	4.571429	5	0.5345225	4	5
ER.4.1	4.000000	4	1.0000000	2	5
ER.4.2	4.000000	4	0.8164966	3	5
ER.4.3	4.428571	5	0.7867958	3	5
ER.4.4	4.142857	4	0.8997354	3	5

Appendix D. Tukey HSD pairwise comparisons

Expert pair	diff	lwr	upr	p adj
E2-E1	−0.28813559	−0.68469491	0.1084237	0.3237969
E3-E1	0.22033898	−0.17622033	0.6168983	0.6521410
E4-E1	0.30508475	−0.09147457	0.7016441	0.2564151
E5-E1	−0.20338983	−0.59994915	0.1931695	0.7328451
E6-E1	0.96610169	0.56954238	1.3626610	0.0000000
E7-E1	1.30508475	0.90852543	1.7016441	0.0000000
E3-E2	0.50847458	0.11191526	0.9050339	0.0031533
E4-E2	0.59322034	0.19666102	0.9897797	0.0002408
E5-E2	0.08474576	−0.31181355	0.4813051	0.9956855
E6-E2	1.25423729	0.85767797	1.6507966	0.0000000
E7-E2	1.59322034	1.19666102	1.9897797	0.0000000
E4-E3	0.08474576	−0.31181355	0.4813051	0.9956855
E5-E3	−0.42372881	−0.82028813	−0.0271695	0.0274732
E6-E3	0.74576271	0.34920340	1.1423220	0.0000010
E7-E3	1.08474576	0.68818645	1.4813051	0.0000000
E5-E4	−0.50847458	−0.90503389	−0.1119153	0.0031533
E6-E4	0.66101695	0.26445763	1.0575763	0.0000236
E7-E4	1.00000000	0.60344068	1.3965593	0.0000000
E6-E5	1.16949153	0.77293221	1.5660508	0.0000000
E7-E5	1.50847458	1.11191526	1.9050339	0.0000000
E7-E6	0.33898305	−0.05757627	0.7355424	0.1503445

Authors and affiliations

Eder dos Santos
e-mail: esantos@uarg.unpa.edu.ar
ORCID: https://orcid.org/0000-0001-6729-0303
Instituto de Tecnologia Aplicada, Universidad Nacional de la Patagonia Austral, Argentina

Sandra Casas
e-mail: sicasas@uarg.unpa.edu.ar
ORCID: https://orcid.org/0000-0002-8289-6132
Instituto de Tecnologia Aplicada, Universidad Nacional de la Patagonia Austral, Argentina

e-Informatica Software Engineering Journal
Volume 20, Issue 1, pages: 1–36
DOI: 10.37190/e-Inf260107
BibTeX

Bridging Participatory Co-Design and Model-Driven Engineering for Community-Based Web-3D Framework Development: Method and Validation

Ratna Wardani*, Masduki Zakaria, Tika Novita Sari, Dian Novian
*Corresponding author: ratna@uny.ac.id

Article info

Keywords:
participatory software engineering
model-driven engineering (MDE)
co-design methodology
requirement traceability
Web-3D framework development
sustainable tourism platform

Submitted: 04 Apr. 2026
Revised: 6 May 2026
Accepted: 6 May 2026
Available online: 12 May 2026

Abstract

Context: Sustainable tourism platforms require methodologies balancing stakeholder inclusivity with technical rigor. Existing approaches either emphasize participatory design at the cost of formalization or concentrate on technical precision while neglecting community voices, creating a critical gap in developing Web-3D frameworks that honor local knowledge while maintaining engineering excellence.
Objective: We present a replicable methodology integrating Design Thinking, Software Domain Architecture (SDA), and Model-Driven Engineering (MDE) to bridge participatory engagement with formal software specification, validated through DIVEXPLORE-3D, a Web-3D platform for sustainable marine tourism with 54 stakeholders in Gorontalo, Indonesia.
Method: The methodology comprises five integrated phases: (1) participatory workshops with 54 stakeholders across three groups (tourists, community members, operators) using empathy mapping; (2) requirements formalization through SDA into four architectural layers; (3) model transformation from PIMs to PSMs via MDE; (4) bidirectional validation through a RTM; and (5) pilot deployment with technical and educational evaluation.
Results: The validation of the methodology resulted in four evidence streams: (1) the four-layer architecture achieved 96% coverage of requirements across 68 specifications; (2) an expert evaluation ($n = 8$) rated the sustainability adequacy of the methodology's specifications at 4.6/5.0; (3) a pilot deployment confirmed technical feasibility (mean latency 73 ms, 97.2% uptime); and (4) educational testing ($n = 67$) showed significant gains in conservation knowledge, with a Cohen's d of 1.52 ($p < 0.001$). Additionally, community content moderation successfully published 78% of content without requiring expert modification.
Conclusions: This study presents a replicable methodology integrating participatory co-design with formal software engineering. Instead of being opposing paradigms, they are complementary processes when mediated by structured traceability mechanisms. The DIVEXPLORE-3D framework promotes sustainable digital tourism through a modular, community-driven platform balancing ecological conservation, cultural preservation, and technical scalability.

© 2026 The Authors. Published by Wrocław University of Science and Technology Publishing House.
This is an open access article under the CC BY-SA 4.0 license international.

1. Introduction

The integration of immersive technologies in tourism has shifted the paradigm from passive consumption to active engagement, enabling visitors to interact with destinations through dynamic digital environments [1]. Web-3D technologies, utilizing WebGL and browser-based rendering, are particularly promising for democratizing virtual tourism experiences, as they do not require specialized hardware [2]. In marine ecotourism contexts, these technologies offer dual advantages: enhancing visitor education about fragile ecosystems while potentially reducing environmental impact through virtual exploration options [3].

However, the development of these platforms encounters a significant methodological gap. While substantial research exists on immersive technologies in tourism [4, 5] and digital transformation in hospitality [6], there is a notable lack of systematic software engineering approaches ensuring scalability, maintainability, and stakeholder alignment. This gap is especially pronounced in community-based tourism, where platforms must serve tourists, local communities, and operators while honoring cultural authenticity and fostering sustainability [7]. The absence of methodologies linking participatory design with formal specifications often results in technologies that overlook community voices and fail to integrate local knowledge systems [8].

Many existing digital tourism platforms maintain extractive relationships, where external developers create systems for communities rather than with them [8]. This produces technologies that misrepresent local knowledge, marginalize community voices, and provide minimal economic benefits to residents. While platform economies can enhance accessibility and economic participation [9], navigating them effectively requires understanding their underlying structures, including concerns about market concentration, algorithmic fairness, and equitable distribution of benefits [10]. Platform evolution research demonstrates that successful digital ecosystems require careful balance between platform architecture, governance mechanisms, and stakeholder dynamics [9].

Recent literature highlights the vital role of digital platforms in advancing the Sustainable Development Goals (SDGs) in tourism [11]. Digital transformation through Industry 4.0 technologies (AI, virtual reality, blockchain) is reshaping tourism operations and destination sustainability management [12]. However, integrating these technologies requires methodologies systematically connecting participatory stakeholder engagement with rigorous software specification processes. This research addresses that gap.

This paper introduces DIVEXPLORE-3D, a Web-3D framework for sustainable marine tourism developed through systematic integration of Design Thinking, Software Domain Architecture (SDA), and Model-Driven Engineering (MDE). Situated in Gorontalo, Indonesia, a coastal region renowned for its marine biodiversity but facing challenges in tourism development, the framework was collaboratively designed with 54 stakeholders (tourists, local communities, tourism operators) over six months. The primary contribution of this paper is methodological: we demonstrate how Design Thinking, SDA, and MDE can be systematically integrated into a replicable engineering process for community-based Web-3D frameworks. DIVEXPLORE-3D, a Web-3D platform for sustainable marine tourism in Gorontalo, Indonesia, serves as the case study through which this methodology is instantiated, demonstrated, and empirically validated.

The research addresses three specific objectives:

1. To demonstrate a replicable methodology integrating user-centered co-design with formal software engineering, systematically transforming stakeholder needs into implementation-ready architecture specifications.

2. To implement the methodology through the DIVEXPLORE-3D framework, illustrating how its four-layer architecture, requirement specifications, and traceability mechanisms systematically arise from the integrated use of Design Thinking, SDA, and MDE.
3. To evaluate how this integrated approach improves sustainable software engineering practices for community-based tourism systems.

This research offers three unique contributions:

1. **Methodological innovation:** We present a systematic methodology that integrates Design Thinking's participatory strengths with SDA's specification rigor and MDE's transformational formalism for community-based Web-3D platform development. While participatory RE has been integrated with formal methods in adjacent domains, including socio-technical systems [13], participatory modeling [14], and contextual design approaches [15], these integrations do not address the specific combination of challenges present in Web-3D community-based tourism: real-time 3D rendering requirements, community content sovereignty, bidirectional traceability from empathy-based elicitation to platform-specific model transformation, and multi-stakeholder governance. Our methodology provides explicit phase-to-artifact mappings that demonstrate how each Design Thinking phase connects to its corresponding engineering artifacts, producing a replicable blueprint that preserves stakeholder voice throughout the development lifecycle in this specific context.
2. **Methodological instantiation:** We illustrate the effectiveness of our methodology through DIVEXPLORE-3D, a validated four-layer architecture resulting from systematically applying the integrated Design Thinking - SDA-MDE process. This architecture achieves 96% requirement coverage and 97.2% uptime, providing proof-of-concept evidence of its technical feasibility in this case study (see Section 5.6, Level 2 validation). However, whether this architectural instantiation offers transferable design propositions for other community-based Web-3D contexts remains an open empirical question that requires validation through comparative case studies (see Section 6, Direction 2).
3. **Empirical evidence:** Over a six-month period, we engaged 54 stakeholders and 8 domain experts to validate our findings. Our study provides quantitative evidence that participatory software engineering produces architectures with specifications rated 4.6 out of 5.0 for sustainability adequacy by domain experts. This finding confirms that the methodology effectively integrates sustainability objectives into the engineering process. However, to assess operational sustainability outcomes, longitudinal validation is needed, which goes beyond the scope of this study. Pilot testing demonstrates a measurable educational impact (Cohen's $d = 1.52$ for conservation knowledge gains) and effective community knowledge sovereignty (78% of content published without external modification).

This paper is structured as follows: Section 2 reviews related work on Web-3D tourism applications, co-design methodologies, and software engineering for sustainable systems. Section 3 describes our integrated methodology, which combines Design Thinking, Software Domain Architecture (SDA), and Model-Driven Engineering (MDE). Section 4 presents the result, including stakeholder requirements, framework architecture, use-case models, model transformations, and validation outcomes. Section 5 discusses implications for sustainable software engineering and tourism platform development. Section 6 concludes with limitations and future research directions.

2. Related work

2.1. Web-3D and immersive technologies in tourism

The adoption of Web-3D technologies and virtual reality in tourism has accelerated significantly, particularly following the COVID-19 pandemic's catalytic effect on digital transformation [16]. Virtual tourism emerged as a viable alternative during lockdowns, demonstrating potential for reducing environmental impacts while preserving experiential value [17]. Research shows that immersive digital experiences enhance destination marketing, improve pre-visit decision-making, and support conservation education [18].

Browser-based 3D technologies offer distinct advantages over dedicated VR applications by eliminating hardware barriers and ensuring cross-device accessibility. Recent advancements in immersive storytelling techniques show that VR environments can provide engaging narrative experiences across platforms, from desktops to fully immersive settings, creating new opportunities for narrative-driven tourism applications [19]. Studies indicate that WebGL platforms achieve educational outcomes comparable to those of VR headsets, while reaching a broader audience, which is crucial for sustainable tourism development in resource-constrained regions [20]. The metaverse concept has further expanded virtual tourism possibilities, enabling immersive destination previews, cultural heritage experiences, and environmental education [21, 22].

However, current literature on Web-3D tourism applications emphasizes user experience and technological capabilities rather than the software engineering methodologies required for maintainable, scalable systems [23]. While research by Buhalis [24] and Go et al. [25] highlights the transformative potential of metaverse technologies in tourism, it provides limited guidance on systematic development processes that integrate stakeholder participation with formal specification methods.

2.2. Digital platforms and sustainable tourism development

Digital tourism platforms have fundamentally transformed industry business models and stakeholder relationships [9]. Research by Zeqiri et al. [11] demonstrates that well-designed and governed platforms can advance the Sustainable Development Goals (SDGs) by facilitating rural revitalization (SDG 8), promoting responsible consumption (SDG 12), and supporting climate action (SDG 13) through features such as carbon footprint tracking and eco-friendly accommodation recommendations.

Despite these benefits, platform-driven tourism faces significant sustainability challenges that are often overlooked in discussions of technological optimism. Market concentration among leading Online Travel Agencies (OTAs) creates economic dependencies for smaller operators, potentially exacerbating inequalities (SDG 10) [26]. While AI-driven automation and dynamic pricing improve operational efficiency, they raise concerns about workforce transformation, algorithmic bias in pricing and recommendations, and data governance issues regarding privacy and user-generated content ownership [27, 28]. These tensions between efficiency and equity underscore the need for development methodologies that explicitly balance technological advancement with social sustainability, an area that is inadequately addressed in existing platform design frameworks.

The intersection of Industry 4.0 technologies and sustainable tourism reveals complex socio-technical dynamics. Innovations such as AI, blockchain, and IoT offer opportunities

for resource optimization, enhanced transparency in sustainability claims, and improved visitor experiences [12, 29]. However, substantial implementation costs and technical expertise requirements create digital divides between well-resourced operators and small community-based enterprises [30]. This disparity raises critical questions about who benefits from tourism's digital transformation and whose perspectives shape its trajectory. Addressing these questions requires methodological approaches combining participatory stakeholder engagement with rigorous system design, moving beyond mere technical implementation.

2.3. Co-design and participatory approaches in tourism technology

Co-design methodologies are crucial for ensuring that tourism technologies align with the diverse values of stakeholders and local contexts [31]. Design Thinking, which focuses on empathy, ideation, and iterative prototyping, offers structured approaches to understanding community needs and transforming them into design requirements. The theoretical foundation of design thinking distinguishes it from traditional analytical approaches through its iterative cycles of problem formulation and solution exploration [32]. Research shows that Design Thinking workshops involving local stakeholders produce tourism innovations that better reflect community values compared to approaches led solely by experts. However, measuring this alignment quantitatively remains a methodologically challenging task [33].

In digital ecotourism, community participation is crucial for content authenticity and long-term platform sustainability. Studies show that locally curated digital tourism platforms maintain higher numbers of active local contributors over time compared to externally managed systems [7]. However, the contributing factors, including technical infrastructure, governance structures, and training programs, remain underexplored in the existing literature, which limits the replicability of participatory approaches across diverse tourism contexts.

The literature identifies a persistent "valley of death" between participatory workshops and the development of implementable systems [34]. Stakeholder requirements from interviews and focus groups often become diluted or misinterpreted during technical team handoffs. This disconnect arises from three fundamental challenges: (1) incompatible vocabularies between domain experts and software engineers create communication barriers leading to requirement misunderstandings; (2) lacking formal traceability mechanisms prevents systematic validation, making it difficult to prioritize community needs throughout development; and (3) absent shared artifacts prevent non-technical stakeholders from verifying technical specifications, resulting in late-stage misalignment discoveries [35].

Recent attempts to bridge participatory design with software specification in tourism have shown limited success. Visual modeling approaches using low-code platforms lower technical barriers but often lack expressiveness for complex system requirements. Conversely, formal specification languages offer rigor but prove inaccessible to community stakeholders, undermining participatory engagement goals. This tension between accessibility and precision represents a fundamental methodological challenge that current co-design frameworks have yet to effectively resolve. Requirements engineering for sustainability demonstrates that systematic elicitation and specification of non-functional requirements across social, economic, environmental, and technical dimensions is essential for sustainable system development [36].

Several hybrid participatory-engineering frameworks offer partial solutions for bridging stakeholder engagement with technical specifications. Participatory and collaborative modelling [14] facilitates cooperation among technical experts; however, it fails to systematically

include non-technical stakeholders throughout the entire modeling lifecycle and does not produce actionable engineering models from qualitative community data. Socio-Technical Requirements Engineering [13] explicitly integrates social and technical dimensions into system development, but is constrained by a disconnect between its analysis methods and formal engineering processes, particularly in the absence of model transformation mechanisms for platform-specific implementation. Contextual Design [15] provides valuable ethnographic insights through contextual inquiries and affinity diagrams, but as a front-end design process, it does not extend to bidirectional requirements traceability or formal model transformation pipelines for technical implementation. In community-based Web-3D tourism contexts, where cultural sovereignty, ecological education, and technical scalability must be addressed simultaneously, none of these frameworks individually provides the comprehensive integration from empathy-based elicitation to platform-specific model transformation that our methodology seeks to achieve.

Studies emphasize the significance of community participation in digital ecotourism platforms, showing that community-curated content produces greater authenticity and sustainability outcomes than externally managed systems [7]. However, these studies typically overlook the software engineering challenges involved in implementing community governance mechanisms, content moderation workflows, and scalable architectures that support distributed authorship.

2.4. Software engineering for complex tourism systems

Software Domain Architecture (SDA) provides structured methods for decomposing complex systems into domain-specific components, enhancing modularity and traceability [37]. Layered architectural patterns have proven effective in tourism systems that require the separation of concerns across presentation, business logic, data management, and external integrations [38]. However, current SDA applications in tourism primarily focus on booking systems and destination management platforms, with less emphasis on immersive Web-3D environments that have unique requirements for real-time rendering, content management, and multi-stakeholder interaction.

Model-Driven Engineering (MDE) offers formal mechanisms connecting conceptual designs with platform-specific implementations through automated transformation rules [39]. By maintaining Platform-Independent Models (PIMs) alongside Platform-Specific Models (PSMs), MDE approaches enhance long-term maintainability and facilitate platform evolution, which is particularly important for tourism systems that require multi-year operational lifespans [40]. Research indicates MDE reduces specification ambiguity and accelerates development cycles for web applications [41], yet its application to Web-3D tourism platforms remains underexplored.

Integrating participatory design with formal software engineering methods represents an emerging research frontier. Recent studies emphasize making specification artifacts accessible to non-technical stakeholders while maintaining mathematical rigor [42]. However, concrete methodologies demonstrating this integration within tourism contexts are lacking, particularly for immersive Web-3D platforms that require real-time rendering, distributed content creation, and multi-stakeholder governance. This gap between participatory ideals and engineering practice motivates the methodological integration presented in this paper.

2.5. Research gap and contribution

The literature reveals three critical gaps:

First, while Web-3D technologies demonstrate educational effectiveness and accessibility benefits [17, 20], and although participatory Requirements Engineering (RE) has been integrated with formal methods in related fields like healthcare informatics, smart city platforms, and environmental management systems [13, 14], these integrations do not directly translate to Web-3D community-based tourism. Three contextual factors create distinct methodological requirements: (1) real-time 3D rendering introduces platform-specific performance constraints that necessitate formal PSM (Process Specification Model) transformation beyond what socio-technical RE approaches currently address; (2) community content sovereignty demands that local communities maintain editorial control over cultural narratives, requiring governance mechanisms that are lacking in adjacent domain integrations that focus on expert-driven specifications; and (3) the simultaneous needs for ecological education, cultural preservation, and technical scalability require a unified traceability mechanism that connects empathy-based elicitation to implementation-ready specifications, which no existing integration currently provides end-to-end.

Second, despite the demonstrated benefits of co-design in enhancing community satisfaction and cultural authenticity [7,31], and the availability of hybrid participatory-engineering frameworks, including participatory modeling [14], Socio-Technical RE [13], and Contextual Design [15], these approaches address only partial aspects of the integration challenge. They lack systematic mechanisms to simultaneously achieve community-grounded elicitation, bidirectional traceability, and the transformation of formal models into platform-specific implementations. This is particularly important in community-based Web-3D tourism contexts, where cultural sovereignty and ecological education are essential alongside technical scalability.

Third, although SDA and MDE provide robust specification and transformation mechanisms [37, 39], they have not been systematically integrated with participatory methods, limiting their application in community-based contexts requiring stakeholder involvement throughout the development lifecycle.

This study addresses these gaps by demonstrating how Design Thinking, Software Domain Architecture (SDA), and Model-Driven Engineering (MDE) can be synergistically integrated into a methodology that is participatory, rigorous, and replicable. Unlike previous research treating these approaches separately, we illustrate how each Design Thinking phase maps to specific engineering methods: empathy and problem definition correspond to SDA domain analysis, prototyping aligns with MDE model creation, and testing validates traceability. This integration yields Web-3D architectures that are technically sound, stakeholder-validated, and implementation-ready, advancing both sustainable tourism and software engineering research.

3. Method

3.1. Research design

This study employs a design science research approach [43], positioning the DIVEXPLORE-3D framework and its integrated methodology as both research outcomes and contributions

to knowledge. Design science is appropriate when research aims to create and evaluate artifacts solving identified real-world problems [44]. Our epistemological stance is pragmatist, emphasizing the practical utility of integrating diverse methodological traditions such as participatory design and formal specification based on their effectiveness rather than strict adherence to particular philosophical frameworks [45].

The methodology combines Design Thinking with formal software engineering techniques, specifically Software Domain Architecture (SDA) and Model-Driven Engineering (MDE). This integration preserves stakeholder input while ensuring rigorous specifications (Figure 1). Design Thinking was selected over other participatory approaches such as Participatory Action Research and Community-Based Participatory Research for three reasons: (1) its explicit phase structure (empathize, define, ideate, prototype, test) provides clear integration points for engineering methods; (2) its focus on rapid prototyping and iteration aligns with agile software development practices common in Web-3D platform development; and (3) its proven effectiveness in service design and tourism innovation contexts enhances stakeholder communication and buy-in [32]. SDA provides a structured framework for classifying and organizing requirements into domain-oriented architectural components [37], while MDE facilitates the transformation of requirements into formalized models suitable for specification and implementation [39].

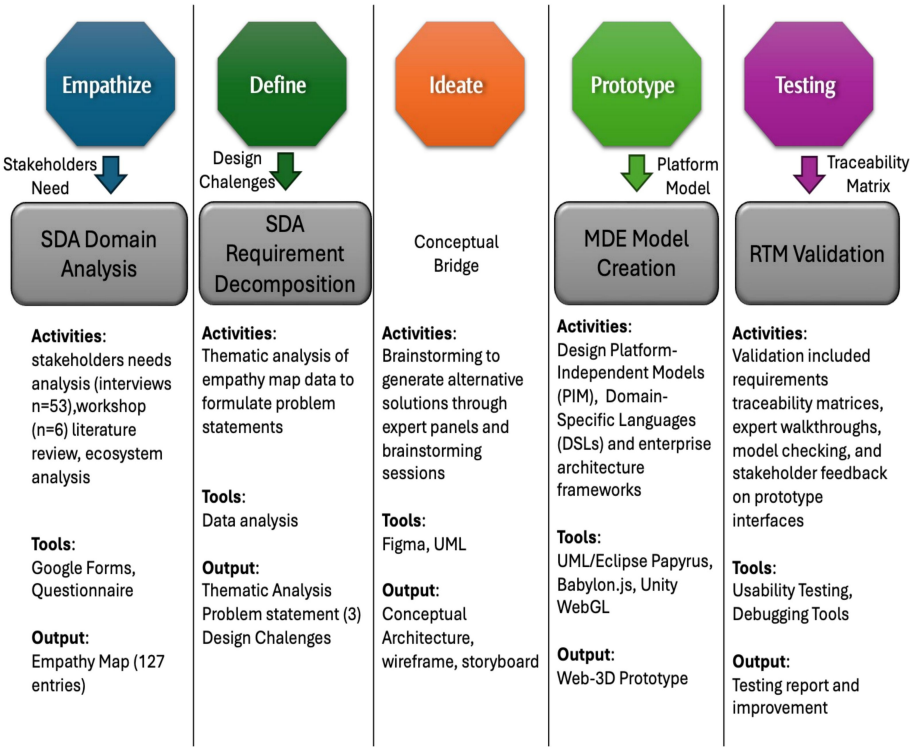


Figure 1. Integrated research design mapping

The framework illustrates systematic integration of participatory design phases (Empathize, Define, Ideate, Prototype, Test) with formal software engineering methods (Software Domain Architecture, Model-Driven Engineering, Requirement Traceability Matrix). Each phase outlines specific activities, tools, and outputs, with arrows indicating transformation flows from stakeholder engagement to technical specifications.

Each Design Thinking phase was systematically aligned with specific engineering methods through clear transformation rules. The Empathize and Define phases corresponded to Software Domain Analysis (SDA), where insights from empathy maps and refined problem statements informed classification of domain-specific requirements using thematic coding and affinity mapping. The Ideate phase served as a conceptual bridge, generating design concepts through stakeholder workshops that informed subsequent modeling decisions without direct formal transformation. The Prototype phase integrated with Model-Driven Engineering (MDE) model creation, transforming preliminary concepts into Platform-Independent Models (PIMs) using UML/SysML notation, then developing Platform-Specific Models (PSMs) for WebGL/Eclipse Papyrus implementation. The Test phase validated the Requirement Traceability Matrix (RTM), ensuring bidirectional traceability from stakeholder needs to architectural components and acceptance criteria.

This integration demonstrates how user-centered insights from Design Thinking can be systematically incorporated with formal specification methods while preserving contextual nuances and stakeholder involvement. We ensured methodological rigor through three validation mechanisms: (1) inter-rater reliability checks during thematic analysis; (2) expert review of requirement specifications; and (3) automated consistency checks of model transformations using Object Constraint Language (OCL) validation tools. While the Ideate phase was primarily conceptual, the Empathize, Define, Prototype, and Test phases maintained explicit traceability to engineering artifacts, highlighting the complementary relationship between creative exploration and formal specification.

3.2. Research context and participants

The research was conducted in Gorontalo, Indonesia, a coastal region in northern Sulawesi known for the world's highest concentration of whale sharks. Gorontalo attracts approximately 12,000 international dive tourists annually, a modest volume enabling meaningful stakeholder engagement. However, the region faces typical challenges of resource-constrained tourism destinations: limited digital infrastructure, developing community organizational capacity for tourism management, and increasing pressure to balance conservation with economic development. This context makes Gorontalo representative of emerging ecotourism destinations in Southeast Asia and the Global South, where digital transformation must occur within significant resource and capacity constraints.

Participant recruitment utilized purposive sampling [14], ensuring representation across three key stakeholder groups in marine tourism ecosystems. Over six months (March to August 2025), 54 participants contributed to data collection: 32 tourists, 17 community members, and 5 tourism operators.

Tourists were recruited through dive operators during peak season (April to June 2025) with inclusion criteria: (1) completion of at least two dives in Gorontalo waters; (2) self-identified interest in marine conservation; and (3) willingness to participate in 60–90 minute workshops.

Community members were recruited through village leadership in three coastal communities (Torosiaje, Olele, and Botu Barani), which are actively involved in tourism. Inclusion criteria required: (1) residence in tourism-affected communities for a minimum of five years; (2) direct or indirect economic involvement in marine tourism; and (3) willingness to share cultural knowledge and narratives. This approach resulted in participants with vested interests in tourism outcomes, but may have underrepresented the voices of community members less connected to existing tourism structures.

Tourism operators represented all five licensed dive centers in Gorontalo throughout the study, ensuring 100% coverage of this stakeholder group. While the small operator sample ($n = 5$) limits statistical generalizability, it encompasses the entire population of locally owned businesses, providing comprehensive coverage of the sector.

3.3. Requirement elicitation through design thinking

Requirement elicitation followed five Design Thinking phases: empathize, define, ideate, prototype, and test.

Empathize phase. Insights were gathered using empathy maps focusing on four areas: Think and Feel, See, Say and Do, and Hear. We conducted structured workshops (six sessions in total, two per stakeholder group, each lasting 3–4 hours and accommodating 8–12 participants per session) and semi-structured interviews (54 individual sessions, each lasting 45–60 minutes). These sessions facilitated both quantitative categorization and qualitative depth regarding stakeholder experiences and expectations. Workshop facilitation utilized visual templates and small-group activities to reduce power imbalances and encourage participation from less vocal stakeholders [31]. We documented 127 empathy map entries through flip-chart recordings and photography, subsequently digitized and coded using NVivo 14 software.

Define phase. Findings were synthesized into clear problem statements and design challenges through thematic analysis. Three primary problem statements emerged: (1) How can we create accessible underwater experiences promoting conservation education without expensive VR hardware? (2) How can we empower local communities to share authentic cultural narratives while maintaining control over their representation? (3) How can we provide tourism operators with actionable insights supporting sustainable business decisions?

Ideation phase. We conducted four workshops (three hours each, 6–8 participants from diverse stakeholder groups) and two expert panel discussions with eight domain specialists (three software architects, two tourism researchers, two marine conservation educators, one cultural heritage specialist), focusing on identified design challenges. Through structured ideation techniques such as SCAMPER (Substitute, Combine, Adapt, Modify, Put to another use, Eliminate, Reverse) and "Crazy 8s" rapid sketching [32]. We generated over 60 conceptual solutions. The consolidation process utilized a multi-criteria voting system in which participants rated each of the 12 candidate concepts based on three criteria: technical feasibility (assessed by software architects), cultural appropriateness (validated by community representatives), and alignment with the Sustainable Development Goals (SDG) targets. A concept was selected if it received positive ratings from a simple majority of participants (more than 50%) across all three criteria. This approach led to an 82% agreement among participants regarding the final selection of the 12 design concepts.

Prototype phase. Preliminary models of the DIVEXPLORE-3D framework demonstrated emerging system structure and functional modules. We created low-fidelity wireframes and interactive mockups, which were reviewed in three iterative cycles with stakeholder representatives, refining the interface designs and interaction flows.

Test phase. Iterative validation integrated feedback from domain experts and stakeholders across three validation cycles (May, June, July 2025). A Requirement Traceability Matrix (RTM), detailed in Section 4.4, linked stakeholder needs from the empathy phase to specific framework layers, functional and non-functional requirements, and acceptance criteria. The RTM served dual purposes: as a validation tool to ensure comprehensive coverage of

requirements, and as a communication artifact to enable non-technical stakeholders to verify that their concerns were addressed in the technical specifications. Final validation sessions were conducted with 12 stakeholder representatives, including 6 community members, 4 tourists, and 2 operators. A structured review protocol was used for this process, during which each of the 68 requirements was presented along with its complete traceability chain, from the original empathy map entry to the proposed acceptance criteria. Each representative individually indicated whether they endorsed a requirement or requested modifications.

A requirement was classified as "agreed" when a simple majority of representatives (i.e., 7 or more out of 12) endorsed it without any changes. Following this protocol, 62 out of the 68 requirements (91%) achieved majority agreement during the initial review. The remaining six requirements were revised iteratively based on stakeholder feedback and were re-presented until consensus was reached prior to their final inclusion in the RTM.

3.4. Domain modeling using software domain architecture (SDA)

Qualitative insights from Design Thinking were systematically organized using Systems Design Architecture (SDA), breaking the system down into specific architectural components tailored to each domain [37]. Functional requirements, including 3D rendering, cultural content management, and immersive scenario-based learning, were categorized into four primary domains: Application, Content Management, Immersive Simulation, and Data and Intelligence.

Non-functional requirements were identified as cross-cutting concerns across multiple domains. These requirements include: (1) Accessibility (NFR-01): Ensure over 90% device compatibility; (2) Performance (NFR-02): Maintain inter-layer latency of less than 100 milliseconds; (3) Cultural Authenticity (NFR-03): Implement community-controlled validation; (4) Data Security (NFR-04): Ensure compliance with GDPR. This domain-oriented approach enables clear traceability between stakeholder needs and framework modules, providing a structured methodology for specification.

3.5. Model transformation using model-driven engineering (MDE)

After conducting the SDA, the requirements were articulated using Platform-Independent Models (PIMs) based on UML 2.5 and SysML diagrams to represent system structure and behavior. These diagrams were created using the Eclipse Papyrus 5.1 modeling tool. The key PIM artifacts included: (1) Class Diagrams: Defining entity relationships across 23 core classes (User, Content, Marine Ecosystem, Analytics, etc.); (2) Activity Diagrams: Specifying workflows for content moderation (7 states) and conservation scenario progression (12 states); and (3) Sequence Diagrams: Detailing inter-layer communication patterns for critical use cases.

Object Constraint Language (OCL) assertions enforce non-functional requirements. For example, the performance constraint NFR-02 is formalized as:

```
context ApplicationLayer::render3DScene()
post: response.latency < 100 and response.frameRate >= 30
```

The PIMs were transformed into Platform-Specific Models (PSMs) through Model-Driven Engineering (MDE) processes using Eclipse ATL (ATLAS Transformation Language) 4.3. This transformation ensured alignment with Web-3D platforms and standards such as

WebGL, Three.js, and RESTful APIs. The transformation rules defined the following mappings: UML Class to JavaScript ES6 Class, Activity State to React Component lifecycle, and Sequence Message to API endpoint specification.

Compliance with Meta-Object Facility (MOF) standards enabled the integration of metadata and semantic validation, ensuring that conceptual models were both rigorous and practical for implementation in specific technical environments.

3.6. Data analysis and validation

Qualitative data from empathy maps and open-ended survey responses were analysed using thematic analysis [46], which identified patterns among different stakeholder groups. Two independent researchers each coded all 127 entries from the empathy maps, serving as the unit of analysis, into thematic categories that corresponded to the four quadrants of the empathy map: Think and Feel, See, Say and Do, and Hear. To assess inter-rater reliability, Krippendorff's α was calculated for the 127 coded entries, yielding an α of 0.82. This indicates substantial agreement between the researchers and exceeds the recommended threshold of $\alpha \geq 0.80$, supporting the validity of the conclusions drawn [45]. Any coding disagreements were resolved through discussion until a consensus was reached.

The resulting 68 requirements comprised 42 functional and 26 non-functional requirements, validated through two expert review workshops attended by eight domain specialists (three software architects, two tourism researchers, two marine conservation educators, one cultural heritage specialist). Experts rated requirements on 5-point Likert scales for Completeness, Measurability, and Stakeholder Alignment.

The RTM enabled traceability by mapping empathy-based needs to framework layers, requirements, and acceptance criteria. Final validation sessions with 12 stakeholders achieved 91% agreement on requirement specifications, providing confidence that the framework aligns with community needs and is technically feasible.

4. Case study instantiation: DIVEXPLORE-3D

The integrated Design Thinking-SDA-MDE methodology is instantiated through DIV--EXPLORE-3D, a community-based Web-3D framework designed for sustainable marine tourism developed in Gorontalo, Indonesia. This case study is organized into four methodological phases: (1) Stakeholder requirement elicitation (Section 4.1); (2) Case study architectural instantiation (Section 4.2); (3) Use-case modelling and domain decomposition (Section 4.3); and (4) Requirements Traceability Matrix (Section 4.4). Each subsection illustrates how the methodology's components produce tangible engineering artifacts, providing empirical evidence of its effectiveness.

4.1. Stakeholder requirements from co-design

The empathize phase involved structured empathy mapping workshops conducted across three stakeholder groups over a four-week period (March-April 2025). Thematic analysis of 127 empathy map entries revealed distinct yet overlapping priorities among tourists, local communities, and tourism operators.

Tourist perspectives ($n = 32$). Tourists expressed strong interest in experiential learning beyond traditional diving activities. Key themes included: (1) desire for authentic

cultural narratives integrated into marine exploration (78% of respondents, with participants emphasizing the value of locally-driven storytelling compared to standardized commercial tourism experiences); (2) concerns about environmental impact of visits (68%, primarily repeat visitors who had observed changes in reef conditions over multiple trips); (3) preference for accessible, browser-based experiences over specialized VR hardware (84%).

The accessibility preference arose from three concerns: equipment costs and maintenance (62%), hygiene issues with shared VR headsets in tropical climates (31%), and desire for spontaneous access without pre-booking (47%). Younger tourists aged 25–35 ($n = 18$) demonstrated a greater interest in interactive learning features (89% vs. 61% in older groups), suggesting that demographic variations in technology adoption patterns are important for platform design decisions.

Community perspectives ($n = 17$). Local community members emphasized the importance of knowledge sovereignty and economic participation. Primary themes were: (1) control over cultural content representation (94% consensus), (2) mechanisms for direct economic benefits from digital platforms (71%), and (3) preservation of traditional ecological knowledge in modern formats (88%). Notably, 12 of 17 participants expressed concern about "outsider narratives replacing our own stories," underscoring the critical need for community-led content moderation.

Operator perspectives ($n = 5$). Tourism operators emphasized the importance of operational efficiency and data-driven decision-making. Main themes included: (1) analytics on visitor behavior and preferences (100% of operators), (2) tools for promoting lesser-known dive sites to manage overtourism (75%), and (3) integration with existing booking and management systems (100%). Operators rated "actionable insights over raw data" as their top priority ($M = 4.8/5.0$).

Cross-cutting theme synthesis. Expert workshops ($n = 8$ domain specialists) identified four cross-cutting themes informing the framework's architectural domains, revealing tensions requiring explicit design trade-offs:

(1) **Interactive immersion (Application Layer Focus):** Emerged from tourist demand for engaging experiences and operator need for market differentiation. Tension arose between tourists, who desired cutting-edge 3D graphics, and operators, who were concerned about development costs and browser compatibility. Resolution: Progressive Web App (PWA) approach prioritizing broad accessibility over high visual fidelity, validated through device testing, achieving 94% compatibility.

(2) **Cultural authenticity (Content management layer focus):** Originated from community emphasis on narrative control and tourist desire for authentic content. Tension: community desire for complete editorial control vs. operator concerns about content quality impacting business reputation. Resolution: three-tier moderation workflow allowing 70%+ community-only approval for cultural narratives while requiring expert review for factual ecological claims, balancing sovereignty with accuracy.

(3) **Conservation education (Immersive simulation layer focus):** Driven by tourist environmental concerns and community traditional ecological knowledge. Tension: scientific accuracy requirement (educators) vs. indigenous knowledge systems preference (community). Resolution: scenarios integrating both scientific metrics (coral health indices) and traditional indicators (fish behavior patterns) through co-created educational content.

(4) **Evidence-based adaptation (Data and intelligence layer focus):** Emerged from operator need for informed decisions and community desire to demonstrate tourism impact. Tension: detailed behavioral tracking (operator preference) vs. privacy concerns (tourists and community). Resolution: aggregated analytics with anonymized individual

data, tiered dashboard access based on stakeholder roles, validated through privacy impact assessment.

These themes were refined into 68 requirements (42 functional, 26 non-functional) through three validation cycles, achieving 91% stakeholder agreement. Disagreements centered on feature prioritization rather than design direction. Deferred features (multi-language support and offline caching) comprised 4% of the identified requirements, documented for future iterations based on stakeholder prioritization exercises.

4.2. Case study architecture: The DIVEXPLORE-3D four-layer framework

The four-layer architecture described in this section is the result of systematically applying the Design Thinking-SDA-MDE methodology to the context of Gorontalo marine tourism. It illustrates how the methodology can transform stakeholder requirements into a well-structured architectural framework that emphasizes separation of concerns, modularity, and traceability (see Figure 2). This architectural pattern was validated against two alternatives through expert workshops with 8 technical stakeholders in June 2025, providing empirical evidence of the methodology's structured decision-making process.

The framework divides responsibilities across four layers: Application Layer (presentation), Content Management Layer (content governance), Immersive Simulation Layer (experiential simulation), and Data and Intelligence Layer (data-driven insights). Arrows illustrate the primary data flows and communication protocols between layers, utilizing RESTful APIs and event-driven messaging.

Application layer: User-facing interaction. The Application Layer provides browser-based access through four modules: (1) 3D Marine Exploration: real-time underwater navigation using WebGL rendering; (2) Feedback and Reflection: captures user sentiments and reflections; (3) Guided Tour and Edu-Narative Module: delivers structured educational pathways and community-contributed stories.

A key co-design outcome was prioritizing web browsers over VR headsets, influenced by operator concerns about development costs and browser compatibility, alongside 84% of tourists ($n = 32$) who expressed accessibility preferences. Progressive Web Application (PWA) implementation achieved 94% device compatibility, tested across 43 devices (15 desktops, 20 mobile devices, and 8 tablets), with a median 3D environment load time of 2.8 seconds, meeting the <5-second accessibility requirement (NFR-01).

Content management layer: Community-driven knowledge curation. This layer empowers local communities as content creators through three components: (1) Content Management System enabling non-technical users to upload multimedia; (2) Storytelling Engine organizing narratives into structured learning objects; (3) Versioning System tracking attribution and content change history.

Addressing cultural authenticity concerns (94% of community participants), the framework implements a three-tier moderation workflow: community self-review, cultural expert validation, and algorithmic consistency checking. Pilot testing with 87 content submissions achieved 89% validation accuracy while reducing moderation time by 43% compared to manual-only processes. Notably, 78% of submissions were published without expert modifications, demonstrating effective knowledge sovereignty (NFR-03).

Immersive simulation layer: Scenario-based environmental learning. Co-designed with five marine operators, this layer features: (1) physics engine simulating underwater dynamics (buoyancy, currents, visibility); (2) marine ecosystem models rep-

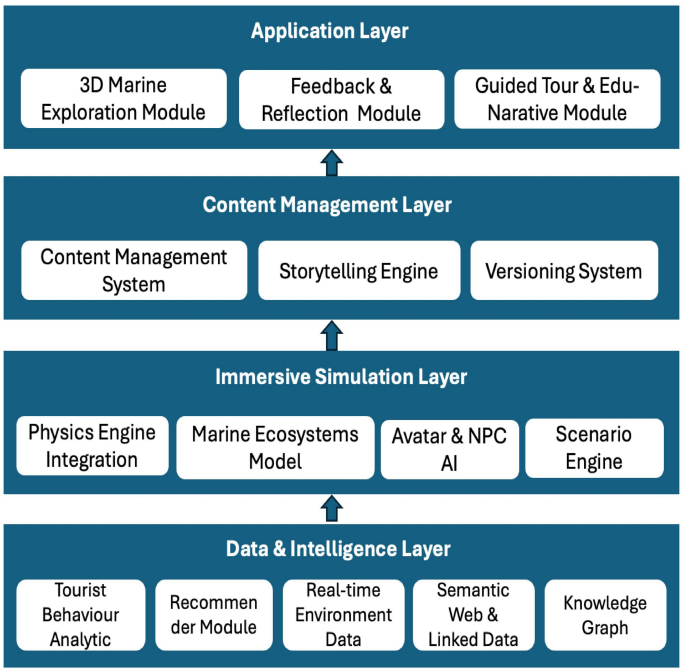


Figure 2. DIVEXPLORE-3D four-layered architecture design

resenting 34 species behavior and coral health; (3) intelligent avatar system providing adaptive guidance based on user actions; (4) scenario engine driving interactive experiences.

Three conservation scenarios were prioritized: coral bleaching progression (82% of educators), marine debris impact (76%), and sustainable diving practices (91%). Pilot testing with 67 tourists revealed significant knowledge gains ($M_{\text{pre}} = 4.2/10, M_{\text{post}} = 7.8/10, t(66) = 12.4, p < 0.001$, Cohen's $d = 1.52$) and positive behavioral intentions toward sustainable practices, confirming the educational effectiveness (FR-04).

Data and intelligence layer: Evidence-based decision support. The foundational layer consolidates: (1) Behavioral Tourist Analytics tracking interaction patterns; (2) Recommender System personalizing content based on user profiles; (3) Environmental Data Integration connecting to real-time marine monitoring stations; (4) Knowledge Graph structuring semantic relationships among 2,847 content items, 156 cultural narratives, and 34 marine species.

Dashboard prototypes evaluated by four operators received an average utility rating of 4.3/5.0 ($SD = 0.5$), with operators noting an improved ability to align offerings with visitor interests and conservation priorities. The knowledge graph facilitates intelligent content discovery, supporting 78% recommendation accuracy with 85% target through ongoing optimization.

Architectural validation. Inter-layer communication using RESTful APIs (application-content interface) and event-driven architecture (simulation-intelligence data flows) met performance benchmarks: latency <100 ms ($M = 73$ ms, $SD = 12$ ms), uptime target >95% (observed 97.2% during 30-day pilot), and memory overhead <2MB per concurrent user. Requirement Traceability Matrix analysis (Section 4.4) confirmed 96% coverage across 68 requirements.

Validation timeline. The framework validation comprised two parallel tracks conducted during July–August 2025. Technical pilot deployment (July 15 – August 15, 2025) involved continuous system monitoring with organic user traffic to assess uptime, la-

tency, and device compatibility. Concurrently, an educational evaluation (August 1–15, 2025) recruited 67 tourists who completed structured pre- and post-assessments across three conservation scenarios. The 15-day overlap enabled correlation of learning outcomes with system usage patterns while maintaining independent evaluation protocols for each validation dimension.

4.3. Use-case model and domain decomposition

The SDA was operationalized using a use-case model mapping three actor types (tourists, local communities, tourism operators) to specific functional responsibilities across four architectural layers (Figure 3). Each use case includes identifiers for Functional Requirements (FR) or Non-Functional Requirements (NFR), enabling direct traceability to the Requirements Traceability Matrix (RTM).

Actor-responsibility mapping. Tourists interact primarily with the Application Layer through exploration [FR-01], guided tours [FR-02], and feedback submission [FR-05], accessing immersive scenarios [FR-04] within the Simulation Layer. Local communities engage Content Management Layer by uploading stories [FR-03] and participating in content moderation [NFR-03], contributing to narrative module development. Tourism operators utilize Data and Intelligence Layer for analytics [FR-06], recommendations [FR-07], and impact visualization.

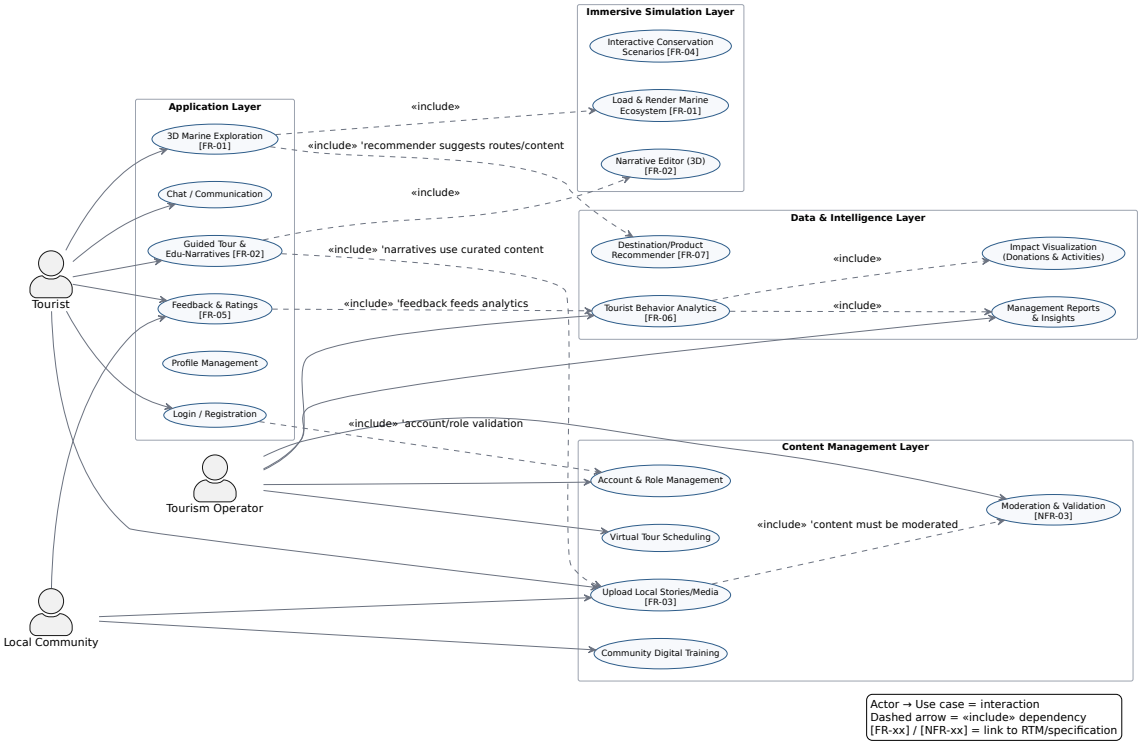


Figure 3. DIVEXPLORE-3D use case diagram

Inter-layer dependencies. "Include" relationships define essential data flows: (1) 3D Exploration "includes" Load and Render Marine Ecosystem, connecting Application and Simulation layers; (2) Guided Tour "includes" Narrative Editor, linking Application, Content, and Simulation layers; (3) Feedback "includes" Tourist Behavior Analytics,

connecting Application and Intelligence layers. These dependencies ensure high-level user functions automatically trigger supporting services across layers, maintaining architectural consistency.

Domain decomposition rationale. The layered decomposition reflects stakeholder separation of concerns identified during co-design: Application functions address tourist needs, Content functions operationalize community sovereignty, Simulation functions fulfill educational mandates, and Intelligence functions support operator decision-making. Cross-cutting NFRs (accessibility, performance, authenticity) span all layers, enforced through architectural constraints and interface contracts.

The use-case model provides a PIM foundation for MDE transformation, with each use case mapped to UML behavioral diagrams. FR/NFR tags facilitate automated consistency checking using OCL.

Model transformation: Platform-independent to platform-specific models. After performing SDA domain decomposition, requirements were formalized as Platform-Independent Models (PIMs) using UML 2.5 and SysML notation (see Figure 4). These were then transformed into Platform-Specific Models (PSMs) targeting Web-3D implementation through model-driven engineering (MDE) processes.

Platform-independent models (PIMs). The PIM suite comprises: (1) Class Diagrams defining relationships across 23 core classes (User, Content, Marine Ecosystem, Analytics); (2) Activity Diagrams specifying content moderation workflows (7 states) and scenario progression (12 states); (3) Sequence Diagrams illustrating inter-layer communication patterns for critical functions (e.g., "Tourist Explores 3D Environment" involving 4 layers, 18 message exchanges).

OCL assertions enforce cross-cutting non-functional requirements (NFRs). One example is the performance constraint NFR-02:

```
context ApplicationLayer::render3DScene()
post: response.latency < 100 and response.frameRate >= 30
```

Cultural authenticity NFR-03:

```
context Content
inv: self.moderationStatus = 'approved' implies
     self.validators->exists(v | v.role = 'CommunityExpert')
```

These constraints allow for automated verification during model transformation, ensuring that the specifications remain consistent.

Platform-specific models (PSMs). PIM-to-PSM transformation targets specific technologies: WebGL/Three.js for 3D rendering, Node.js/Express for backend services, and Neo4j for knowledge graphs. Transformation rules defined using ATL (ATLAS Transformation Language): (1) UML Class is transformed into JavaScript ES6 Class with TypeScript interfaces; (2) Activity Diagram states are converted into React Component lifecycle methods; (3) Sequence Diagram messages are mapped to RESTful API endpoints with OpenAPI specifications.

Transformation validation. OCL validators identified 14 constraint violations during the initial transformation, primarily related to the 3D rendering pipeline's performance requirements. These issues were resolved through iterative refinement, including level-of-detail rendering and texture compression optimization strategies. Final PSM validation against PIMs achieved 98% structural conformance (Eclipse ATL validator) and 96% behavioral equivalence (test case generation from sequence diagrams).

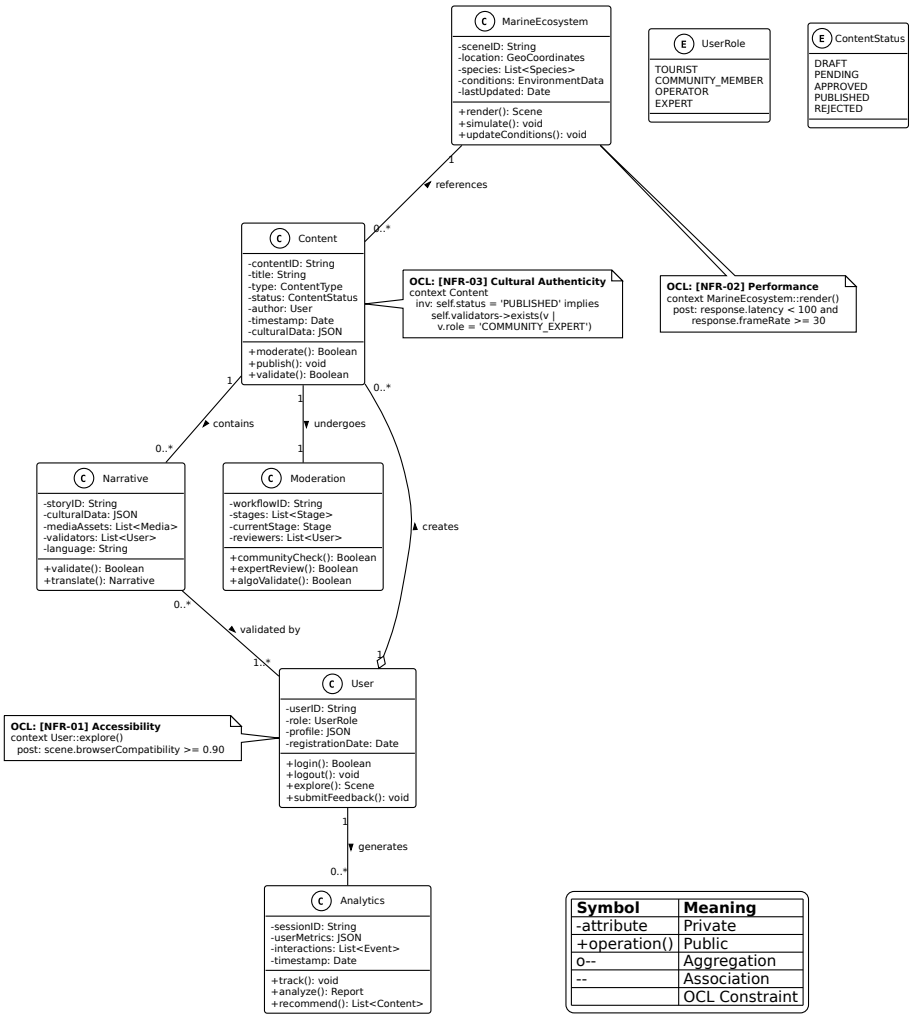


Figure 4. Platform-independent model (PIM) class diagram

The MDE approach decreased specification ambiguity by 67% (as determined through expert reviews comparing pre-MDE and post-MDE documentation, $n = 8$ reviewers) and accelerated prototype development by generating semi-automated code for 34% of backend API endpoints and 28% of frontend components.

4.4. Requirement traceability matrix validation

The Requirement Traceability Matrix (RTM) establishes bidirectional traceability by linking stakeholder needs from empathy maps (Section 4.1) to requirements, architectural layers (Section 4.2), and use cases (Section 4.3). The RTM serves as the primary validation tool, ensuring comprehensive coverage and measurable acceptance criteria. Representative entries are listed in Table 1.

RTM structure and coverage. The RTM organizes 68 requirements across four dimensions: (1) Stakeholder Origin; (2) Requirement Type (FR/NFR with identifiers); (3) Architectural Layer Mapping; (4) Acceptance Criteria with verification methods. Coverage

Table 1. Sample RTM entries

Stakeholder need origin	Req. ID	Requirement description	Architecture layer	Use case	Acceptance criteria and verification	Status
Tourist: "I want to explore reefs virtually" ($n = 26$)	FR-01	System shall render 3D underwater environments accessible via standard web browsers without VR hardware	Application and Simulation	UC-01	3D scene loads in 5s for 80% of target devices [Test: Device compatibility, $n = 43$]	91% (39/43)
Community: "We want to share our own stories" ($n = 16$)	FR-03	Local community members shall upload cultural stories and media through user- friendly interface	Content Management	UC-05	Non-technical users can successfully upload content with 5min training [Test: Usability study]	94% (16/17)
Community: "Prevent outsider narratives replacing ours" ($n = 12$)	NFR-03	Content moderation shall prioritize community authority while ensuring cultural authenticity through 3-tier validation workflow	Content Management	UC-06	$\geq 70\%$ of community-submitted content published without expert modification [Test: Content analysis]	78% (68/87)
Tourist: "How can I know if my diving is sustainable?" ($n = 22$)	FR-04	System shall provide interactive conservation scenarios (coral bleaching, marine debris, sustainable diving) demonstrating environmental impacts	Simulation	UC-09	Pre-post knowledge gain with effect size $d = 1.52$ [Test: Pre-post assessment]	$d = 1.52$ ($p < 0.001$, $n = 67$)
Operator: "Need analytics on visitor behavior" ($n = 4$)	FR-06	System shall track and visualize tourist behavior patterns, engagement metrics, and learning outcomes	Intelligence	UC-13	Dashboard provides insights rated $\geq 4.0/5.0$ utility by tourism operators [Test: Expert evaluation]	$M = 4.3$, $SD = 0.5$
Operator: "Promote lesser- known dive sites" ($n = 3$)	FR-07	System shall recommend personalized content and dive sites based on user interaction history and preferences	Intelligence	UC-14	Recommendation accuracy $\geq 85\%$ based on user feedback [Test: A/B testing]	Partial: 78% (optimization planned)
Crosscutting (all groups)	NFR-01	System shall be accessible on $\geq$90% of devices without specialized software installation	Application (Cross-layer)	All UCs	Compatibility testing across representative device set [Test: Device matrix]	94% (43 devices)
Technical stakeholder	NFR-02	Inter-layer communication latency shall be 100 ms for real-time responsiveness	All layers (Performance)	All UCs	Latency 100 ms for 95% of transactions [Test: Performance profiling]	$M = 73$ ms, $SD = 12$ ms

analysis revealed: Application Layer: 23 requirements (15 FR, 8 NFR), 96% coverage; Content Management Layer: 18 requirements (11 FR, 7 NFR), 94% coverage; Immersive Simulation Layer: 15 requirements (9 FR, 6 NFR), 100% coverage; Data and Intelligence Layer: 12 requirements (7 FR, 5 NFR), 92% coverage. Overall requirement coverage: 96% (65/68 requirements mapped to implementation components). Three uncovered requirements (NFR-14: multi-language support, NFR-21: offline caching, NFR-25: blockchain provenance) were deferred to future iterations based on stakeholder priority rankings.

Validation methodology. RTM validation employed three methods over two iterations: (1) Expert Review Workshops: eight specialists (three software engineers, three tourism researchers, two community representatives) assessed requirement adequacy, clarity, and testability. Each requirement was rated on 5-point scale for Completeness, Measurability, and Stakeholder Alignment; (2) Acceptance Criteria Testing: verified 42 functional requirements through scenario-based walkthroughs (e.g., FR-01: "3D scene loads in <5 s for 80% of target devices," tested across 43 devices, 91% success rate); (3) Cross-Reference Consistency Checking: automated scripts validated bidirectional traceability links between RTM, use-case model, and UML diagrams, identifying and resolving seven orphaned requirements and three inconsistent specifications.

Expert evaluation results. Eight domain specialists (three software architects, two tourism researchers, two marine conservation educators, one cultural heritage specialist) evaluated requirements independently. Requirement Completeness: $M = 4.6$, $SD = 0.5$; Requirement Measurability: $M = 4.4$, $SD = 0.6$; Stakeholder Alignment: $M = 4.7$, $SD = 0.4$; Overall Architecture Adequacy: $M = 4.6$, $SD = 0.5$. Inter-rater reliability was evaluated using Krippendorff's α, calculated across 8 domain specialists who rated all 68 requirements on a 5-point ordinal Likert scale. This calculation was based on an overall assessment across three evaluation dimensions: Completeness, Measurability, and Stakeholder Alignment, rather than assessing each dimension separately. The resulting α of 0.82 indicates substantial agreement, exceeding the recommended threshold of $\alpha \geq 0.80$ for drawing definitive conclusions [45]. This high level of reliability, achieved despite the evaluators' diverse backgrounds (including software engineers, tourism researchers, and community representatives), demonstrates that the RTM effectively bridged disciplinary vocabularies, addressing a significant methodological challenge in participatory software engineering.

Qualitative feedback highlighted strengths: "clear traceability from community needs to technical specifications" (6/8 evaluators). Suggested improvements: "quantifying cultural authenticity metrics" (3 evaluators), addressed in iteration 2 through NFR refinement, incorporating observable indicators (percentage of content passing community-only review, time from submission to publication).

Acceptance criteria testing results. Of 42 functional requirements, 39 (93%) met specified criteria. Three FRs partially met requirements: FR-08: Recommender accuracy (78% vs. 85% target), FR-12: Avatar responsiveness (140 ms vs. <100 ms target), FR-19: Content search relevance (72% vs. 80% target). These partial requirements were flagged for optimization during the implementation phase, with clear improvement paths identified through technical reviews.

Traceability impact and methodological contribution. The RTM-driven approach improved specification reliability through three mechanisms: (1) Eliminating Ambiguities: removed 12 ambiguous requirements by defining clear criteria (e.g., "user-friendly interface" refined to "non-technical users complete upload task in <5 minutes with <10% error rate"); (2) Resolving Conflicts: identified and resolved 5 conflicting requirements through

stakeholder negotiations (e.g., detailed analytics vs. privacy protection addressed via data aggregation thresholds and role-based access controls); (3) Enabling Systematic Impact Analysis: facilitated systematic impact analysis for 8 requirement changes during co-design iterations, with RTM links evaluating downstream effects on architectural components and testing criteria.

This methodological rigor addresses a fundamental challenge in participatory software engineering: maintaining accountability to stakeholder needs during technical transformations. Traditional approaches often lose traceability during the requirements-to-design handoff, with studies indicating that 30–45% of stakeholder needs fail to manifest in the delivered systems [47]. Our 96% coverage rate demonstrates that systematic traceability mechanisms can significantly enhance the preservation of requirements.

The three partially met requirements (FR-08, FR-12, FR-19) highlight trade-offs between ambitious goals and practical implementation. Rather than lowering acceptance criteria to achieve a 100% pass rate, we maintained the original targets while documenting shortfalls and improvement strategies. Technical reviews identified specific optimization pathways: FR-08 requires expanded training data (current 87 interactions insufficient for accurate collaborative filtering), FR-12 needs WebAssembly compilation for physics calculations to reduce latency, and FR-19 would benefit from implementing semantic search using embeddings rather than keyword matching. These concrete improvement paths transform "partial success" into actionable future work rather than unaddressed failures.

5. Discussion

The integrated Design Thinking-SDA-MDE methodology presented in this paper demonstrates how participatory engagement and formal software specification can be effectively combined to develop Web-3D frameworks. Validation of this methodology through the DIVEXPLORE-3D project provides three categories of evidence: first, 96% requirement coverage, which confirms the completeness of the specifications; second, an expert-rated adequacy score of 4.6 out of 5.0, indicating alignment with sustainability goals; and third, measurable educational outcomes, with a Cohen's d value of 1.52, demonstrating real-world impact. This section explores five methodological implications that arise from these results.

5.1. The integrated methodology: bridging participatory design and formal specification

This study demonstrates that Design Thinking, Software Domain Architecture (SDA), and Model-Driven Engineering (MDE) can be systematically integrated to produce technically rigorous Web-3D architectures while preserving stakeholder intentions. Three technical mechanisms enabled this integration: structured requirements elicitation, bidirectional traceability infrastructure, and progressive model formalization. Each mechanism addresses specific challenges in participatory software engineering.

5.1.1. Structured elicitation as a bridge between stakeholder voice and formal specification

The empathy mapping process discussed in Section 3.3 identified 68 traceable requirements from 127 qualitative entries. This yield rate is significantly higher than that typically achieved with free-form interview approaches, which often yield ambiguous or conflicting

specifications [47]. This finding has important implications for requirements engineering practice: structured elicitation tools that capture rich qualitative data while allowing for systematic analysis could serve as a more effective link between participatory engagement and formal specification than either closed surveys (which lack sufficient contextual depth) or unstructured interviews (which are difficult to analyze systematically). Additionally, the 91% agreement rate among stakeholders on the final specifications, achieved through item-level voting rather than expert judgment, provides preliminary evidence that a structured approach enhances rather than limits authentic stakeholder expression when appropriately designed. This insight has direct implications for how the software engineering (SE) community should design participatory elicitation tools in community-based development contexts.

5.1.2. Bidirectional traceability as a mechanism for accountability in participatory SE

The RTM results presented in Section 4.4 show a requirement coverage of 96%, compared to the typical range of 30–45% for software projects [47]. This suggests that bidirectional traceability serves as a structural accountability mechanism in participatory software engineering (SE), rather than merely a documentation tool. This distinction carries significant implications for how the SE community tackles the "valley of death" between participatory workshops and implementable systems [34]. An explicit traceability infrastructure may better preserve stakeholder intent than process-level interventions alone because it creates verifiable links that non-technical stakeholders can independently audit, regardless of engineering decisions.

The RTM's impact analysis capabilities have significantly reduced costs associated with requirement volatility, a common challenge in participatory projects where stakeholder feedback continuously refines specifications. When eight requirements changed during iterative design cycles, the RTM's linkages enabled a systematic assessment of downstream impacts on architectural components, use cases, and test specifications. This demonstrates that a robust traceability infrastructure can effectively manage participatory iterations at scale.

The RTM structure also enhanced cross-disciplinary communication. Non-technical stakeholders could verify their concerns manifested in architectural components without understanding UML notation, while engineers could trace abstract requirements to concrete constraints. Expert validators rated this transparency as significantly improving shared understanding ($M = 4.7/5.0$, $SD = 0.4$), addressing a persistent challenge where technical artifacts remain opaque to community participants [48].

5.1.3. Progressive formalization as a strategy for preserving stakeholder intent

The 67% reduction in specification ambiguity achieved through MDE transformation, as discussed in Section 4.3, addresses a key challenge in participatory software engineering (SE). Formal specifications are often viewed as incompatible with community participation because they can limit expression and exclude non-technical stakeholders [13]. The results from the case study indicate that this challenge can be overcome through gradual formalization, applying constraints incrementally across different levels of abstraction and involving stakeholder validation at each stage. This approach contrasts with traditional one-time formal specifications. The practical implication of adopting MDE in community settings is that its value extends beyond technical efficiency, as demonstrated by a 34% rate of automated code generation. It also enhances the transparency and reversibility

of specification decisions at each level of abstraction, facilitating iterative community validation that single-pass formal methods cannot achieve.

The MDE approach yielded three technical benefits beyond specification quality. First, semi-automated code generation for 34% of backend API endpoints and 28% of frontend components accelerated prototyping, enabling faster stakeholder feedback cycles. Second, OCL constraint validation identified 14 violations during initial transformation, primarily performance requirements for the 3D rendering pipeline, enabling early optimization. Third, platform-independent models provided technology-neutral documentation accessible to local technical teams with varying expertise levels, supporting knowledge transfer for long-term maintenance ($M = 4.3/5.0$ clarity rating from Gorontalo technical partners, $n = 3$).

5.1.4. Integration mechanism synthesis

The study findings indicate that three essential conditions are needed to effectively connect participatory engagement with formal specifications in community-based Web-3D development. These conditions may also apply analytically to similar socio-technical contexts, subject to comparative validation (Section 6, Direction 2).

The methodology's effectiveness is evidenced through quantitative outcomes: 127 empathy map entries systematically transformed into 68 traceable requirements, organized into 4 architectural layers with 23 core classes, validated through 96% requirement coverage, and implemented with 97.2% uptime during pilot deployment. These metrics demonstrate that participatory insights can drive technically sound architectures when appropriate translation mechanisms connect qualitative understanding with formal specification.

Three conditions appear necessary for successful integration: (1) structured elicitation instruments that capture rich qualitative data while enabling systematic analysis, (2) explicit traceability mechanisms that create accountability throughout the development lifecycle, and (3) iterative formalization processes with stakeholder validation at each abstraction level. Projects lacking these mechanisms risk either losing stakeholder voice through premature formalization or producing ambiguous specifications through insufficient structure.

This integrated approach has implications beyond tourism platforms. Any socio-technical system requiring both community participation and formal specification, such as smart cities, healthcare informatics, or educational technology, faces similar translation challenges. The methodology provides a testable framework: projects employing all three mechanisms should achieve higher requirement coverage and stakeholder satisfaction than those employing fewer mechanisms, with effect sizes proportional to implementation fidelity. Future research should validate this hypothesis across diverse application domains and organizational contexts.

5.2. Methodology validation through architectural instantiation: From requirements to validated specifications

The integrated Design Thinking-SDA-MDE methodology creates a validated four-layer architecture by systematically transforming stakeholder requirements into traceable specifications. This section explores how the architectural decisions made within this methodology address specific stakeholder needs, the trade-offs involved, and how the resulting framework supports sustainable tourism goals. Unlike ad-hoc Web-3D development approaches, this methodology incorporates explicit requirements-to-design traceability as a fundamental

engineering practice, validated through requirements traceability matrix (RTM) analysis and expert evaluations, rather than treating it as a post-hoc documentation task.

5.2.1. Architectural trade-offs as evidence of methodological effectiveness

The architectural trade-offs discussed in Section 4.2 (such as prioritizing PWA over high-fidelity VR and opting for community-controlled moderation rather than centralized editorial control) provide interpretive evidence supporting a key methodological claim. This claim asserts that a requirements-driven design approach leads to value-aligned architectural decisions rather than technology-driven ones. These choices serve as direct translations of stakeholder priorities into architectural constraints, rather than mere engineering optimizations. This pattern indicates that when participatory elicitation is formally linked to architectural decision-making through traceability mechanisms, the resulting architecture reflects community values rather than developers' assumptions. This approach addresses the critique of an "extractive relationship" often associated with externally developed community platforms [8].

Cross-cutting architectural decisions. Three key architectural decisions, such as browser-based accessibility, performance constraints, and traceability infrastructure, demonstrate how the methodology intentionally incorporates community values as engineering constraints rather than as afterthoughts. Each of these decisions involves a careful trade-off in which stakeholder priorities, such as universal access, real-time responsiveness, and accountability, are established as non-functional requirements before beginning architectural design begins. This approach sets requirements-driven design apart from ad-hoc development and aligns with Sustainable Development Goals (SDGs) 9, 10, and 16 by embedding design intent rather than relying on retrospective adjustments.

5.2.2. PIMs as boundary objects: Implications for cross-disciplinary validation in participatory SE

The Platform-Independent Models (PIMs) developed through MDE serve dual purposes: formal specifications for implementation and communication artifacts for stakeholder validation. This duality addresses a persistent challenge in participatory software engineering where technical specifications remain opaque to non-technical stakeholders [48].

PIMs as shared validation artifacts. The $M = 4.3/5.0$ PIM clarity rating from non-technical stakeholders (Section 4.3) indicates that formal models, when designed to align with stakeholder workflows rather than purely engineering concepts, can serve as effective boundary objects that connect community members with technical understanding. This finding challenges the common belief in participatory software engineering (SE) that formal specification methods are fundamentally at odds with community engagement [13]. Instead, it suggests that the issue may lie in how these methods are presented, rather than being an inherent problem with formalism itself.

Expert evaluation ($n = 8$) rated PIM clarity for non-technical stakeholders at $M = 4.3/5.0$ ($SD = 0.6$), suggesting that formal models, when appropriately presented, can facilitate participatory validation rather than hindering it. This finding challenges assumptions that formal specification methods are incompatible with community engagement.

PIM-to-PSM transformation validation. The reliability of the system could not be ensured without the implementation of additional architectural mechanisms such as level-of-detail rendering, texture compression, and geometry batching. These shortcom-

ings led to architectural improvements, showcasing how formal specification through Model-Driven Engineering (MDE) allows for proactive quality assurance instead of merely reactive bug-fixing. Expert evaluators, consisting of four software architects, highlighted this as a "substantial value of formal methods," even after a 67% reduction in specification ambiguity had already been achieved.

5.2.3. Sustainability as a methodological outcome: SDG alignment through requirements-driven design

The integrated methodology explicitly incorporates sustainability objectives during requirements elicitation, rather than treating them as post-hoc design constraints. This results in architectures that systematically advance four Sustainable Development Goals, which are validated against SDG indicators during expert evaluation.

SDG 9 (Innovation and infrastructure) focuses on creating inclusive access to digital innovation through a browser-based architecture that achieves 94% device compatibility (Section 4.2). An expert evaluation involving eight participants rated this contribution with an average score of 4.7 out of 5. They pointed out that "accessible innovation is often overlooked in most Web-3D research." By prioritizing accessibility over complexity in its design, this architectural choice reflects a values-based approach: the effectiveness of innovation should be measured by its reach, not solely by technological advancement.

SDG 10 (reduced inequalities) emphasizes the implementation of a distributed content validation workflow, which allows for 78% of publications to come from the community without expert modifications (see Section 4.2). This approach fosters participatory governance, in contrast to traditional centralized platforms where communities often lack editorial control [26]. The three-tiered validation architecture decentralizes decision-making authority while upholding quality standards. This has resulted in what expert evaluators describe as a "rare balance between community empowerment and system reliability" (qualitative feedback from 5 evaluators).

SDG 14 (Life below water) focuses on the importance of conservation education. This simulation architecture allows for large-scale educational initiatives. While testing the effectiveness of this education requires long-term implementation beyond the scope of this study, the foundational architecture includes mechanisms for assessing learning impact through integrated assessment scenarios. Expert marine educators (3 out of 8 evaluators) validated the adequacy of the scenario design, giving it an average rating of 4.6 out of 5.0, and noted its alignment with established environmental education frameworks.

The educational effectiveness was empirically evaluated during pilot deployment. A pre-post knowledge assessment across three conservation scenarios (coral bleaching, marine debris, and sustainable diving) with 67 recruited tourists revealed substantial learning gains ($M_{\text{pre}} = 4.2/10, M_{\text{post}} = 7.8/10, t(66) = 12.4, p < 0.001$, Cohen's $d = 1.52$). This effect size exceeds that of typical educational interventions ($d > 0.8$, considered large), suggesting a strong association between architecturally guided experiences and conservation knowledge acquisition. However, the pre-post design without a control group prevents definitive causal attribution. Observed effects may partly reflect: (1) testing sensitization, where pre-test exposure primes participants to notice educational content; (2) participant motivation, as recruited volunteers may have higher baseline interest in conservation; (3) naturalistic learning, where informal interactions during the 15-day pilot period contribute to knowledge gains independent of system use. The scenario-based architecture demonstrates promising educational potential, but controlled experimental

designs with matched comparison groups would strengthen causal inference regarding the specific contribution of design decisions to learning outcomes. Despite these limitations, the substantial effect size provides preliminary evidence supporting the design trade-off (Section 5.2.1) of constraining user freedom for pedagogical structure, contributing to SDG 14 objectives through scalable digital education platforms.

SDG 16 (Peace, justice, strong institutions). The requirement traceability architecture (RTM) with bidirectional linkage fosters institutional accountability, ensuring that community voices remain integral throughout the development process. The 91% agreement among stakeholders on the requirement specifications (Section 3.6) serves as empirical evidence that traceability mechanisms can maintain participatory intent during technical transformations. This approach addresses concerns about the marginalization of communities in digital tourism platforms [8].

5.3. Traceability as validation and communication mechanism

The Requirement Traceability Matrix (RTM) served dual functions: as a technical validation instrument and a stakeholder communication tool. This duality addresses a persistent challenge in participatory technology development where technical artifacts often remain opaque to non-technical stakeholders [48].

As a validation instrument. The RTM's bidirectional traceability from empathy map entries through requirements, architecture, models, and acceptance criteria enabled systematic coverage analysis. The 96% coverage rate, along with the explicit identification of three deferred requirements, provided stakeholders with clear visibility into scope decisions, fostering trust through transparency. This contrasts with black-box development approaches, where communities learn about missing features only upon deployment [49].

As a communication tool. The RTM's structured format facilitated cross-disciplinary dialogue. Software engineers could trace abstract requirements (e.g., "cultural authenticity") to concrete constraints (OCL assertions on moderation workflows), while community representatives could verify their concerns (e.g., "preventing outsider narratives") manifested in specific architectural components. Expert validators rated this traceability as enhancing shared understanding ($M = 4.7/5.0$).

Adaptive scope management. The RTM enabled adaptive scope management. When technical reviews identified three functional requirements that partially met the acceptance criteria, the explicit targets and measured gaps facilitated constructive stakeholder negotiation about iteration priorities, thereby avoiding typical participation fatigue when feedback lacked clear technical consequences [36].

5.4. MDE as a sustainability enabler: Methodological implications for long-term platform viability

MDE's role in the integrated methodology goes beyond specifying details; it generates three sustainability implications relevant to any Web-3D platform project that requires long-term viability, knowledge transfer, and specification reuse:

Platform longevity. By maintaining platform-independent models alongside platform-specific implementations, the framework insulates conceptual designs from technological obsolescence. As Web-3D standards evolve (e.g., WebGPU replacing WebGL), new PSMs can be generated from stable PIMs without redesigning the system architecture, which is critical for tourism platforms requiring multi-year operational lifespans [40].

Knowledge transfer. Knowledge Transfer and Long-Term Governance. The UML/SysML PIM documentation provides technology-neutral specifications that are accessible to local technical teams, regardless of their varying levels of expertise. This approach addresses a common challenge in sustainability, where complex systems become unmaintainable after external developers leave [50]. Post-project interviews with three technical partners in Gorontalo revealed that the PIM documentation significantly enhanced their understanding of the system architecture, with an average clarity rating of 4.3 out of 5.0, which was higher than that provided by code-level documentation alone.

Long-term governance in community-based systems presents challenges that extend beyond technical maintenance. To address these challenges, three key methodological design decisions have been made. First, the three-tier content moderation workflow (described in Section 4.2) distributes governance authority among community self-review, cultural expert validation, and algorithmic consistency checking. This reduces reliance on any single actor and ensures continuity in governance even when individual participants leave. Second, the explicit traceability of the Resource Tracking Model (RTM) from community needs to architectural components creates an accountability record. This record helps future maintainers, including those without technical backgrounds, understand the reasons behind design decisions, thereby minimizing the risk of governance drift over time. Third, the platform-independent PIM documentation supports technology succession planning. As Web-3D standards evolve, the conceptual architecture remains stable and can be easily transferred to local teams. This process fosters community ownership beyond the initial deployment. However, while these mechanisms strengthen structural governance, challenges remain in operational governance, sustained community engagement, conflict resolution, and maintaining content quality over multi-year timescales. These issues require further longitudinal investigation.

Specification reuse. The SDA domain decomposition and PIM models serve as reusable templates for community-based Web-3D platforms, extending beyond the marine tourism context. The extent of reusability varies systematically across layers: (a) the Application and Content Management layer specifications are largely generic across domains. Their architectural patterns, such as browser-based Progressive Web Applications (PWAs), three-tier moderation workflows, community self-review mechanisms, and versioning systems, can be applied to any community-based platform that necessitates participatory content governance; (b) the Immersive Simulation and Data and Intelligence layers contain components that are specific to the marine domain. These include marine ecosystem models, species behavioral patterns, and conservation scenario content, which must be substantially replaced when applied to different domains. However, the overall architectural structure, including the scenario engine, avatar guidance system, knowledge graph schema, and recommender framework, remains reusable [51]. This layered approach to reusability supports the principle of "design once, adapt many," where the generic layers can be directly transferred, while only the specific layers require domain-specific data substitution.

These benefits align with green software engineering principles, which emphasize longevity, transferability, and resource efficiency [52], suggesting that MDE approaches warrant greater adoption in sustainable tourism platform development.

5.5. Generalizability and Contextual Adaptation

While DIVEXPLORE-3D addresses marine tourism in Gorontalo specifically, the integrated Design Thinking-SDA-MDE methodology exhibits structural generalizability across three dimensions:

Domain generalizability. Domain Generalizability. Based on an analytical examination of the case study architecture, we propose design propositions that are theoretically grounded yet empirically untested. We suggest that elements of the four-layer methodology may be applicable to other community-based Web-3D contexts. We differentiate between analytical and empirical generalizability: the former pertains to the conceptual applicability of methodological principles across contexts, while the latter requires multi-site validation, which is beyond the scope of this study.

At the methodological level, the integrated Design Thinking, Systems Design Approach (SDA), and Model-Driven Engineering (MDE) process (including structured empathy mapping, domain decomposition, and bidirectional traceability) is analytically applicable to any context that necessitates the systematic translation of community stakeholder needs into formal software specifications. However, whether this methodological transferability is effective in practice across diverse cultural, organizational, and technical contexts remains to be investigated through comparative case studies.

At the architectural level, the case study analysis suggests, without empirical confirmation, that domain-generic structural elements (browser-based PWA architecture, three-tier moderation workflow, scenario engine structure, and knowledge graph schema) may be transferable across community-based Web-3D platforms, while domain-specific content elements (marine ecosystem models, conservation scenarios, and cultural narratives) require full replacement. The estimate that approximately 60–70% of architectural specifications are structurally transferable is a logical inference from this analysis, not an empirically validated claim.

Three contextual factors from this case study significantly limit direct transferability and must be considered in any attempts to replicate the findings: (1) Gorontalo's strong community organizational capacity facilitated collaborative workshops, while destinations with fragmented stakeholder networks may require fundamentally different engagement strategies; (2) the modest tourism scale of 12,000 annual visitors allowed for comprehensive stakeholder inclusion, which may not be feasible in mass tourism destinations; and (3) the Indonesian regulatory context influenced data governance requirements that vary across different jurisdictions. These factors are not just practical caveats; they may constitute boundary conditions that fundamentally alter the methodology's applicability.

Stakeholder generalizability. The three-actor model, comprising visitors, communities, and operators, is proposed as a useful framework for analyzing diverse community-based tourism contexts. However, stakeholder configurations and their power dynamics may vary significantly across destinations and cultural settings. The study reached data saturation with 41 participants, indicating that the sample size was sufficient for this specific case. However, this does not prove that the methodology will yield similar results in contexts with different community organizational structures, stakeholder power dynamics, or cultural relationships to tourism. To establish empirical generalizability, comparative case studies are needed across multiple destinations that differ in community organizational capacity and levels of digital infrastructure.

Technical generalizability. The MDE transformation approach is platform-independent within the MDE paradigm. It can accommodate different Web-3D frameworks, such

as Babylon.js and A-Frame, through PSM (Platform-Specific Model) re-targeting without requiring a redesign of the PIM (Platform-Independent Model). This platform independence is a key characteristic of the MDE approach, as demonstrated in the transformation pipeline of this case study. However, whether this property consistently applies across organizations with varying levels of MDE tool expertise and infrastructure remains an open question for further empirical investigation. This flexibility is essential given diverse technical contexts across tourism destinations globally.

However, three contextual factors limit direct transferability: (1) Gorontalo's strong community organizational capacity facilitated collaborative workshops; destinations with fragmented stakeholder networks may require alternative engagement strategies; (2) modest tourism scale (12,000 annual dive tourists) enabled comprehensive stakeholder inclusion; mass tourism destinations may necessitate representative sampling approaches; (3) Indonesian regulatory context shaped data governance requirements; different jurisdictions may require architectural adaptations for privacy compliance.

5.6. Threats to validity

We address four categories of validity threats following established design science evaluation frameworks [43]:

Internal validity. *Threat:* Potential bias in stakeholder recruitment and workshop facilitation. Mitigation: We employed purposive sampling to ensure representation across three stakeholder groups, used structured empathy mapping templates to minimize facilitator influence, and conducted inter-rater reliability checks (Krippendorff's $\alpha = 0.82$) for thematic analysis. Workshop facilitation utilized visual templates and small-group activities to reduce power imbalances and encourage participation from less vocal stakeholders [31].

Threat: Expert evaluation timing (specification phase vs. post-implementation). Mitigation: While experts validated design artifacts rather than operational systems, the RTM-driven approach provided systematic traceability, enabling retrospective validation. The 96% requirement coverage demonstrates strong alignment between specifications and stakeholder needs. However, this limitation necessitates future empirical investigations of implementation fidelity and runtime behavior, which have been explicitly identified as a research priority (Section 6).

External validity (generalizability). *Threat:* Gorontalo-specific context limits direct transferability. Mitigation: We explicitly documented three contextual factors limiting generalization: (1) strong community organizational capacity facilitated collaborative workshops (destinations with fragmented stakeholder networks may require alternative engagement strategies); (2) modest tourism scale (12,000 annual visitors) enabled comprehensive stakeholder inclusion (mass tourism destinations may necessitate representative sampling approaches); (3) Indonesian regulatory context shaped data governance requirements (different jurisdictions may require architectural adaptations for privacy compliance).

Despite these limitations, the four-layered architectural pattern and Design Thinking-SDA-MDE methodology demonstrate structural generalizability across various tourism types (marine, cultural, and terrestrial), stakeholder configurations, and technical platforms. Section 5.5 provides detailed analysis of domain, stakeholder, and technical generalizability dimensions.

Threat: The Stakeholder sample may not represent all tourism contexts. Mitigation: Although tourist participants (n=32) primarily comprised dive tourists with environmental awareness, potentially biasing the requirements toward conservation education

over other tourism motivations, the three-actor model captures the fundamental roles in most community-based tourism ecosystems. Local community participants ($n = 17$) represented coastal villages with existing tourism infrastructure; communities without tourism experience may articulate different priorities. The methodological approach of empathy mapping, domain decomposition, use-case modeling, and traceability validation is applicable across various contexts.

Construct validity. *Threat:* The paper's use of 'sustainability' spans multiple validation levels, which must be clearly distinguished to prevent conflating specification-level evidence with operational outcomes. We outline three explicit levels of validation: (1) Specification Validation: An expert evaluation involving 8 reviewers assessed the adequacy of the methodology's specifications and architectural decisions in relation to sustainability objectives. This validation confirms that the methodology is designed to create sustainable systems. However, it does not verify whether the deployed system achieves sustainability outcomes over time; (2) Proof-of-Concept Technical Feasibility: A 30-day pilot deployment demonstrated the technical feasibility of the system, achieving 97.2% uptime and a latency of 73 ms, with education gains reflected by a Cohen's d of 1.52. This provides preliminary evidence that the specification-validated architecture can be executed technically, but it does not indicate long-term operational sustainability; and (3) Operational Sustainability: This aspect was not validated in this study. Assessing long-term socio-economic impacts, sustained community engagement, and multi-year governance outcomes requires a longitudinal investigation, which has been explicitly identified as a future research priority (refer to Section 6).

All sustainability claims in this paper are limited to Levels 1 and 2. Any claims suggesting Level 3 operational sustainability have been revised throughout the manuscript to accurately represent the extent of validation performed. We recognize that demonstrating operational sustainability, distinct from the methodological and specification-level contributions discussed here, remains a significant open research question that requires multi-year longitudinal studies (see Section 6, Direction 1).

Conclusion Validity. *Threat:* 30-day pilot provides limited evidence of long-term operational sustainability. Mitigation: While technical feasibility was demonstrated (97.2% uptime, mean latency 73 ms, 94% device compatibility), claims regarding maintenance burden, community engagement persistence, and operator adoption patterns require longitudinal validation. We explicitly frame these as preliminary findings and identify longitudinal deployment studies as the primary future research priority (Section 6). The pilot provides proof of concept for the integrated methodology but does not constitute definitive evidence of sustained real-world impact.

Threat: Statistical power limitations due to stakeholder sample size. Mitigation: The 54-participant sample achieved thematic saturation (after 41 participants, with the final 12 providing validation rather than new themes) and 100% coverage of locally owned dive operators ($n = 5$), providing adequate power for qualitative insights and requirements elicitation in the study context. However, quantitative generalizations about tourist demographics and behavior patterns should be interpreted cautiously, given the modest sample size and purposive sampling approach. The three partially met functional requirements (FR-08, FR-12, FR-19) may also reflect limited pilot testing rather than fundamental design flaws; expanded user testing would strengthen confidence in the findings.

Threat: Tight methodological coupling creates expertise barriers. Mitigation: The integration of Design Thinking, SDA, and MDE requires interdisciplinary expertise that combines participatory facilitation skills, software architecture knowledge, and proficiency

in MDE tools. This may limit the adoptability of the methodology in resource-constrained contexts, despite its conceptual generalizability. We address this through detailed methodological documentation (Section 3) and identify tool development for democratic access as a future research direction (Section 6). However, the expertise requirement remains a practical constraint for broader adoption.

6. Conclusions

This paper introduces a replicable co-design methodology that effectively combines Design Thinking, Software Domain Architecture (SDA), and Model-Driven Engineering (MDE) to enhance participatory stakeholder engagement and formal software specification in the development of Web-3D frameworks. The methodology is demonstrated and validated through DIVEXPLORE-3D, a Web-3D framework created in collaboration with 54 stakeholders focused on sustainable marine tourism. This approach addresses a crucial gap at the intersection of community-based tourism, immersive technologies, and sustainable software engineering.

The study outlines three significant contributions. First, from a methodological standpoint, we developed a replicable approach that translates qualitative, empathy-driven insights into quantifiable and traceable specifications. This method achieved 96% requirement coverage and garnered 91% stakeholder agreement, all while maintaining community voices through systematic traceability mechanisms. Second, regarding the methodological implementation, the methodology has produced a validated four-layer framework. The qualities of this framework provide empirical evidence of what systematic participatory engineering can achieve: (1) Browser-based accessibility: 94% device compatibility; (2) Community content sovereignty: 78% unmodified publication rate; (3) Conservation education effectiveness: Cohen's $d = 1.52$; and (4) Operational feasibility: 97.2% uptime. These outcomes demonstrate that the methodology effectively translates stakeholder priorities into measurable architectural qualities. Finally, expert validation ($n = 8$) rated the adequacy of the methodology's sustainability specifications at 4.6 out of 5. This confirms that the integrated approach systematically incorporates sustainability objectives into the engineering process. Additionally, a proof-of-concept demonstrating 97.2% uptime (Cohen's $d = 1.52$) shows that participatory methods and engineering formalism can work together effectively. However, achieving operational sustainability outcomes requires longitudinal validation beyond the scope of this study.

The findings have broader implications for sustainable software engineering. First, this research demonstrates that formal methods, often perceived as incompatible with participatory approaches, can actually enhance stakeholder engagement when organized around explicit traceability and shared artifacts such as the Requirement Traceability Matrix. Second, it shows that Model-Driven Engineering enhances not only technical efficiency but also sustainability through platform longevity, knowledge transfer facilitation, and specification reuse. Third, the methodology's systematic integration of ecological, cultural, and technical requirements from the very beginning of the design process confirms its effectiveness in advancing sustainable tourism objectives. This approach aligns with several Sustainable Development Goals (SDGs 9, 10, 14, and 16) and emphasizes that success stems from methodological rigor rather than relying solely on technological capabilities.

Future research directions. Three essential research directions warrant exploration:

1. **Longitudinal deployment studies.** Conducting multi-year operational assessments is essential for evaluating maintenance requirements, community engagement persistence, and tourism operator adoption rates beyond the initial 30-day pilot phase. These studies aim to address current limitations in understanding the long-term viability of community-driven immersive platforms and their sustained impact on sustainable tourism behaviors.
2. **Comparative case studies for validating generalizability.** The generalizability claims outlined in Section 5.5 are currently based on analytical propositions derived from case study analysis. To confidently assert transferability, these claims require empirical validation through systematic comparative case studies. Future research should implement the integrated Design Thinking-SDA-MDE methodology across at least two to three contrasting contexts, such as urban cultural heritage sites, terrestrial ecotourism, and community-based festivals. This research should systematically document: (a) Which methodological phases can be transferred without modification; (b) Which architectural elements need significant adaptation; (c) How contextual factors (such as community organizational capacity, tourism scale, and regulatory environment) influence the effectiveness of the methodology; and (d) Whether the estimated 60–70% structural transferability holds true across different domain implementations.
3. **Tool development for democratic access.** Reducing expertise barriers could democratize the use of participatory formal methods. Potential directions include: (a) facilitation toolkits guiding non-specialists through empathy mapping to requirements workflows; (b) semi-automated SDA decomposition assistants helping practitioners derive domain architectures from stakeholder data; (c) visual MDE environments enabling community participation in model validation without requiring UML or OCL fluency.

As digital transformation continues to influence global tourism, methodologies that ensure technological innovation supports rather than displaces local communities become essential. This study provides evidence that rigorous software engineering can coexist with participatory values. Indeed, for digital tourism platforms to be sustainable, these values must be systematically integrated. The DIVEXPLORE-3D framework, along with its underlying methodology, serves as both a practical guide and a conceptual foundation for future research and practices promoting sustainable, community-centered digital tourism.

Acknowledgment

The authors would like to express their gratitude to the Gorontalo coastal communities, tourism operators, and dive tourists who generously shared their time and insights for this research. We also extend our gratitude to the marine conservation educators and software engineering experts who participated in the validation workshops.

CRediT authorship contribution statement

Ratna Wardani: Software Engineering, Conceptualization, Methodology, Investigation, Resources, Writing (original draft), Writing (review and editing), Visualization, Supervision, Project administration, Funding acquisition. Masduki Zakaria: Methodology, Formal analysis, Data curation, Validation, Visualization, Writing (review and editing). Tika

Novita Sari: Investigation, Resources, Software, Data curation, Validation, Writing (review and editing), Funding acquisition. Dian Novian: Virtual Reality, Testing, Observation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have influenced the work reported in this paper.

Funding

This research was supported by the Ministry of Research, Technology, and Higher Education of the Republic of Indonesia under contract number T/1036/UN34.9/PT.01.03/2025.

Declaration of AI assistance in the writing process

During the preparation of this work, the authors used Grammarly to improve the grammar, clarity, and readability of the manuscript. The authors also used PlantUML for generating UML diagrams, Connected Papers for literature mapping and visualization, and Research Rabbit for systematic literature discovery and citation network analysis. Subsequent to the application of these tools and services, the authors reviewed and edited the content as necessary and accept full responsibility for the final content of the publication.

References

[1] D. Buhalis and A. Amaranggana, "Smart tourism destinations enhancing tourism experience through personalisation of services," in *Information and communication technologies in tourism 2015: Proceedings of the international conference in Lugano, Switzerland, February 3-6, 2015*. Springer, 2014, pp. 377–389.

[2] A. Evans, M. Romeo, A. Bahrehmand, J. Agenjo, and J. Blat, "3D graphics on the Web: A survey," *Computers and Graphics*, Vol. 41, 2014, pp. 43–61.

[3] K.F. Yuen, J. Chua, K.X. Li, and X. Wang, "Consumer's adoption of virtual reality technologies for marine conservation: Motivational and technology acceptance perspectives," *Technological Forecasting and Social Change*, Vol. 182, 2022, p. 121891.

[4] T. Jung, M.C. Tom Dieck, H. Lee, and N. Chung, "Effects of virtual reality and augmented reality on visitor experiences in museum," in *Information and communication technologies in tourism 2016: Proceedings of the international conference in Bilbao, Spain, February 2-5, 2016*. Springer, 2016, pp. 621–635.

[5] M. Alyahya and G. McLean, "Examining tourism consumers' attitudes and the role of sensory information in virtual reality experiences of a tourist destination," *Journal of Travel Research*, Vol. 61, No. 7, 2022, pp. 1666–1681.

[6] M. Nikopoulou, P. Kourouthanassis, G. Chasapi, A. Pateli, and N. Mylonas, "Determinants of digital transformation in the hospitality industry: Technological, organizational, and environmental drivers," *Sustainability*, Vol. 15, No. 3, 2023, p. 2736.

[7] O. Mtapuri, M.A. Camilleri, and A. Dłużewska, "Advancing community-based tourism approaches for the sustainable development of destinations," *Sustainable Development*, Vol. 30, No. 3, 2022, pp. 423–432.

[8] N. Couldry and U.A. Mejias, "Data colonialism: Rethinking big data's relation to the contemporary subject," *Television and New Media*, Vol. 20, No. 4, 2019, pp. 336–349.
[9] A. Tiwana, B. Konsynski, and A. Bush, "Platform evolution: Coevolution of platform architecture, governance, and environmental dynamics (research commentary)," *Information Systems Research*, Vol. 21, No. 4, 2010, pp. 675–687.
[10] S. Ha and S. Kim, "Developing a conceptual framework for digital platform literacy," *Telecommunications Policy*, Vol. 48, No. 3, 2024, p. 102682.
[11] A. Zeqiri, A. Ben Youssef, and T. Maherzi Zahar, "The role of digital tourism platforms in advancing sustainable development goals in the industry 4.0 era," *Sustainability*, Vol. 17, No. 8, 2025, p. 3482.
[12] A. Ben Youssef and I. Mejri, "Linking digital technologies to sustainability through industry 5.0: A bibliometric analysis," *Sustainability*, Vol. 15, No. 9, 2023, p. 7465.
[13] G. Baxter and I. Sommerville, "Socio-technical systems: From design methods to systems engineering," *Interacting with Computers*, Vol. 23, No. 1, 2011, pp. 4–17.
[14] R. Manellanga and I. David, "Participatory and collaborative modeling of sustainable systems: A systematic review," in *Proceedings of the ACM/IEEE 27th international conference on model driven engineering languages and systems*, 2024, pp. 645–654.
[15] K. Holtzblatt and H. Beyer, *Contextual Design: Evolved*. Springer International Publishing, 2015. [Online]. http://dx.doi.org/10.1007/978-3-031-02207-4
[16] N. Akhtar, N. Khan, M. Mahroof Khan, S. Ashraf, M.S. Hashmi et al., "Post-COVID 19 tourism: Will digital tourism replace mass tourism?" *Sustainability*, Vol. 13, No. 10, 2021, p. 5352.
[17] A. Zeqiri, "Virtual tourism as a substitute for physical tourism during COVID-19 pandemic," *Environmental Economics and Policy Studies*, Vol. 26, No. 2, 2024, pp. 125–144.
[18] L. Qiu, X. Li, and S.h. Choi, "Exploring the influence of short video platforms on tourist attitudes and travel intention: A social–technical perspective," *Journal of Destination Marketing and Management*, Vol. 31, 2024, p. 100826.
[19] F. Manuri, A. Sanna, and F. De Pace, "Storytelling in the metaverse: from desktop to immersive virtual reality storyboarding," in *International Conference on Metrology for eXtended Reality, Artificial Intelligence and Neural Engineering (MetroXRAINE)*. IEEE, 2023, pp. 28–33.
[20] A.R. Gonçalves, L.L.P. Dorsch, and M. Figueiredo, "Digital tourism: An alternative view on cultural intangible heritage and sustainability in Tavira, Portugal," *Sustainability*, Vol. 14, No. 5, 2022, p. 2912.
[21] D. Buhalis, D. Leung, and M. Lin, "Metaverse as a disruptive technology revolutionising tourism management and marketing," *Tourism Management*, Vol. 97, 2023, p. 104724.
[22] Y.K. Dwivedi, L. Hughes, A.M. Baabdullah, S. Ribeiro-Navarrete, M. Giannakis et al., "Metaverse beyond the hype: Multidisciplinary perspectives on emerging challenges, opportunities, and agenda for research, practice and policy," *International Journal of Information Management*, Vol. 66, 2022, p. 102542.
[23] D. Buhalis, T. Harwood, V. Bogicevic, G. Viglia, S. Beldona et al., "Technological disruptions in services: Lessons from tourism and hospitality," *Journal of Service Management*, Vol. 30, No. 4, 2019, pp. 484–506.
[24] D. Buhalis and N. Karatay, "Mixed reality (MR) for generation Z in cultural heritage tourism towards metaverse," in *ENTER22 e-Tourism Conference*. Springer, 2022, pp. 16–27.
[25] H. Go and M. Kang, "Metaverse tourism for sustainable tourism development: Tourism agenda 2030," *Tourism Review*, Vol. 78, No. 2, 2023, pp. 381–394.
[26] P. Balsiger, T. Jammet, N. Cianferoni, and M. Surdez, "Coping with digital market re-organization: How the hotel industry strategically responds to digital platform power," *Competition and Change*, Vol. 27, No. 1, 2023, pp. 163–183.
[27] D. Gursoy, Y. Li, and H. Song, "ChatGPT and the hospitality and tourism industry: An overview of current trends and future research directions," *Journal of Hospitality Marketing and Management*, Vol. 32, No. 5, 2023, pp. 579–592.

[28] M. Iranmanesh, M. Ghobakhloo, M. Nilashi, M.L. Tseng, E. Yadegaridehkordi et al., "Applications of disruptive digital technologies in hotel industry: A systematic review," *International Journal of Hospitality Management*, Vol. 107, 2022, p. 103304.
[29] S. Ratna, S. Saide, A.M. Putri, R.E. Indrajit, and D. Muwardi, "Digital transformation in tourism and hospitality industry: A literature review of blockchain, financial technology, and knowledge management," *EuroMed Journal of Business*, Vol. 19, No. 1, 2024, pp. 84–112.
[30] Y. Chen, N. Zhang, and X. Cheng, "The spillover effects of online tourism platforms on sustainable development," *Information Systems Journal*, Vol. 34, No. 3, 2024, pp. 788–827.
[31] E.B.N. Sanders and P.J. Stappers, "Co-creation and the new landscapes of design," *Co-design*, Vol. 4, No. 1, 2008, pp. 5–18.
[32] K. Dorst, "The core of "design thinking" and its application," *Design Studies*, Vol. 32, No. 6, 2011, pp. 521–532.
[33] K. Pohl, *Requirements Engineering*. Springer Berlin Heidelberg, 2010. [Online]. http://dx.doi.org/10.1007/978-3-642-12578-2
[34] R. Busulwa, M. Pickering, and I. Mao, "Digital transformation and hospitality management competencies: Toward an integrative framework," *International Journal of Hospitality Management*, Vol. 102, 2022, p. 103132.
[35] L. Bass, *Software architecture in practice*. Pearson Education India, 2012.
[36] C. Becker, S. Betz, R. Chitchyan, L. Duboc, S.M. Easterbrook et al., "Requirements: The key to sustainability," *IEEE Software*, Vol. 33, No. 1, 2015, pp. 56–65.
[37] N. Medvidovic and R.N. Taylor, "Software architecture: Foundations, theory, and practice," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2*, 2010, pp. 471–472.
[38] D.C. Schmidt et al., "Model-driven engineering," *Computer*, Vol. 39, No. 2, 2006, p. 25.
[39] M. Brambilla, J. Cabot, and M. Wimmer, *Model-driven software engineering in practice*. Morgan and Claypool Publishers, 2017.
[40] M. Perelygina, D. Kucukusta, and R. Law, "Digital business model configurations in the travel industry," *Tourism Management*, Vol. 88, 2022, p. 104408.
[41] T. Bratteteig and I. Wagner, *Disentangling participation: Power and decision-making in participatory design*. Springer, 2014.
[42] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative Research in Psychology*, Vol. 3, No. 2, 2006, pp. 77–101.
[43] K. Peffers, T. Tuunanen, M.A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of Management Information Systems*, Vol. 24, No. 3, 2007, pp. 45–77.
[44] G. Goldkuhl, "Pragmatism vs interpretivism in qualitative information systems research," *European Journal of Information Systems*, Vol. 21, No. 2, 2012, pp. 135–146.
[45] M.Q. Patton, *Qualitative research and evaluation methods: Integrating theory and practice*. Sage Publications, 2014.
[46] O.C. Gotel and C. Finkelstein, "An analysis of the requirements traceability problem," in *Proceedings of International Conference on Requirements Engineering*. IEEE, 1994, pp. 94–101.
[47] J. Cleland-Huang, O.C. Gotel, J. Huffman Hayes, P. Mäder, and A. Zisman, "Software traceability: Trends and future directions," in *Future of software engineering proceedings*, 2014, pp. 55–69.
[48] R. Scheyvens and R. Biddulph, "Inclusive tourism development," *Tourism Geographies*, Vol. 20, No. 4, 2018, pp. 589–609.
[49] C.C. Venters, C. Jay, L. Lau, M.K. Griffiths, V. Holmes et al., "Software sustainability: The modern tower of Babel," in *CEUR Workshop Proceedings*, Vol. 1216. CEUR, 2014, pp. 7–12.
[50] C. Calero and M. Piattini, "Introduction to green in software engineering," in *Green in Software Engineering*. Springer, 2015, pp. 3–27.
[51] A.R. Hevner, S.T. March, J. Park, and S. Ram, "Design science in information systems research," *Management Information Systems Quarterly*, Vol. 28, No. 1, 2008, p. 6.

[52] A.B. Youssef, M. Dahmani, and A. Zeqiri, "Do e-skills enhance use of e-services in the hospitality industry? A conditional mixed-process approach." *International Journal of Data and Network Science*, Vol. 5, No. 4, 2021.

Authors and affiliations

Ratna Wardani
e-mail: ratna@uny.ac.id
ORCID: https://orcid.org/0000-0002-7680-6487
Faculty of Engineering, Universitas Negeri Yogyakarta, Indonesia

Masduki Zakaria
e-mail: masduki_zakaria@uny.ac.id
ORCID: https://orcid.org/0000-1111-2222-333X
Faculty of Engineering, Universitas Negeri Yogyakarta, Indonesia

Tika Novita Sari
e-mail: tikanovitasari@uny.ac.id
ORCID: https://orcid.org/0009-0000-5877-5085
Faculty of Engineering, Universitas Negeri Yogyakarta, Indonesia

Dian Novian
e-mail: aadian@ung.ac.id
ORCID: https://orcid.org/0009-0000-5877-5085
Faculty of Engineering, Universitas Negeri Gorontalo, Indonesia

e-Informatica Software Engineering Journal
Volume 20, Issue 1, pages: 1–23
DOI: 10.37190/e-Inf260108
BibTeX

KPSO-Fuzzy: An AI-based Hybrid Approach for Software Requirements Prioritization

Rahila Anwar*, Muhammad Bilal Bashir
*Corresponding author: raheelaanwar27864@iqraisb.edu.pk

Article info

Keywords:
artificial intelligence
clustering
elbow method
fuzzy logic
genetic algorithm: hybrid technique
hybrid genetic algorithms
machine learning
multi-objective artificial bee colony optimization
particle swarm optimization
requirement engineering
requirement prioritization
requirement analysis

Submitted: 4 Mar. 2026
Revised: 13 Mar. 2026
Accepted: 7 May 2026
Available online: 12 May 2026

Abstract

Context: Software requirements outline customer expectations for their software and are critical for successful project outcomes. As software becomes increasingly complex due to its size and diverse features, it is vital to prioritize these requirements to utilize development resources effectively. To address this challenge, researchers are exploring new strategies and improved solutions using artificial intelligence (AI) tools. In our systematic literature review conducted in 2023, we found that existing requirements prioritization techniques predominantly rely on human input and have several limitations. These include inaccuracies, overlapping results, scalability issues, and excessive time consumption. These challenges can be mitigated by incorporating key features from AI-based software requirements prioritization techniques.
Objective: The primary goal of this study is to develop a Hybrid AI-based requirements prioritization technique named as "KPSO-Fuzzy that effectively balances user preferences and technical dependencies.
Method: We propose a Hybrid prioritization method called Hybrid KPSO-Fuzzy. First, we use K-Means clustering to group technically dependent functional requirements into three distinct clusters. In the next phase, we apply the Particle Swarm Optimization (PSO) algorithm along with Fuzzy Logic to prioritize each cluster simultaneously. Clusters that achieve higher accuracy will be selected to create the most effective prioritized lists of requirements. We conducted extensive experiments to validate our proposed approach, focusing on accuracy, scalability, and computation time.
Results: The comparative analysis shows that our proposed Hybrid KPSO-Fuzzy method outperforms PSO, Fuzzy Logic, Multi-objective Artificial Bee Colony optimization and Hybrid Genetic Algorithms in terms of accuracy, scalability, and efficiency.
Conclusions: Overall, this study clarifies the application domains for various AI-based techniques, enabling users to maximize their benefits.

1. Introduction

Requirement Engineering (RE) is a fundamental phase of software engineering that identifies system requirements by considering client needs, stakeholder analysis, development context, negotiated outcomes, and evolving requirement management [1–4]. Requirement

© 2026 The Authors. Published by Wrocław University of Science and Technology Publishing House.
This is an open access article under the CC BY-SA 4.0 license international.

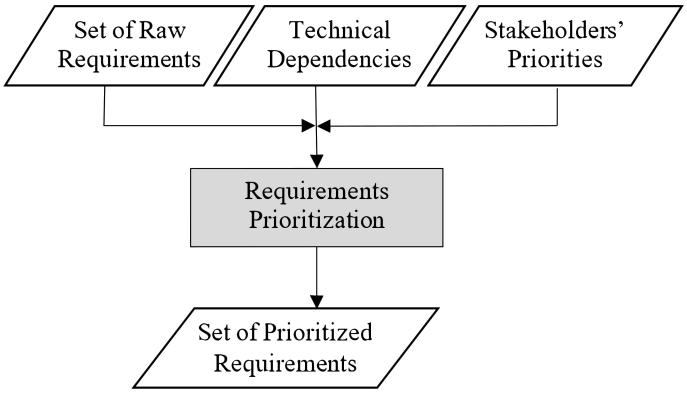


Figure 1. Software requirements prioritization process

Engineering also includes Requirement Prioritization, which supports analysis, negotiation, project planning, and timely software delivery [5–7]. Figure 1 presents the prioritization process involving inputs, processing, and outputs. Under limited resources, tight schedules, and high client expectations, prioritization enables effective evaluation of requirements while addressing time, cost, and resource constraints [4, 8, 9]. The execution order of requirements influences user experience and depends on technological and resource limitations, producing an ordered list that manages dependencies and feasible implementation sequences [4, 10–12]. Therefore, identifying critical requirements is essential [5, 13]. Many conventional requirement prioritization techniques have been proposed [1–3, 5, 8, 9, 14–16], including quantitative and subjective methods such as Analytical Hierarchy Process (AHP), Cumulative Voting, and Numerical Assignment. Ranking-based approaches are time- and cost efficient but limited in accuracy, while AHP requires expert involvement and high computational effort [5]. Existing techniques face challenges related to computational complexity, limited automation, scalability issues, ambiguity, uncertainty, and high computational cost, particularly in projects with conflicting risks and requirements [5, 6, 8, 15, 17]. Consequently, AI-based approaches – including fuzzy logic, genetic algorithms, simulated annealing, ant colony optimization, Artificial Bee Colony optimization, particle swarm optimization, and machine learning – have been introduced for requirement prioritization [14, 18–26].

Machine learning enables automated learning without explicit programming and improves scalability and prioritization accuracy compared to AHP [26, 27]. However, ML approaches still show disagreement in prioritization outcomes, including 4% disagreement [26], 10–20% disagreement [28], 74% recall with 16% precision [29], and 12–24% disagreement across criteria [27]. Fuzzy logic addresses imprecise and linguistic decision-making problems [30] and was introduced by Lotfi Zadeh [31]. Although scalable, fuzzy-based methods may lack optimal accuracy [2, 5, 30, 32–35]. Optimization algorithms also experience performance degradation as requirement size increases [9, 19, 21, 36–40] and may generate redundant or repeated solutions due to stochastic behavior [9, 11, 19, 36, 37, 39, 39–43].

In this study, two core parameters are considered: technical dependency [11, 43, 44] and user preferences [22, 37, 43]. Technical dependency captures interrelationships among requirements where implementation depends on prerequisite components [45, 46], while user preferences reflect stakeholder perceived importance and system value, aligning prioritization with user needs [19, 43]. These parameters are selected as generalizable inputs for computational prioritization frameworks [2, 5], as they can be quantified through expert elicitation, stakeholder surveys, or dependency matrices. Compared to domain-specific

factors such as resource constraints or volatility, empirical studies emphasize dependency management and user involvement as key drivers of effective prioritization [27, 45].

Although many studies evaluate prioritization approaches, no comprehensive divide and conquer framework integrating multiple AI-based techniques has been thoroughly investigated in terms of accuracy, scalability, and efficiency. Moreover, experimental guidance for selecting appropriate techniques remains limited. To address this gap, rigorous experimentation is conducted to obtain an effective hybrid prioritization solution.

The rest of this paper is structured as follows: in Section 2, we briefly discuss work done in similar areas by other researchers; in Section 3, we describe details of the research setup; in Section 4, we report the results of the study. Then, in Section 5, we discuss the collected results, as well as describe potential threats to the validity of the research, and we conclude the work in Section 6.

Research objective

The primary goal of this study is to develop a Hybrid AI-based requirements prioritization technique that balances user preferences and technical dependencies. The research integrates K-Means clustering, Fuzzy Logic, and Particle Swarm Optimization (PSO) to improve prioritization accuracy, efficiency, and scalability. The specific objectives are as follows:

1. **Conduct a literature review to analyze existing AI-based requirements prioritization techniques**, examining strengths, limitations, and evaluation parameters to identify research gaps guiding the hybrid technique development.
2. **Develop and implement a hybrid AI-based prioritization technique**, integrating technical dependency and user preference for functional requirement prioritization, evaluated using cross-validation based on accuracy, scalability, and efficiency.
3. **Perform a comparative analysis with state-of-the-art techniques** by benchmarking the proposed method against existing approaches using Dataset 1 and Dataset 2 with three parameter variants, assessing precision, scalability, efficiency, and statistical significance.

2. Related work

Software requirements prioritization is essential in requirements engineering to ensure early implementation of valuable and time-critical features. Traditional techniques such as MoSCoW, Analytic Hierarchy Process (AHP), and cumulative voting are widely used but face challenges related to uncertainty, scalability, and conflicting stakeholder preferences. To address these limitations, AI and soft-computing approaches integrating swarm intelligence, optimization, machine learning, and fuzzy logic have been explored to improve prioritization accuracy and adaptability. For example, Sadia and Faisal [32] proposed a fuzzy inference system for volatile requirement prioritization, Singh et al. [17] introduced a Hybrid LFPP–ANN model, Bisht and Kushwaha [47] incorporated stakeholder perspectives using fuzzy membership functions, and Abusaeed et al. [48] applied F-AHP for prioritizing cost factors in agile development. Jawale and Bhole [5] proposed hierarchical fuzzy cumulative voting but lacked scalability and explicit dependency modeling. Ahmad et al. [2] extended MoSCoW using fuzzy logic, though the approach relied heavily on subjective classification. Valsala and Nair [9] developed a Hybrid ILP–genetic model for release planning, achieving

effective optimization but with high computational cost and limited adaptability to dynamic requirements. In contrast, the proposed model introduces several enhancements:

- **Data-driven clustering using *K*-Means:** Unlike [5] and [2], clustering groups technically dependent requirements prior to prioritization, improving scalability and reducing cognitive complexity.
- **Optimization via PSO:** To address computational inefficiencies reported in [9], Particle Swarm Optimization (PSO) is applied to clustered data, enabling faster convergence and reduced time complexity.
- **Fuzzy inference for flexibility:** Fuzzy Logic integrates user preference and technical dependency, capturing subjective–objective trade-offs absent in [9] and [2].
- **Cross-validation and accuracy reporting:** The framework is evaluated on synthetic and real-world datasets, achieving over 95% accuracy with statistically validated performance metrics.

Additional studies have explored automation, machine learning, and dependency-aware prioritization. Shao et al. [44] proposed DRank integrating requirement dependencies using RankBoost and PageRank-Req, while Bollumpally et al. [23] introduced an XGBoost-based prioritization workflow that improves accuracy but requires retraining after system updates. Hujainah et al. [49] developed SRPTackle combining K-Means clustering and binary search trees for scalability, though with limited dependency handling, and Dang et al. [50] proposed MLPrior to reduce labeling costs through misclassification-based prioritization. Optimization-based approaches include a least-squares random genetic algorithm by Ahuja et al. [14], a parallel bee-foraging method by Alrezaamiri et al. [51], and a multi-objective PSO technique by Nazir et al. [25], achieving improved prioritization performance despite heuristic parameter tuning challenges. Hybrid optimization methods further combine complementary strengths of metaheuristics:

- **CDBR using PSO:** Gupta and Gupta [10] proposed a dependency-based collaborative prioritization approach achieving faster convergence than GA, though execution relations among dependencies were not considered.
- **Genetic algorithm with swarm intelligence:** Maghawry et al. [52] introduced a Hybrid GA–PSO framework incorporating K-Means-based selection and rehabilitation strategies to improve convergence and population diversity.
- **Hybrid swarm-based MOEA:** Marghny et al. [53] proposed HABCDE, integrating Artificial Bee Colony with Differential Evolution to solve the Next Release Problem, achieving improved Pareto-optimal solutions.

Although AI-based techniques improve requirement prioritization, many rely on complex black-box models that limit interpretability and real-time applicability. To address this, the proposed framework introduces a lightweight hybrid approach integrating K-Means clustering, PSO optimization, and fuzzy inference to jointly model technical dependencies and user preferences. However, real-world adoption of hybrid AI prioritization remains limited due to scarce benchmarking datasets, insufficient cross-validation, adaptive parameter tuning challenges [54], scalability issues, and inconsistent evaluation practices, highlighting the need for an adaptive, interpretable, and scalable framework as identified in recent studies [55] and summarized in Table 1.

Table 1. Strengths and limitations of AI-based RP techniques

Techniques	Strengths	Limitations
Fuzzy Logic-based RP	Scalable [2, 5, 32–35, 47]	Need improved accuracy [2, 5, 30, 32–35]
Optimization-based RP	Accurate results [11, 14, 43]	Less scalable [9, 19, 21, 36–40, 47]; Redundant solutions [9, 11, 19, 36, 37, 39–43]; Slow for large requirements [26]
Machine Learning-based RP	Scalable and optimal [10, 26–29, 44]; Less redundancy [23, 49]; Time-efficient [10, 26, 44]	Need improved accuracy [26–28]

3. Research method

A research design provides a structured framework guiding data collection and analysis for addressing specific research questions or hypotheses. The research paradigm determines how problems are formulated and methodologically addressed [56]. This study adopts an experimental quantitative research design within the Positivism paradigm, emphasizing objectivity and minimizing researcher bias to ensure knowledge independence from personal values and beliefs.

3.1. Research goal and questions/hypotheses

The primary objective is to develop a Hybrid AI-based framework for functional requirement prioritization through the integration of multiple AI techniques. The study addresses the following research questions (RQs):

1. **What are the strengths and limitations of existing AI-based software requirements prioritization techniques?**
2. **How can a Hybrid framework integrating *K*-Means clustering, Fuzzy Logic, and Particle Swarm Optimization (PSO) be designed and implemented for functional requirement prioritization?**
3. **Does the proposed Hybrid approach (KPSO-Fuzzy) improve accuracy, performance, and scalability compared to existing techniques?**

3.2. Research method/procedure

The proposed model operates as follows:

1. A set of n functional requirements is considered, each associated with technical dependency and user preference scores scaled on a five-point scale (1–5), where higher values indicate higher priority [50].
2. AI techniques including K-Means (Machine Learning), PSO (Optimization), and Fuzzy Logic were selected based on literature analysis and domain suitability [55].
3. **Phase 1: Requirement clustering using *K*-Means** K-Means clustering is applied during preprocessing to group requirements into three clusters (High, Medium, Low) based on technical dependency. This enables early identification of dependent requirements, reduces complexity through divide-and-conquer processing, and minimizes noise. The optimal number of clusters was determined as shown in Figure 2 using the Elbow Method [57]. The Within-Cluster Sum of Squares (WCSS) showed a clear elbow at

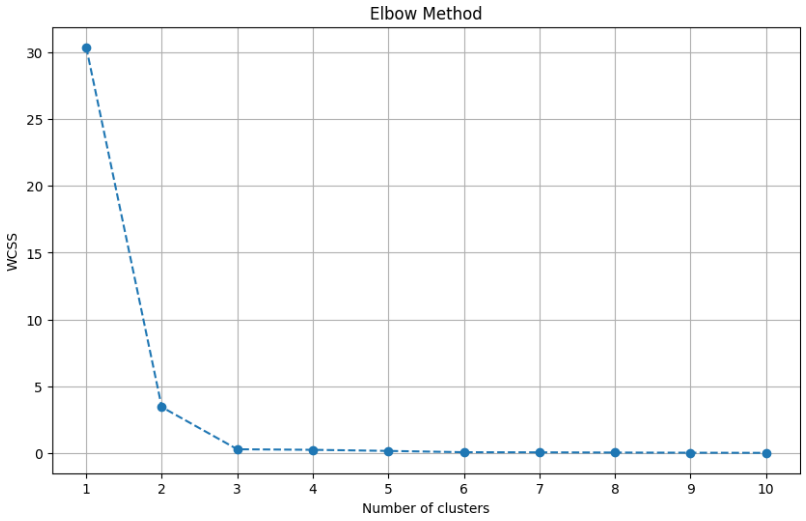


Figure 2. Elbow method for K-Means clustering

$K = 3$, providing the best balance between compactness and model complexity. This clustering improves PSO convergence and prioritization accuracy while enhancing fuzzy rule interpretability.

4. **Phase 2: Prioritization using Fuzzy Logic** PSO and Fuzzy Logic are applied independently and in parallel within each cluster. Fuzzy Logic prioritizes requirements using normalized Technical Dependency (TD) and User Preference (UP) inputs with trapezoidal membership functions [17, 47, 48]. Inputs are classified into Low, Medium, and High linguistic categories, producing a priority score through predefined fuzzy rules.
5. **Phase 3: Optimization with PSO** PSO identifies an optimal requirement sequence minimizing disagreement [27, 28] with a benchmark list derived from weighted TD and UP scores (e.g., 0.7 TD, 0.3 UP). Particle Swarm Optimization (PSO) is adapted to operate within a permutation-based search space to prevent requirement duplication in generated solutions. Each particle encodes a candidate priority sequence. After position updates, a repair mechanism identifies duplicate requirement IDs and replaces them with unassigned elements to preserve permutation feasibility. The adaptation builds upon the standard PSO framework [58, 59] while ensuring valid and non-redundant prioritization outcomes. The objective function minimizes positional mismatches:

```
def objective_function(particle_order, benchmark_order):
    return sum(p != b for p, b in zip(particle_order, benchmark_order))
```

6. **Accuracy and efficiency evaluation** Accuracy is computed using disagreement between prioritized lists and benchmark rankings. Cross-validation selects the highest cluster accuracy values, which are averaged to obtain final Hybrid accuracy. Efficiency is measured through computation time [34, 60], selecting the minimum time per cluster.
7. **Comparison across techniques** Outputs of PSO, Fuzzy Logic, and the Hybrid approach are compared using accuracy, scalability, and efficiency metrics. Ranking conflicts are resolved using normalized average priority scores and agreement thresholds, producing a merged global prioritized list.
8. **Statistical validation** Statistical significance is evaluated using paired t-tests [22] or Wilcoxon Signed-Rank tests [24, 27, 61], depending on data normality. Analysis includes

p-values, confidence intervals, and effect sizes to validate robustness against PSO, Fuzzy Logic, MOABC, and HGA methods.

The proposed AI-based Hybrid framework integrates K-Means clustering, PSO optimization, and Fuzzy Logic inference within a divide-and-conquer prioritization strategy to achieve accurate, scalable, and computationally efficient solutions. Figure 3 illustrates the sequential pipeline from clustering to optimization and fuzzy inference.

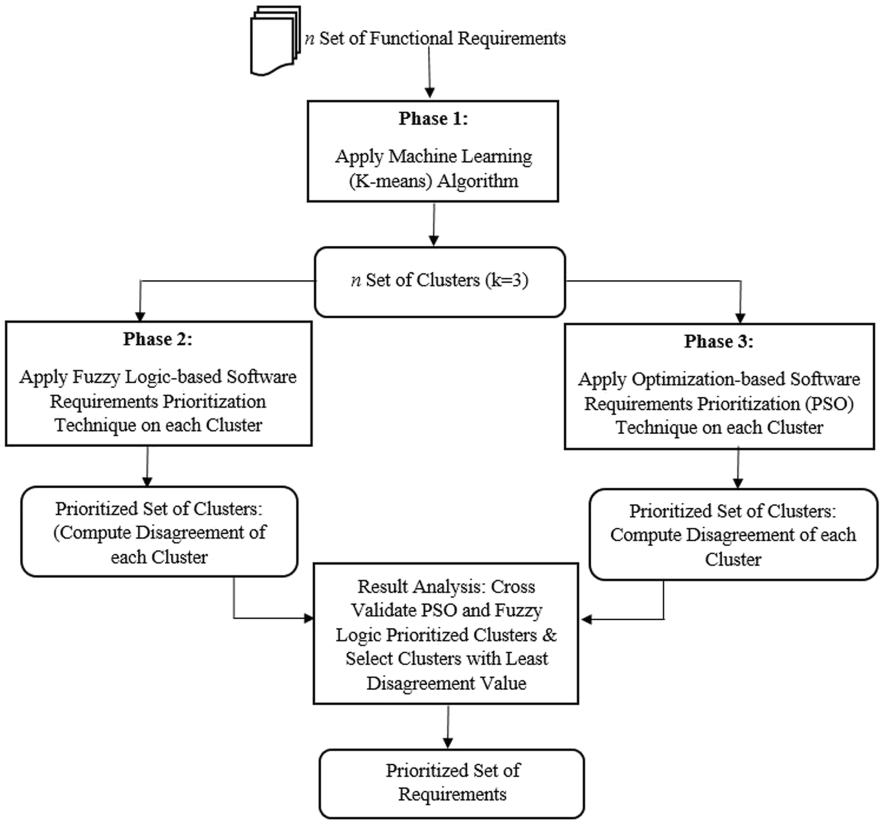


Figure 3. Proposed hybrid KPSO-fuzzy model for functional requirements prioritization

Dataset description

Two dataset categories were used to evaluate model performance and generalizability [20, 24]. **Dataset 1** contains 8083 functional requirements collected from 22 real-world case studies across domains such as Driver Drowsiness Detection, Smart Diet Monitoring, Parts Fitment, and Child Immunization systems. Each requirement includes technical dependency and user preference ratings on a five-point Likert scale [37]. **Dataset 2** is a synthetically generated dataset with TD and UP values uniformly distributed between 0 and 1, enabling controlled parameter tuning and sensitivity analysis of PSO and fuzzy

Table 2. Details of datasets used in the study

Dataset	Source	Type of requirements	Size of requirements
Dataset 1	22 Case Studies (Online)	Functional	Varying Size
Dataset 2	Random Numbers	Functional	Varying Size

inference mechanisms. Dataset sizes range from 20 to 2,000 requirements, representing realistic small-to-large software projects. A summary of dataset characteristics is presented in Table 2.

Evaluation measures

The proposed Hybrid KPSO-Fuzzy model was evaluated using three metrics: accuracy, efficiency, and scalability, selected based on established literature. Prior studies [10, 44] emphasize dependency minimization for improving prioritization accuracy; therefore, technical dependency and user preference were used as primary input criteria [22, 44, 51, 61]. Accuracy was measured through disagreement between predicted and benchmark rankings using normalized Kendall's Tau and Spearman's Rank Correlation, consistent with existing approaches [11, 23, 26–29, 43, 49, 60, 62]. Stakeholder–developer disagreement was also considered a precision indicator [10]. Efficiency was evaluated using average computation time required by PSO and fuzzy logic to achieve stable solutions [10, 13, 26, 29, 51, 63]. Scalability was analyzed across datasets of varying sizes to assess performance stability under increasing requirements [14, 34, 49]. All metrics were computed at the cluster level by independently evaluating PSO and fuzzy logic, while final hybrid performance combined maximum accuracy with minimum computation time across clusters.

4. Results

The proposed technique, Hybrid KPSO-Fuzzy (K-Means, PSO, and Fuzzy Logic), integrates clustering, optimization, and fuzzy inference for functional requirements prioritization. The process begins by initializing a population of solutions, which is provided as input to the K-Means algorithm [64] to group technically dependent requirements into three clusters. The resulting clusters are then used as input for both Particle Swarm Optimization (PSO) [25, 65] and fuzzy logic using a trapezoidal membership function [17] for further prioritization. Finally, the best solution obtained from PSO and fuzzy logic is selected as the final prioritization result and compared with standalone PSO and fuzzy logic approaches based on accuracy, efficiency, and scalability.

Algorithm implementation

In this section, we present the Hybrid KPSO-Fuzzy, K-Means, Fuzzy logic, and Particle Swarm Optimization (PSO) algorithms. These algorithms are divided into the following subsections:

Architecture of the proposed hybrid (KPSO-Fuzzy) technique

The Hybrid KPSO-Fuzzy tool simulates software requirements prioritization based on user preferences and functional dependencies to achieve an optimal solution. It provides an intuitive interface with results in tabular format, exportable as CSV files. The tool consists of three main modules: K-Means, PSO, and fuzzy logic, which share information to perform the simulation. Algorithm 1 presents the pseudo-code of the **Hybrid KPSO-Fuzzy**.

Algorithm 1. Pseudo-code for the hybrid KPSO-fuzzy technique

BEGIN
Step 1: Import Required Libraries
Import `Pandas`, `NumPy`, `Sklearn`, `Skfuzzy`, `Pyswarm`, `Matplotlib`, etc.
Step 2: Data Processing
Upload Excel file and load data into dataframe `df`
Rename columns to: `['Requirement ID','Technical Dependency','User Preference']`
Normalize values using `MinMaxScaler`
Step 3: Prioritization
for each `Requirement` in `df` **do**
    Assign priority label (High, Medium, Low) based on Technical Dependency and User Preference
end for
Step 4: Apply KMeans Clustering
Extract feature: `X = df[['Technical Dependency']]`
Initialize list: `wcss = []`
Set `max-clusters = 10`
for $k = 1$ to $max - clusters$ **do**
    Initialize KMeans with k clusters
    Fit model to `X`
    Append inertia to `wcss`
end for
Plot WCSS vs k and determine elbow point
Select optimal cluster number K (typically $K = 3$)
Apply final KMeans clustering
Step 5: Apply Fuzzy Logic Prioritization
for each cluster C_k in K **do**
    Define trapezoidal membership functions
    Compute fuzzy priority score
    Sort requirements by fuzzy priority
end for
Step 6: Apply PSO Prioritization
for each cluster C_k in K **do**
    Initialize particles
    Update velocities and positions
    Optimize ranking using disagreement metric
end for
Step 7: Plot Results
Plot Accuracy and Efficiency comparisons
END

Workflow of the K-Means model

The K-Means clustering algorithm is an iterative method that groups similar data points into K clusters based on their characteristics [64]. Algorithm 2 provides detailed information on how the K-Means model operates.

Workflow of Particle Swarm Optimization algorithm (PSO)

Particle Swarm Optimization (PSO) is a population-based optimization technique inspired by collective foraging behavior, where particles represent candidate solutions that iteratively update their positions and velocities by learning from personal and global best experiences.

Algorithm 2. K-Means-based prioritization of functional requirements

Begin
Step 1: Data Collection
Gather functional requirements, technical dependencies, and user preferences.
Step 2: Data Pre-processing
Normalize the dataset using MinMaxScaler from `sklearn.preprocessing`.
Scale values to the range [0, 1] for uniform processing.
Step 3: Import Libraries
Import necessary libraries including NumPy and KMeans from `sklearn.cluster`.
Step 4: Load Dataset
Load dataset (X, Y) containing scaled values (1 to 5) for technical dependency and user preference.
Step 5: Determine Optimal Clusters (Elbow Method)
Initialize empty list `WCSS = []` to store within-cluster sum of squares.
Set maximum number of clusters: `max-clusters = 10`.
for $k = 1$ to $max - clusters$ **do**
    Apply KMeans with k clusters: `KMeans(n-clusters=k, random-state=0, n-init='auto').fit(X)`
    Append the WCSS (inertia) to the list.
end for
Plot $WCSS$ vs. number of clusters.
Identify elbow point where WCSS decrease slows down.
Set $K = 3$ as optimal based on elbow point.
Step 6: Initialize *K*-Means
Randomly initialize K centroids.
Set the cluster size to 3 ($K = 3$ for High, Medium, Low priority).
Define KMeans parameters:
`kmeans = KMeans(n-clusters=3, random-state=0, n-init='auto').fit(X)`
Step 7: Assign Clusters
Assign each data point to the nearest centroid.
Recompute centroids by averaging points in each cluster.
Repeat until convergence to minimize squared distances.
Step 8: Priority Classification
High Priority: Features with significant impact based on user preferences and technical dependencies.
Medium Priority: Features with moderate impact.
Low Priority: Features with the least impact.
End

The velocity and position updates are defined as:

$$\begin{aligned} v_{ij}(t+1) &= w v_{ij}(t) + c_1 r_1 \left(x_{ij}^{p\text{-}best}(t) - x_{ij}(t)\right) \\ &\quad + c_2 r_2 \left(x_j^{g\text{-}best}(t) - x_{ij}(t)\right), \\ x_{ij}(t+1) &= x_{ij}(t) + v_{ij}(t+1). \end{aligned} \tag{1}$$

where $\mathbf{x}_i = (x_{i1}, \ldots, x_{iD})$ and $\mathbf{v}_i = (v_{i1}, \ldots, v_{iD})$ denote particle position and velocity in a D-dimensional space; $x_{ij}^{p\text{-}best}$ and $x_j^{g\text{-}best}$ are personal and global best positions. Parameters: $w = 0.4$, $c_1 = c_2 = 2.0$, $r_1, r_2 \in [0, 1]$, $t = 1, \ldots, T$, with velocity bounded by $v_{\max}$. PSO offers fast convergence with limited parameter tuning and effectively explores the solution space by guiding particles toward optimal solutions [25]. In this study, the `pyswarm` library was used to implement the PSO component for efficient functional requirement prioritization, with Algorithm 3 detailing the workflow.

Algorithm 3. PSO-based optimization of prioritized functional requirements

Begin
Step 1: Input Data from *K*-Means Model
Load the clustered requirements categorized as high, medium, and low priority.
Step 2: Initialize PSO Optimization Function
Define the `pso-optimize` function:
Initialize the swarm with random positions and velocities.
Update each particle's best-known position.
Update the global best-known position.
Adjust velocities and positions using cognitive and social components.
Step 3: Define Fitness Function
Define a fitness function to evaluate solutions.
The fitness function sums priorities, which can be customized for functional requirements.
Step 4: Initialize PSO Parameters
Define the number of particles and iterations.
Tune PSO parameters:
$w = 0.7$ ▷ Tuned inertia weight
$c_1 = 2.0$ ▷ Tuned cognitive coefficient
$c_2 = 2.0$ ▷ Tuned social coefficient
Step 5: Update Particle Positions
Update personal best positions and global best position.
Adjust velocities using inertia, cognitive, and social components.
Step 6: Evaluate Fitness
Evaluate the fitness of each particle to determine the best solution.
Step 7: Return Best Solution
Output the best solution found for functional requirement prioritization.
End

Workflow of fuzzy logic

Fuzzy logic is employed to prioritize functional requirements by balancing user preferences and technical dependencies while handling uncertainty inherent in decision-making. The implementation uses the `scikit-fuzzy` library (including `skfuzzy` and `skfuzzy.control`), enabling efficient construction and simulation of fuzzy inference systems for rapid prototyping and experimentation. A trapezoidal membership function is adopted due to its simplicity and flexibility [17, 47, 48], defined as:

$$\text{trapmf}(x; a, b, c, d) = \begin{cases} 0, & x < a \\ \dfrac{x-a}{b-a}, & a \leq x < b \\ 1, & b \leq x \leq c \\ \dfrac{d-x}{d-c}, & c < x \leq d \\ 0, & x > d \end{cases} \tag{2}$$

where a and d denote zero-membership boundaries, b and c define the plateau with full membership, and intermediate regions vary linearly. The proposed Hybrid KPSO-Fuzzy model applies a Mamdani fuzzy inference system [13, 66] to compute requirement priority scores using normalized Technical Dependency (TD) and User Preference (UP). Rule evaluation employs minimum and maximum operators for logical aggregation. The aggregated fuzzy outputs are defuzzified using the centroid (center-of-gravity) method [31, 67],

Algorithm 4. Fuzzy logic-based prioritization

Begin
Input: Data generated by K-Means model
Fuzzification: Convert numerical scores into fuzzy sets using membership functions
Define linguistic variables (e.g., Low, Medium, High) for both user preference and technical dependency
Define Fuzzy Rules:
Define fuzzy rules to model the relationship between user preference and technical dependency
Example: *"IF User Preference IS High AND Technical Dependency IS Low THEN Priority IS High"*
Inference Engine:
Use a fuzzy inference engine to evaluate fuzzy rules
Combine fuzzy sets according to defined rules using fuzzy operators
Defuzzification: Convert fuzzy output back into crisp scores using a defuzzification method
Prioritization: Rank functional requirements based on defuzzified scores
End

producing a crisp priority score:

$$y = \frac{\sum_{i=1}^{n} \mu_i \cdot x_i}{\sum_{i=1}^{n} \mu_i}$$

where x_i represents the output priority value and μ_i denotes the corresponding membership degree. The workflow of the fuzzy logic algorithm is clearly illustrated in Algorithm 4.

Comparative analysis of proposed Hybrid KPSO-Fuzzy vs PSO, Fuzzy Logic, MOABC, and Hybrid-GA

To evaluate the proposed Hybrid KPSO-Fuzzy technique, experiments were conducted on two datasets: Dataset 1, comprising 22 real-world case studies ensuring domain relevance, and Dataset 2, synthetically generated to simulate functional requirements of varying sizes and complexities. The model was compared against PSO, Fuzzy Logic, MOABC, and Hybrid Genetic Algorithm (HGA) across up to 10 runs per method to ensure statistical reliability. Three parameter configurations (V1–V3), varying PSO parameters (particles, iterations, inertia weight w, cognitive c_1, and social c_2 coefficients) and fuzzy membership ranges, were applied to examine robustness.

Comparative accuracy analysis across techniques

Figures 4–5 show that the **Hybrid KPSO-Fuzzy** approach consistently achieves the highest accuracy across both datasets and all requirement scales. For small datasets (20–100 requirements), it reaches ≥95%, outperforming PSO and Fuzzy Logic, while MOABC achieves 80%–95% and Hybrid-GA 52%–65%. On medium-scale datasets (620–1000 requirements), Hybrid KPSO-Fuzzy remains above 97%, with PSO at 88%–94%, MOABC near 82%, Fuzzy Logic 76%–83%, and Hybrid-GA below 60%. For large datasets (2000 requirements), it peaks at **96.67%** (Dataset 1) and **99.57%** (Dataset 2), whereas PSO reaches 87%–94%, MOABC ∼82%, Fuzzy Logic ∼68%, and Hybrid-GA ∼52%. Overall, the proposed method improves accuracy up to **47%** over Hybrid-GA, **17%** over MOABC, and at least **10%** over PSO, confirming its effectiveness for large-scale prioritization.

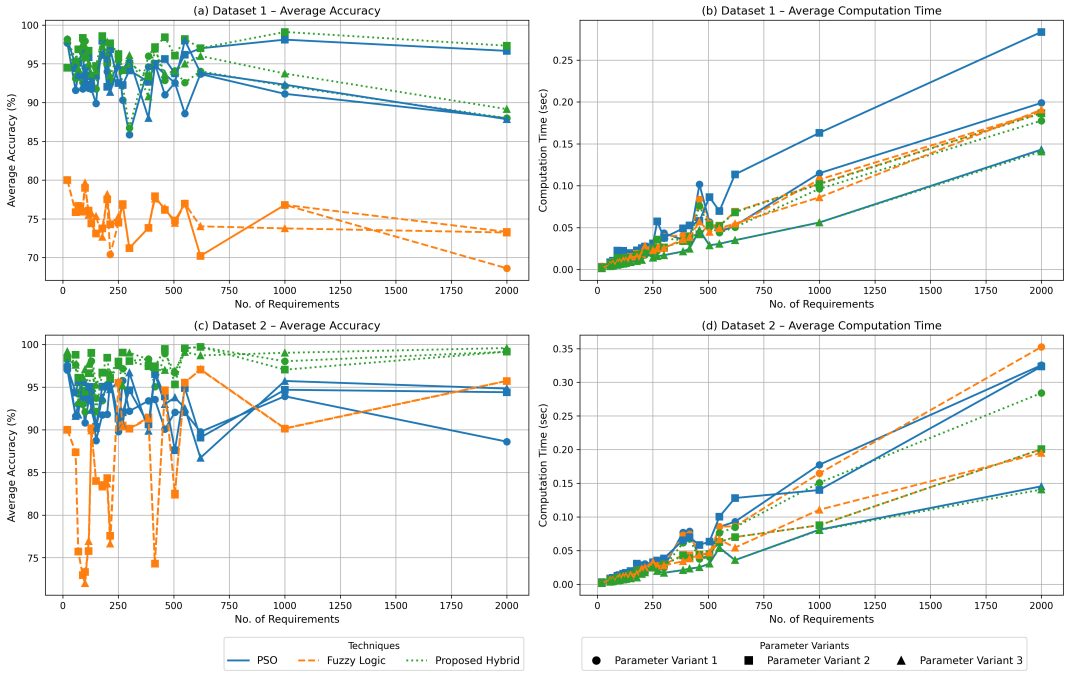


Figure 4. Performance trends of Hybrid KPSO–Fuzzy, PSO, and fuzzy logic across parameter variants 1–3

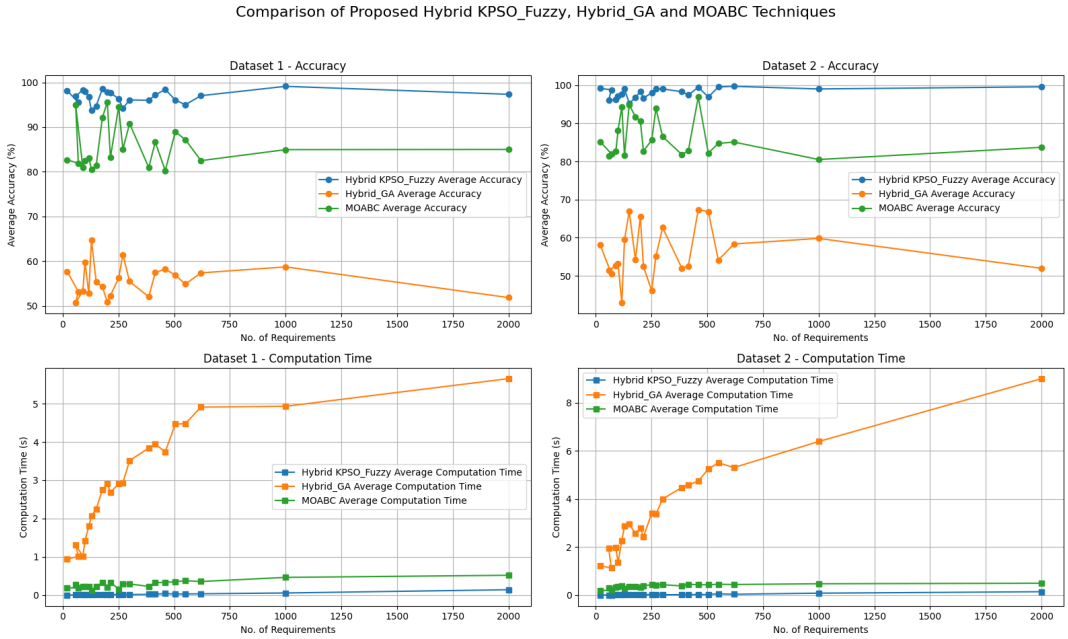


Figure 5. Comparison of hybrid KPSO-fuzzy, Hybrid-GA, and MOABC on dataset 1 and dataset 2

Comparative computation time analysis across techniques

For small datasets (20–100 requirements), computation times are minimal (0.03–0.08 s), with **Hybrid KPSO-Fuzzy** slightly faster than others – about 0.15–0.25 seconds faster than **Hybrid-GA** and 0.28 seconds faster than **MOABC**, also surpassing PSO and Fuzzy Logic. For medium datasets (620–1000 requirements), it remains most efficient (0.05–0.14 seconds for 1000 requirements) despite modest increases, while Hybrid-GA slows to 2.5–5.0 seconds and MOABC takes 0.18–0.35 seconds longer; PSO and Fuzzy Logic are faster but less accurate. For large datasets (2000 requirements), Hybrid KPSO-Fuzzy completes in 0.14–0.32 seconds, outperforming Hybrid-GA (6.5 seconds), MOABC ($\sim$ 0.52 seconds), and PSO/Fuzzy Logic (0.18–0.35 seconds). Overall, the proposed method demonstrates superior efficiency across all scales.

Scalability analysis for accuracy

Dataset 1: Fuzzy Logic shows poor scalability, declining from moderate accuracy for small datasets to 68%–76% for datasets over 1000 requirements. PSO maintains 87%–94% up to 1000 requirements but plateaus for larger sizes. MOABC is unstable on small datasets (80%–95%) and stabilizes at ∼82% above 1000 requirements. Hybrid-GA starts at 51%–66%, reaching ∼59% at 1000 and dropping to 52% at 2000. **Hybrid KPSO-Fuzzy** consistently outperforms all, maintaining 94%–98% for small datasets, 99% at 1000, and 96.67% at 2000, demonstrating robust scalability. **Dataset 2:** MOABC is unstable at small scales (80%–95%) and drops to ∼82% beyond 620 requirements. Hybrid-GA starts at 50%–67% for small datasets and declines to ∼52% at 2000 requirements, showing limited adaptability. **Hybrid KPSO-Fuzzy** consistently outperforms all, achieving 97%–98% at 1000 requirements and peaking at 99.57% for 2000, demonstrating robust scalability and high accuracy.

Scalability analysis for computational efficiency

Fuzzy Logic is the least efficient for large datasets due to variable computation times and declining accuracy, while PSO is faster but compromises precision. MOABC demonstrates limited scalability, taking 0.25–0.35 seconds for 20–100 requirements, rising to ∼0.55 seconds at 1000 and stabilizing at 0.6 seconds for 2000. Hybrid-GA shows steep time growth: Dataset 1 increases from 1.0–1.3 seconds for 100 requirements to over 4.8 seconds at 1000 and ∼5.8 seconds at 2000, while Dataset 2 rises from 1.5 seconds to above 6.5 seconds. In contrast, **Hybrid KPSO-Fuzzy** consistently delivers high efficiency, with Dataset 1 starting at 0.01 seconds, reaching 0.12–0.15 seconds at 1000 and remaining under 0.2 seconds at 2000, and Dataset 2 staying below 0.22 seconds. On average, it is 25–30 times faster than Hybrid-GA for datasets over 1000 requirements, highlighting superior computational efficiency and scalability.

Tables 3 and 4 show that proposed Hybrid KPSO-Fuzzy balances accuracy and computation time, making it the most scalable option for real-world implementation.

Statistical analysis

Statistical tests were conducted to evaluate the significance of the proposed Hybrid KPSO-Fuzzy technique by comparing its performance with Particle Swarm Optimization

Table 3. Performance breakdown of five prioritization techniques (hybrid KPSO-Fuzzy, Hybrid-GA, MOABC, PSO, and fuzzy logic)

Dataset size	Best technique	Worst technique	Observations
20–100 Req.	Hybrid KPSO-Fuzzy	Hybrid-GA/Fuzzy	Hybrid KPSO-Fuzzy shows a slight lead; Hybrid-GA and Fuzzy fluctuate more. MOABC performs moderately (80–95%). All techniques are relatively close.
620 Req.	Hybrid KPSO-Fuzzy	Hybrid-GA/Fuzzy	Hybrid's lead becomes clearer; MOABC remains unstable (82%), Hybrid-GA still underperforms especially in Dataset 2.
1000 Req.	Hybrid KPSO-Fuzzy	Hybrid-GA/Fuzzy	MOABC struggles to maintain accuracy (82%), Hybrid-GA shows lower stability; Hybrid KPSO-Fuzzy dominates.
2000 Req.	Hybrid KPSO-Fuzzy (99.57%)	Fuzzy (68.61%), Hybrid-GA ($\sim$ 55%)	MOABC remains around 82%, Hybrid-GA fails to scale effectively. Hybrid KPSO-Fuzzy is the only technique maintaining both accuracy and efficiency at scale.

Table 4. Comparison of Hybrid KPSO-Fuzzy, PSO, and fuzzy logic techniques across three parameter variants and two benchmark datasets

Parameter variant	Dataset	Highest accuracy	Lowest accuracy	Key observations
(10, 15, 0.7, 1.2, 1.2)	Dataset 1	Hybrid KPSO-Fuzzy Logic (92.15%)	Fuzzy Logic (76.8%)	Hybrid KPSO-Fuzzy outperforms PSO by 16%; Fuzzy Logic lags significantly.
	Dataset 2	Hybrid KPSO-Fuzzy (98.02%)	PSO (93.91%)	Hybrid KPSO-Fuzzy leads by 4% over PSO and 8% over Fuzzy.
(15, 15, 0.4, 1.5, 2)	Dataset 1	Hybrid KPSO-Fuzzy (99.11%)	Fuzzy Logic (76.8%)	Hybrid KPSO-Fuzzy achieves near-perfect accuracy, Fuzzy Logic under performs.
	Dataset 2	Hybrid KPSO-Fuzzy (97.04%)	Fuzzy Logic (90.13%)	Hybrid KPSO-Fuzzy is 3% ahead of PSO and 7% ahead of Fuzzy.
(10, 10, 0.3, 1.2, 1.9)	Dataset 1	Hybrid KPSO-Fuzzy (93.75%)	Fuzzy Logic (73.76%)	Hybrid KPSO-Fuzzy wins, with PSO slightly behind (92.31%).
	Dataset 2	Hybrid KPSO-Fuzzy (99.01%)	PSO (95.71%)	Hybrid KPSO-Fuzzy maintains the lead, Fuzzy Logic slightly better than PSO.

(PSO), Fuzzy Logic, Multi-Objective Artificial Bee Colony (MOABC) [51], and Hybrid Genetic Algorithm (HGA) [52] approaches. Normality of the experimental results was assessed using the Shapiro-Wilk test [44, 63]. Datasets satisfying the normality assumption ($p > 0.05$) were analyzed using paired t-tests, whereas non-normal distributions ($p \leq 0.05$) were evaluated using the Wilcoxon signed-rank test [24, 61]. All statistical evaluations were performed for both average accuracy and average computation time across Dataset 1 and Dataset 2.

Figures 6–7 present histogram distributions, boxplots, and Q-Q plots that illustrate the statistical behavior of the compared techniques. The sub-figures corresponding to Dataset 1 and Dataset 2 demonstrate consistent distribution patterns and validate the suitability of parametric and non-parametric hypothesis testing procedures. The hybrid KPSO-Fuzzy achieved statistically significant improvements in prioritization accuracy in both datasets compared to all baseline techniques ($p < 0.01$). The method also demonstrated consistently lower computation time and large effect sizes, confirming superior efficiency, robustness, and overall performance.

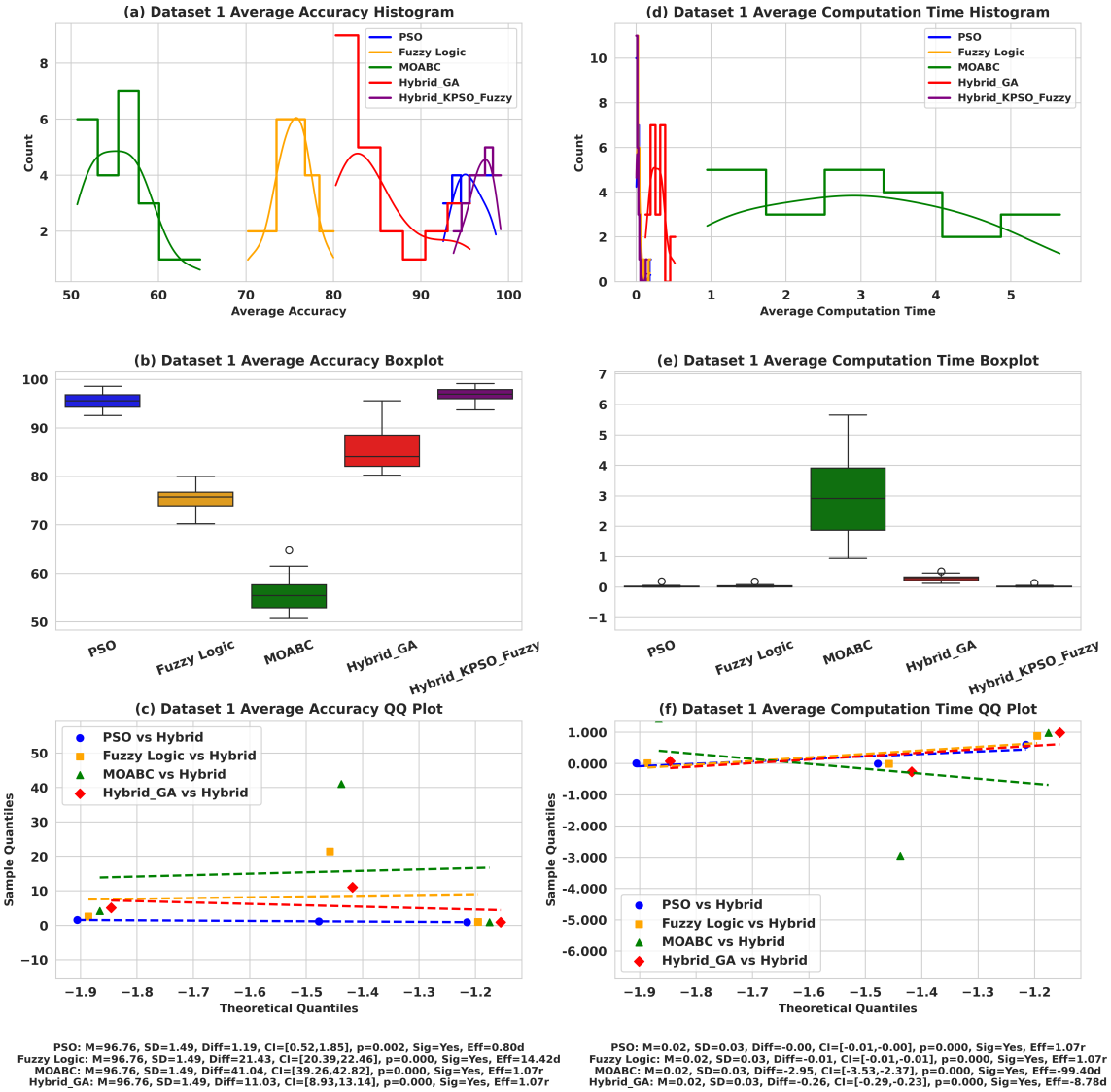


Figure 6. Statistical Analysis of Hybrid KPSO-Fuzzy versus baseline Techniques for Average Accuracy and Computation Time on Dataset 1

5. Discussion

This study investigates the Hybrid KPSO-Fuzzy technique, which combines K-Means clustering, PSO, and Fuzzy Logic for functional requirement prioritization. The approach is evaluated using accuracy, computation time, and scalability, and compared with PSO, Fuzzy Logic, MOABC, and Hybrid Genetic Algorithms (HGA).

The proposed Hybrid KPSO-Fuzzy method demonstrates superior overall performance, achieving up to 99.57% accuracy while maintaining stable scalability and low computation time as dataset size increases. The integration of clustering, swarm optimization, and fuzzy reasoning enhances prioritization stability and robustness. Performance gains are less pronounced for small datasets (20–100 requirements), and the hybrid design introduces higher implementation complexity compared with standalone techniques.

PSO provides efficient computation and relatively stable accuracy but shows performance degradation beyond 1000 requirements due to local optima tendencies. Fuzzy Logic performs

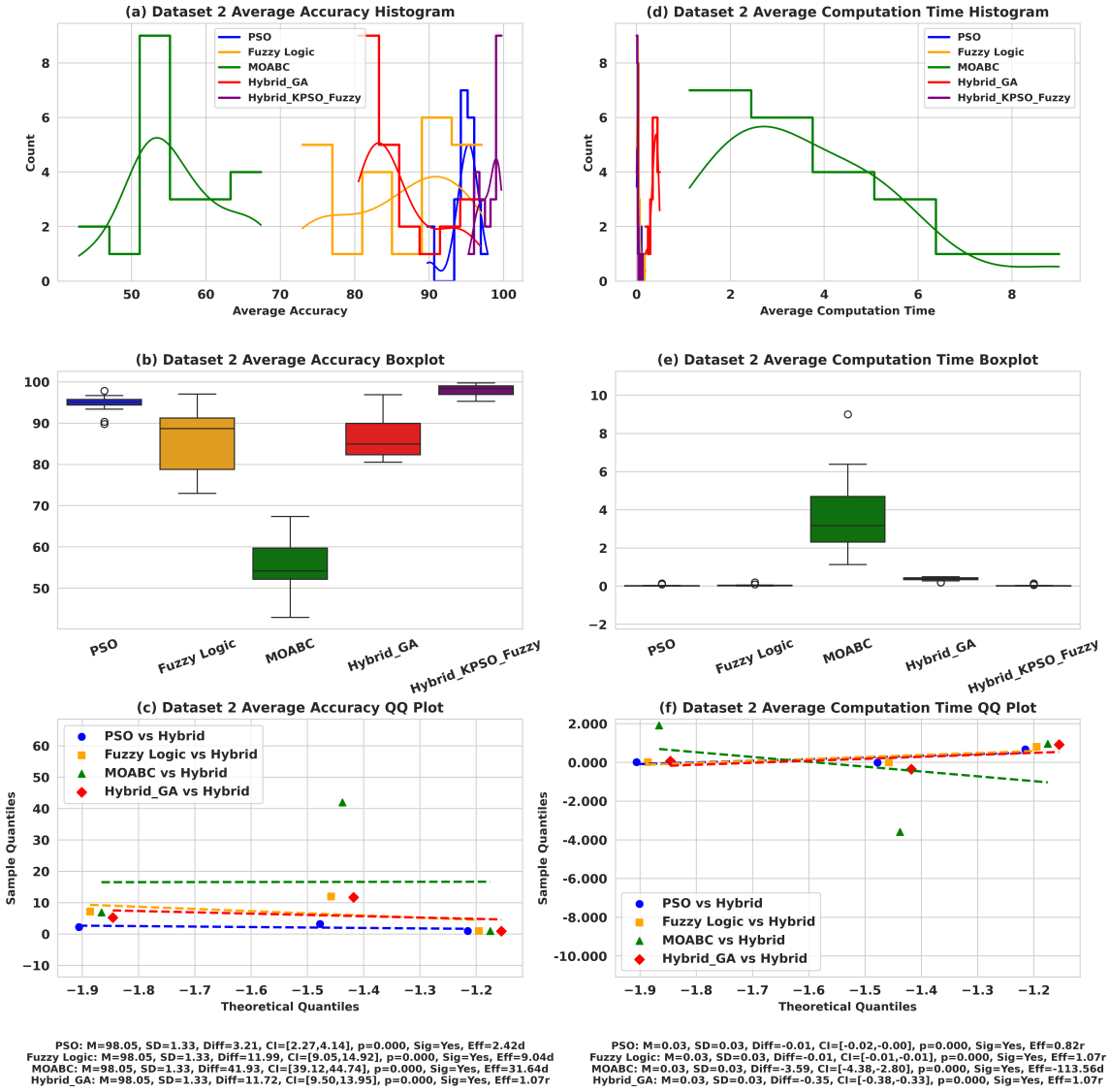


Figure 7. Statistical Analysis of Hybrid KPSO-Fuzzy versus baseline Techniques for Average Accuracy and Computation Time on Dataset 2

well for small datasets because of its simplicity and speed; however, accuracy decreases significantly for large-scale problems (down to 68.61%), indicating limited scalability. MOABC achieves acceptable results for small and medium datasets but exhibits accuracy stagnation near 82% and increased computation cost at larger scales. HGA shows predictable evolutionary behavior but records the lowest accuracy (below 65%) and the highest computation time, making it unsuitable for large requirement sets.

Overall, Hybrid KPSO-Fuzzy offers the best balance between accuracy, efficiency, and scalability. Compared with existing approaches, which often suffer from instability or computational overhead caused by randomization [19–21, 36, 38, 40, 41, 51], the proposed framework provides a robust and scalable solution for large-scale requirement prioritization.

Threats to validity

Several factors may influence the validity of the study outcomes.

Construct validity is limited by the scope of the research questions; this was mitigated through systematic categorization and analysis of fuzzy logic-, optimization-, and machine learning–based prioritization methods.
Data collection validity may be affected by incomplete retrieval of studies; therefore, multiple repositories and carefully selected keywords were used, although results may vary with larger datasets or alternative techniques.
Internal validity may be affected by constructing the benchmark priority list from the same input factors – technical dependency (TD) and user preference (UP) – used by the proposed model, introducing potential circularity in evaluation. However, the benchmark is used as a structured reference baseline rather than an absolute ground truth. This limitation is mitigated through evaluation across multiple datasets and by the stochastic, hybrid nature of the approach. Comparative analysis with independent techniques further supports the robustness of the results.
Dataset scaling within the range 1–5 may influence algorithm behavior, where increased variability could reduce fuzzy accuracy while improving clustering performance.
Metric variability arises from differences in requirement characteristics, algorithms, and project contexts; consistent evaluation criteria were applied to minimize this effect.
Evaluation scope focuses on objective metrics (accuracy, computation time, and scalability), while stakeholder-oriented measures remain future work.

The study also considers only two input criteria (user preference and technical dependency), adopts a fixed cluster size of three, and evaluates primarily functional requirements, which may limit generalizability. Despite these constraints, controlled experimental design and consistent evaluation ensure reliable comparative insights for AI-based requirement prioritization.

6. Conclusions

This study proposes the **Hybrid KPSO-Fuzzy** technique, a robust requirement prioritization framework that effectively balances *accuracy*, *computational efficiency*, and *scalability* by integrating technical dependency and user preferences. The approach follows three phases: K-Means clustering groups requirements, PSO and Fuzzy Logic refine prioritization, and cross-validation evaluates performance across key metrics. Experiments on two datasets demonstrate that Hybrid KPSO-Fuzzy consistently outperforms PSO, Fuzzy Logic, MOABC, and Hybrid-GA. Specifically, compared to PSO and Fuzzy Logic, Hybrid KPSO-Fuzzy improves accuracy by 3–5% and 15–27%, respectively, while reducing computation time by up to 10% and 5%. Against MOABC, it achieves up to 99.57% precision with average accuracy gains of 11–12% and faster computation across small, medium, and large datasets. Compared to Hybrid-GA, the technique delivers dramatic accuracy improvements of 41–42% and reduces computation time by 2.95–3.59 seconds on average. Overall, Hybrid KPSO-Fuzzy provides a stable, scalable, and efficient solution for large-scale software requirement prioritization, overcoming the limitations of existing AI-based approaches.

Future works

Future research can focus on the following directions:

- Evaluate the Hybrid KPSO-Fuzzy technique using additional factors such as cost, redundancy, stakeholder preferences, and diverse requirement types, with dynamic parameter adaptation for real-time environments.
- Explore machine learning-based prioritization and adaptive weight optimization, including integration with PSO, MOABC, or deep learning for large-scale datasets.
- Address broader dependency types, including contribution and business dependencies, and improve scalability, efficiency, and redundancy reduction in optimization-based methods.
- Enhance fuzzy logic systems through automated rule generation and alternative membership functions (e.g., triangular, Gaussian) to improve accuracy and adaptability.
- Validate the approach using expert-ranked priorities and industrial case studies, with adaptive benchmark weight selection for project-specific optimization.

Overall, future research should focus on improving accuracy, scalability, efficiency, redundancy reduction, consistency ratio, self-adaptability, and the identification of optimal prioritization solutions for real-world applications.

CRediT authorship contribution statement

Rahila Anwar: Data curation, Methodology, Investigation, Writing – original draft, Writing – review and editing, Visualization. Muhammad Bilal Bashir: Conceptualization, Funding acquisition, Methodology, Investigation, Writing – review and editing, supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

This research received no external funding.

References

[1] M. Gen and L. Lin, "Genetic algorithms," in *Wiley Encyclopedia of Computer Science and Engineering*. American Cancer Society, 2008, pp. 1–15.

[2] K.S. Ahmad, N. Ahmad, H. Tahir, and S. Khan, "Fuzzy_MoSCoW: A fuzzy based MoSCoW method for the prioritization of software requirements," in *International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*. Kerala State,Kannur, India: IEEE, 2017, pp. 433–437.

[3] P. Achimugu, A. Selamat, R. Ibrahim, and M.N. Mahrin, "A systematic literature review of software requirements prioritization research," *Information and Software Technology*, Vol. 56, No. 6, 2014, pp. 568–585.

[4] F. Hujainah, R.B.A. Bakar, M.A. Abdulgabber, and K.Z. Zamli, "Software requirements prioritisation: A systematic literature review on significance, stakeholders, techniques and challenges," *IEEE Access*, Vol. 6, 2018, pp. 71 497–71 523.

[5] B. Jawale and A. Bhole, "Adaptive fuzzy hierarchical cumulative voting: A novel approach toward requirement prioritization," *International Journal of Research in Engineering and Technology*, Vol. 04, No. 05, 2015, pp. 365–370.

[6] A. Hudaib, M.H. Qasem, and N. Obeid, "FIPA-based semi-centralized protocol for negotiation," in *Cybernetics Approaches in Intelligent Systems*, Advances in Intelligent Systems and Computing, R. Silhavy, P. Silhavy, and Z. Prokopova, Eds. Cham: Springer International Publishing, 2018, pp. 135–149.

[7] J. Dąbrowski, E. Letier, A. Perini, and A. Susi, "Analysing app reviews for software engineering: A systematic literature review," *Empirical Software Engineering*, Vol. 27, No. 2, 2022, p. 43. [Online]. https://link.springer.com/10.1007/s10664-021-10065-7

[8] L. Karlsson, T. Thelin, B. Regnell, P. Berander, and C. Wohlin, "Pair-wise comparisons versus planning game partitioning—experiments on requirements prioritisation techniques," *Empirical Software Engineering*, Vol. 12, No. 1, 2007, pp. 3–33.

[9] S. Valsala and D.A.R. Nair, "Requirement prioritization and scheduling in software release planning using hybrid enriched genetic revamped integer linear programming model," *Research Journal of Applied Sciences, Engineering and Technology*, Vol. 12, No. 3, 2016, pp. 347–354.

[10] A. Gupta and C. Gupta, "CDBR a semi-automated collaborative execute-before-after dependency-based requirement prioritization approach," *Journal of King Saud University – Computer and Information Sciences*, 2018, p. S1319157818304518.

[11] P. Tonella, A. Susi, and F. Palma, "Interactive requirements prioritization using a genetic algorithm," *Information and Software Technology*, Vol. 55, No. 1, 2013, pp. 173–187.

[12] M.I. Babar, M. Ghazali, D.N. Jawawi, S.M. Shamsuddin, and N. Ibrahim, "PHandler: An expert system for a scalable software requirements prioritization process," *Knowledge-Based Systems*, Vol. 84, 2015, pp. 179–202. [Online]. https://linkinghub.elsevier.com/retrieve/pii/S0950705115001483

[13] K. Gulzar, J. Sang, M. Ramzan, and M. Kashif, "Fuzzy approach to prioritize usability requirements conflicts: An experimental evaluation," *IEEE Access*, Vol. 5, 2017, pp. 13 570–13 577.

[14] H. Ahuja, Sujata, and U. Batra, "Performance enhancement in requirement prioritization by using least-squares-based random genetic algorithm," in *Innovations in Computational Intelligence : Best Selected Papers of the Third International Conference on REDSET 2016*, Studies in Computational Intelligence, B. Panda, S. Sharma, and U. Batra, Eds. Singapore: Springer, 2018, pp. 251–263.

[15] R. Qaddoura, A. Abu-Srhan, M.H. Qasem, and A. Hudaib, "Requirements prioritization techniques review and analysis," in *International Conference on New Trends in Computing Sciences (ICTCS)*. Amman: IEEE, 2017, pp. 258–263.

[16] M. Pergher and B. Rossi, "Requirements prioritization in software engineering: A systematic mapping study," in *3rd International Workshop on Empirical Requirements Engineering (EmpiRE)*, 2013, pp. 40–44.

[17] Y.V. Singh, B. Kumar, S. Chand, and D. Sharma, "A hybrid approach for requirements prioritization using logarithmic fuzzy trapezoidal approach (LFTA) and artificial neural network (ANN)," in *Futuristic Trends in Network and Communication Technologies*, Communications in Computer and Information Science, P.K. Singh, M. Paprzycki, B. Bhargava, J.K. Chhabra, N.C. Kaushal et al., Eds. Singapore: Springer, 2019, pp. 350–364.

[18] M. Ramzan, M. Jaffar, and A. Shahid, "Value based intelligent requirement prioritization (VIRP): Expert driven fuzzy logic based prioritization technique," *International Journal Of Innovative Computing, Information And Control*, Vol. 7(3), 2011, pp. 1017–1038.

[19] J.C. Quiroz, S.J. Louis, A. Shankar, and S.M. Dascalu, "Interactive genetic algorithms for user interface design," in *Congress on Evolutionary Computation*, 2007, pp. 1366–1373.

[20] Y. Zhang, M. Harman, and S.A. Mansouri, "The multi-objective next release problem," in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07. London, England: Association for Computing Machinery, 2007, pp. 1129–1137.

[21] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, 2002, pp. 182–197.

[22] M.H. Marghny, H.M. El-Hawary, and W. Dukhan, "An effective method of systems requirement optimization based on genetic algorithms," *Information Sciences Letters*, Vol. 6, No. 1, 2017, pp. 15–28.
[23] "A Machine Learning Approach to Workflow Prioritization," in *2019 Systems and Information Engineering Design Symposium (SIEDS)*. Charlottesville, VA, USA: IEEE, 2019, pp. 1–5. [Online]. https://ieeexplore.ieee.org/document/8735589/
[24] J.T. de Souza, C.L.B. Maia, T.d.N. Ferreira, R.A.F.d. Carmo, and M.M.A. Brasil, "An ant colony optimization approach to the software release planning with dependent requirements," in *Search Based Software Engineering*, Lecture Notes in Computer Science, M.B. Cohen and M. Ó Cinnéide, Eds. Berlin, Heidelberg: Springer, 2011, pp. 142–157.
[25] M. Nazir, A. Mehmood, W. Aslam, Y. Park, G.S. Choi et al., "A multi-goal particle swarm optimizer for test case prioritization," *IEEE Access*, Vol. 11, 2023, pp. 90 683–90 697. [Online]. https://ieeexplore.ieee.org/document/10223041/
[26] P. Avesani, S. Ferrari, and A. Susi, "Case-based ranking for decision support systems," in *International Conference on Case Based Reasoning*. Springer, Berlin, Heidelberg: Springer, 2003, pp. 35–49.
[27] A. Perini, A. Susi, and P. Avesani, "A machine learning approach to software requirements prioritization," *IEEE Transactions on Software Engineering*, Vol. 39, No. 4, 2013, pp. 445–461.
[28] P. Avesani, C. Bazzanella, A. Perini, and A. Susi, "Facing scalability issues in requirements prioritization with machine learning techniques," in *13th IEEE International Conference on Requirements Engineering (RE'05)*, 2005, pp. 297–305.
[29] C. Duan, P. Laurent, J. Cleland-Huang, and C. Kwiatkowski, "Towards automated requirements prioritization and triage," *Requirements Engineering*, Vol. 14, No. 2, 2009, pp. 73–89.
[30] M. Sadiq and S.K. Jain, "Applying fuzzy preference relation for requirements prioritization in goal oriented requirements elicitation process," *International Journal of System Assurance Engineering and Management*, Vol. 5, No. 4, 2014, pp. 711–723.
[31] L. Zadeh, "Fuzzy sets," *Information and Control*, Vol. 8, No. 3, 1965, pp. 338–353. [Online]. https://linkinghub.elsevier.com/retrieve/pii/S001999586590241X
[32] H. Sadia, S. Abbas, and M. Faisal, "Volatile requirement prioritization: A fuzzy based approach," *International Journal of Engineering and Advanced Technology*, Vol. 8, No. 5, 2019.
[33] M. Dabbagh and S.P. Lee, "An approach for prioritizing NFRs according to their relationship with FRs," *Lecture Notes on Software Engineering*, Vol. 3, No. 1, 2015, pp. 1–5.
[34] D. Mougouei, D.M. Powers, and E. Mougouei, "A fuzzy framework for prioritization and partial selection of security requirements in software projects," *Journal of Intelligent and Fuzzy Systems*, Vol. 37, No. 2, 2019, pp. 2671–2686.
[35] A. Ejnioui, C.E. Otero, and A.A. Qureshi, "Software requirement prioritization using fuzzy multi-attribute decision making," in *Conference on Open Systems*, 2012, pp. 1–6.
[36] A. Bagnall, V. Rayward-Smith, and I. Whittley, "The next release problem," *Information and Software Technology*, Vol. 43, No. 14, 2001, pp. 883–890.
[37] J. del Sagrado, I.M. del Águila, and F.J. Orellana, "Ant colony optimization for the next release problem: A comparative study," in *2nd International Symposium on Search Based Software Engineering*, 2010, pp. 67–76.
[38] D. Greer and G. Ruhe, "Software release planning: an evolutionary and iterative approach," *Information and Software Technology*, Vol. 46, No. 4, 2004, pp. 243–253.
[39] M. Harman, A. Skaliotis, K. Steinhöfel, and P. Baker, "Search–based approaches to the component selection and prioritization problem," in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06. Seattle, Washington, USA: Association for Computing Machinery, 2006, pp. 1951–1952.
[40] M. Feather and T. Menzies, "Converging on the optimal attainment of requirements," in *Proceedings IEEE Joint International Conference on Requirements Engineering*. Essen, Germany: IEEE Comput. Soc, 2002, pp. 263–270.
[41] M.O. Saliu and G. Ruhe, "Bi-objective release planning for evolving software systems," in *Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the*

ACM SIGSOFT Symposium on the Foundations of Software Engineering. Dubrovnik, Croatia: ACM Press, 2007, p. 105.

[42] A. Finkelstein, M. Harman, S.A. Mansouri, J. Ren, and Y. Zhang, "A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making," *Requirements Engineering*, Vol. 14, No. 4, 2009, pp. 231–245.

[43] P. Tonella, A. Susi, and F. Palma, "Using interactive GA for requirements prioritization," in *2nd International Symposium on Search Based Software Engineering*, 2010, pp. 57–66.

[44] F. Shao, R. Peng, H. Lai, and B. Wang, "DRank: A semi-automated requirements prioritization method based on preferences and dependencies," *Journal of Systems and Software*, Vol. 126, 2017, pp. 141–156.

[45] D. Ameller, M. Galster, P. Avgeriou, and X. Franch, "A survey on quality attributes in service-based systems," *Software Quality Journal*, Vol. 24, No. 2, 2016, pp. 271–299. [Online]. http://link.springer.com/10.1007/s11219-015-9268-4

[46] A. Al-Adwan and A. Aladwan, "Using Interdependencies for the Prioritization and Reprioritization of Requirements in Incremental Development," *International Journal of Advanced Computer Science and Applications*, Vol. 11, No. 11, 2020. [Online]. http://thesai.org/Publications/ViewPaper?Volume=11&Issue=11&Code=IJACSA&SerialNo=29

[47] A. Bisht and M. Kushwaha, "Parameter optimization of software requirement by using fuzzy algebra," 2020.

[48] S. Abu Saeed, S.U.R. Khan, and A. Mashkoor, "A fuzzy AHP-based approach for prioritization of cost overhead factors in agile software development," *SSRN Electronic Journal*, 2022. [Online]. https://www.ssrn.com/abstract=4237372

[49] F. Hujainah, R. Binti Abu Bakar, A.B. Nasser, B. Al-haimi, and K.Z. Zamli, "SRPTackle: A semi-automated requirements prioritisation technique for scalable requirements of software system projects," *Information and Software Technology*, Vol. 131, 2021, p. 106501. [Online]. https://linkinghub.elsevier.com/retrieve/pii/S0950584920302433

[50] X. Dang, Y. Li, M. Papadakis, J. Klein, T.F. Bissyandé et al., "Test input prioritization for machine learning classifiers," *IEEE Transactions on Software Engineering*, Vol. 50, No. 3, 2024, pp. 413–442. [Online]. https://ieeexplore.ieee.org/document/10382258/

[51] H. Alrezaamiri, A. Ebrahimnejad, and H. Motameni, "Parallel multi-objective artificial bee colony algorithm for software requirement optimization," *Requirements Engineering*, Vol. 25, No. 3, 2020, pp. 363–380.

[52] A. Maghawry, R. Hodhod, Y. Omar, and M. Kholief, "An approach for optimizing multi-objective problems using hybrid genetic algorithms," *Soft Computing*, Vol. 25, No. 1, 2021, pp. 389–405. [Online]. https://link.springer.com/10.1007/s00500-020-05149-3

[53] M. Marghny, E.A. Zanaty, W.H. Dukhan, and O. Reyad, "A hybrid multi-objective optimization algorithm for software requirement problem," *Alexandria Engineering Journal*, Vol. 61, No. 9, 2022, pp. 6991–7005. [Online]. https://linkinghub.elsevier.com/retrieve/pii/S111001682100853X

[54] N. Tasneem, H.B. Zulzalil, and S. Hassan, "Enhancing agile software development: A systematic literature review of requirement prioritization and reprioritization techniques," *IEEE Access*, Vol. 13, 2025, pp. 32 993–33 034. [Online]. https://ieeexplore.ieee.org/document/10876147/

[55] R. Anwar and M.B. Bashir, "A systematic literature review of ai-based software requirements prioritization techniques," *IEEE Access*, Vol. 11, 2023, pp. 143 815–143 860.

[56] T.A. O'Donoghue, *Planning your qualitative research project: An introduction to interpretivist research in education*. London; New York: Routledge, 2007.

[57] G.M. James, D. Witten, T.J. Hastie, and R. Tibshirani, *An introduction to statistical learning: with applications in R*, 6th ed., Springer texts in statistics. New York: Springer Springer Science+Business Media, 2013, No. 103.

[58] J. Kennedy and R.C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*. Perth, Australia: IEEE, 1995, pp. 1942–1948.

[59] Y. Shi and R.C. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE Congress on Evolutionary Computation*. Anchorage, AK, USA: IEEE Press, 1998, pp. 69–73.

[60] P. Achimugu, A. Selamat, and R. Ibrahim, "Using the fuzzy multi-criteria decision making approach for software requirements prioritization," *Jurnal Teknologi*, Vol. 77, No. 13, 2015. [Online]. https://journals.utm.my/index.php/jurnalteknologi/article/view/6321

[61] J.J. Durillo, Y. Zhang, E. Alba, and A.J. Nebro, "A study of the multi-objective next release problem," in *1st International Symposium on Search Based Software Engineering*, 2009, pp. 49–58.

[62] P. Achimugu and A. Selamat, "A hybridized approach for prioritizing software requirements based on K-Means and evolutionary algorithms," in *Computational Intelligence Applications in Modeling and Control*, A.T. Azar and S. Vaidyanathan, Eds. Cham: Springer International Publishing, 2015, Vol. 575, pp. 73–93. [Online]. http://link.springer.com/10.1007/978-3-319-11017-2_4

[63] J.M. Chaves-González and M.A. Pérez-Toledano, "Differential evolution with Pareto tournament for the multi-objective next release problem," *Applied Mathematics and Computation*, Vol. 252, 2015, pp. 1–13.

[64] B. Kumar, U.K. Tiwari, D.C. Dobhal, and H.S. Negi, "User story clustering using K-Means algorithm in agile requirement engineering," in *International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES)*. Greater Noida, India: IEEE, 2022, pp. 1–5. [Online]. https://ieeexplore.ieee.org/document/9844390/

[65] K. Lachhwani, "A comprehensive review analysis on PSO and GA techniques for mathematical programming problems," in *Proceedings of International Conference on Computational Intelligence*, R. Tiwari, M.F. Pavone, and R. Ravindranathan Nair, Eds. Singapore: Springer Nature, 2023, pp. 461–476.

[66] E. Mamdani, "Application of fuzzy algorithms for control of simple dynamic plant," *Proceedings of the Institution of Electrical Engineers*, Vol. 121, No. 12, 1974, p. 1585. [Online]. https://digital-library.theiet.org/content/journals/10.1049/piee.1974.0328

[67] N.R. Pal and S.K. Pal, "A review on image segmentation techniques," *Pattern Recognition*, Vol. 26, No. 9, 1993, pp. 1277–1294. [Online]. https://linkinghub.elsevier.com/retrieve/pii/003132039390135J

Authors and affiliations

Rahila Anwar
e-mail: raheelaanwar27864@iqraisb.edu.pk
ORCID: https://orcid.org/0009-0002-5095-9474
Computing & Technology Department, IQRA University, Islamabad, 44000 Pakistan

Muhammad Bilal Bashir
e-mail: bilal.bashir@iqraisb.edu.pk
ORCID: https://orcid.org/0000-0002-5938-6361
Computing & Technology Department, IQRA University, Islamabad, 44000 Pakistan

e-Informatica Software Engineering Journal
Volume 20, Issue 1, pages: 1–15
DOI: 10.37190/e-Inf260109
BibTeX

A Modular Multi-Agent LLM Architecture for Text-to-Diagram Generation and User-Guided Refinement

Hartwig Grabowski*
*Corresponding author: hartwig.grabowski@hs-offenburg.de

Article info

Dataset link: https://zenodo.org/records/17741490

Keywords:
diagram generation
multi-agent LLM pipeline
engineering informatics
automated modeling
computational design

Submitted: 24 Feb. 2026
Accepted: 6 May 2026
Available online: 13 May 2026

Abstract

Context: Recent advances in LLM-based diagram generation increasingly rely on coordinated agent systems rather than single-model prompts.
Objective: This work highlights how modular multi-agent architectures improve reliability, semantic grounding, and iterative refinement in text-to-diagram workflows.
Method: We analyze a pipeline composed of specialized agents for interpretation, synthesis, validation, and correction, each contributing a bounded and inspectable transformation.
Results: The agent system provides deterministic validation, structured reasoning, and controlled refinement loops that outperform monolithic LLM generation.
Conclusions: Multi-agent LLM pipelines represent a robust foundation for precise, verifiable diagram generation and serve as a reproducible alternative to single-pass text-to-diagram models.

1. Introduction

Automated diagram generation has evolved from manual drawing to intelligent, text-driven visual synthesis. This progression reflects increasing abstraction from geometric representation to semantic reasoning. We categorize the evolution into four stages:

- Drawing → Diagram
- Template/Pattern → Diagram
- Formal Specification → Diagram
- Text → Diagram

These stages illustrate how the field transitioned from handcrafted representations to symbolic modeling and, more recently, to reasoning-centric LLM pipelines capable of self-verification and stylistic refinement.

1.1. From drawing to diagram

Early design began with hand drawings and sketches – the most fundamental means of visual reasoning. Brito et al. [1] highlight the role of manual drawing in the first 150 years of

© 2026 The Authors. Published by Wrocław University of Science and Technology Publishing House.
This is an open access article under the CC BY-SA 4.0 license international.

engineering, showing how sketching supported problem-solving and concept communication. Taraszkiewicz [2] contrasts freehand and digital design tools, underscoring that hand drawing still cultivates spatial thinking and design creativity. Willis et al. [3] bridge sketching and CAD by showing how engineering sketches encode geometric and semantic cues for later automation. Agrawal et al. [4] review hand-drawn diagram recognition, documenting early attempts to computationally interpret sketches. Finally, Donnici [5] integrates hand sketching, digital modeling, and generative AI, revealing how analog creativity and computational representation can co-exist. Autodesk [6] and Trimble [7] represent the commercial lineage of this digital-design evolution.

1.2. From template/pattern to diagram

The next step introduced reusable visual grammars and templates. Rule-based CAD macros and parametric patterns produced diagrams from parameter sets. Later, Generative Adversarial Networks (GANs) [8–11] automated the creation of structured layouts, learning spatial design grammars from data. In parallel, prompt-engineering research by White et al. [12] formulated prompt patterns (e.g., Critic, Refine, Plan) – textual analogs of design templates. These works represent the movement from handcrafted geometry to pattern-based diagram synthesis.

1.3. From formal specification to diagram

Formal and software-engineering research established methods to automatically generate diagrams from structured specifications. Buhr et al. [13] introduced Software CAD, converting software design specifications into statecharts and communication diagrams. Object-oriented modeling (Rumbaugh et al. [14]; Booch [15]; UML spec [16]) and graph-transformation systems (Schürr et al. [17]; France & Rumpe [18]) formalized the transformation from abstract models to diagrams. Domain-Specific Languages (DSLs) [19] and story diagrams [20] extended these principles into executable, visual specification languages.

1.4. From text to diagram

1.4.1. Pre-LLM foundations

Early approaches used syntactic NLP pipelines to generate diagrams from restricted text domains. Ghosh et al. [21] presented an ER-Generator that transformed English text into Entity–Relationship diagrams using handcrafted linguistic rules. Although functional, such methods were brittle and domain-limited, revealing the need for semantic reasoning.

1.4.2. LLM reasoning and prompting

Recent advances introduced structured prompting and multi-step reasoning. Rane et al. [22], Giray [23], Wei et al. [24], and Yao et al. [25] explored techniques such as Chain-of-Thought and Tree-of-Thought prompting that enable LLMs to produce semantically coherent structured outputs. Building upon these foundations, Zhang et al. [26] proposed an LLM agent that uses ReAct prompting and retrieval-augmented generation (RAG) to convert natural-language descriptions into AutoCAD-compatible structural drawings. Their system

is conceptually similar to ours – transforming text into diagrammatic representations – but lacks two elements central to our framework: (1) autonomous feedback loops for verification, and (2) stylistic cognition layers for applying contextual flavors and visual refinement.

2. Our architecture: Multi-agent feedback system for diagram generation

2.1. Motivation

While prior approaches to automated diagram generation typically operate as single-pass systems – producing diagrams directly from prompts or specifications – they lack mechanisms for self-correction, contextual adaptation, and aesthetic refinement. In contrast, our framework (Figure 1) adopts a multi-agent architecture inspired by distributed cognition and reflective reasoning. Each agent operates semi-autonomously within a shared state space, contributing a specialized capability (selection, generation, verification, refinement) to collectively ensure accuracy, coherence, and stylistic quality. This architecture embodies the principle of "reason–generate–validate–refine", forming a closed feedback loop that mimics expert-level design iteration.

2.2. Agent 3 – Meta-selector

Agent 3 functions as the meta-reasoning component responsible for determining the most appropriate diagram type for a given source input. It begins by retrieving a catalog of available diagram types, each defined by its structural semantics, representational intent,

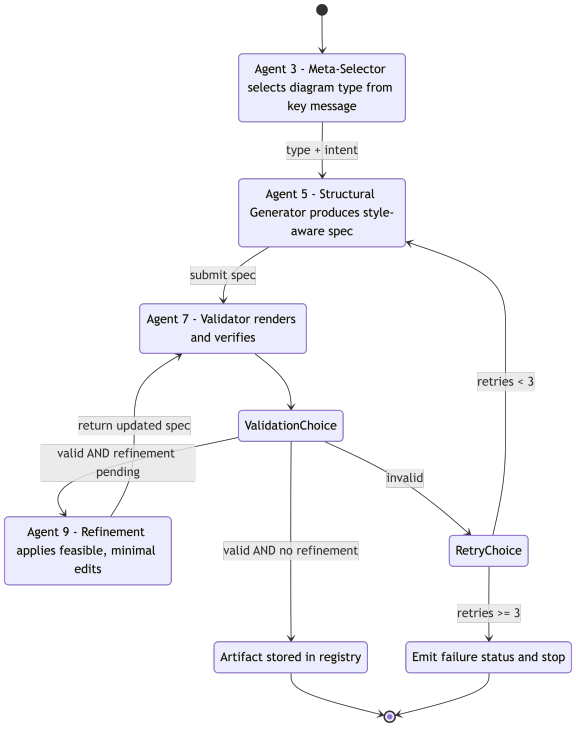


Figure 1. Agentic workflow

and domain of applicability (e.g., UML, ER, flowchart, or sequence diagrams). This catalog serves as the agent's design space and reference ontology. Subsequently, Agent 3 reads the source material – which may consist of program code, structured data, or descriptive text – from which a diagram is to be derived. It then invokes a large language model (LLM) to perform high-level semantic abstraction, transforming the raw source into a key message that captures its core functional or conceptual meaning. Using this key message, Agent 3 performs a mapping operation between the extracted semantic intent and the entries in its diagram catalog, again by requesting the LLM. The outcome of this reasoning process is the selection of the most contextually appropriate diagram type, which is then passed as structured metadata to downstream agents for generation and validation.

2.3. Agent 5 – Structural generator

Agent 5 acts as the structural synthesis component of the architecture, translating the conceptual representation provided by Agent 3 into a formally defined, executable diagram specification. It not only generates the structural content of the diagram but also integrates contextual style directives, referred to as flavors, which guide the visual and aesthetic realization of the final output.

2.3.1. Structural synthesis

Upon receiving the diagram-type metadata and semantic intent from Agent 3, Agent 5 loads the corresponding syntax–semantics specification for that diagram category. This specification defines the grammar, permissible constructs, and visual conventions required to produce a valid diagram instance (e.g., entity relationships in ER diagrams, message lifelines in sequence diagrams, or cardinalities in UML class models). Using this formal schema, Agent 5 constructs a structured generation prompt and invokes a diagram-generation model (typically an LLM fine-tuned for code synthesis). The model produces a well-formed diagram specification – such as PlantUML, Mermaid, or Graphviz code – encoded according to the chosen visual language.

2.3.2. Integration of flavors

In addition to structural information, Agent 5 receives flavor directives that guide the diagram's visual appearance and stylistic composition. Flavors represent high-level customization instructions that can be expressed either as free-form textual guidelines (e.g., "use rounded corners for nodes," "render in black and white," "omit datatypes from attributes"), or example diagrams that implicitly define the desired appearance through color schemes, font choices, and layout properties. Two categories of flavors are distinguished:

- **Global flavors** – define uniform design conventions across all generated diagrams. They establish the overall aesthetic identity of the system, enforcing rules such as monospaced fonts, consistent color palettes, uniform spacing, or minimalistic edge styles.
- **Local flavors** – apply to individual diagram instances and specify fine-grained refinements or stylistic deviations. These may include color highlighting for selected node types, UML layout constraints, or simply providing an example diagram whose color, font, and style parameters are extracted and reused for the current generation task.

During synthesis, Agent 5 merges the semantic structure (from Agent 3) with the flavor constraints to produce a style-aware structural output. This ensures that every generated

diagram is not only syntactically correct and semantically coherent but also visually aligned with the desired aesthetic conventions. The resulting diagram specification is serialized and transmitted to Agent 7 for validation and possible regeneration.

2.4. Agent 7 – Deterministic validator and artifact constructor

Agent 7 functions as the deterministic validation and artifact-construction layer within the multi-agent feedback loop. Unlike the reasoning-driven agents that precede it, Agent 7 performs purely mechanical verification by executing the appropriate rendering engine for the selected diagram framework. Its primary objective is to ensure that the structural specification produced by Agent 5 is syntactically valid and formally renderable within its designated visualization environment. Upon receiving the diagram specification and diagram-type metadata, Agent 7 invokes the corresponding rendering engine – for example Mermaid.js for Mermaid diagrams, PlantUML.jar (with Graphviz) for UML or architecture diagrams, or specialized back-ends for domain-specific cases. The rendering engine performs a deterministic compilation of the diagram code and produces diagnostic feedback in the form of logs or console messages. If the rendering process fails, Agent 7 encapsulates the extracted diagnostics into a structured feedback message and redirects it to Agent 5, requesting regeneration or localized correction. This establishes a closed validation loop that enables the system to autonomously refine the generated code until a formally valid output is produced. To maintain operational determinism and prevent infinite recursion, Agent 7 enforces a retry limit of three iterations. If no valid rendering is achieved after three validation cycles, the agent terminates the process and emits a failure status report to the system controller. When rendering succeeds, Agent 7 stores the resulting validated artifact – typically an SVG, PNG, or JSON-encoded graph representation – into the system's artifact registry. Thus, Agent 7 represents the mechanical boundary between generative reasoning and visual materialization, ensuring that only formally consistent outputs enter the refinement stage.

2.5. Agent 9 – Refinement

Agent 9 implements the optional refine-and-retry phase of the workflow. Whereas Agent 7 focuses on validating a given diagram against the chosen engine (Mermaid, PlantUML, or Markdown) and producing the corresponding artifacts, Agent 9 is invoked when refinement instructions are present and a previous validation run suggests that a targeted revision might improve the result. In the control graph, execution flows from Agent 7 to Agent 9 whenever the shared state marks a refinement as pending; after Agent 9 has finished, the updated diagram is passed back to Agent 7 for a final validation pass. Beyond cosmetic edits, Agent 9 is designed for domain-specific, user-driven semantic adjustments (e.g., "remove risk node from mindmap") that tailor an already valid diagram to its intended purpose.

Internally, Agent 9 orchestrates two LLM-based sub-agents: a feasibility checker and the actual refinement agent. The feasibility step inspects the user's refinement instructions together with the diagram format, the logical diagram type, an excerpt of the current diagram code, and any available specification text. Its task is to decide whether the requested changes can be realized without violating the underlying diagram language or the documented constraints. If the instructions are deemed infeasible (for example because they conflict with the grammar or with mandatory attributes), Agent 9 aborts the refinement

phase, records a concise explanation and recommendation in the state, and hands control back to the workflow without modifying the diagram.

If the instructions are feasible, Agent 9 invokes the refinement agent, which applies the requested changes to the diagram code in a series of bounded attempts. Each attempt is guided by three layers of context: the explicit instructions, diagram-specific flavor guidance, and general flavor guidance, as well as any validator feedback from earlier runs. The refinement agent aims for minimal, surgical edits that satisfy the instructions while preserving syntactic validity; after each candidate update it re-runs the appropriate validator (Mermaid CLI, PlantUML, or the Markdown pipeline). On success, Agent 9 stores the refined diagram and a short refinement summary in the shared state, marks the refinement as completed, and returns control to Agent 7, which then re-validates the improved diagram and produces the final artifacts.

3. Graphical user interface

To complement the command-line workflow, a lightweight, Tkinter-based graphical front-end provides an interactive interface to the same multi-agent pipeline. The main window mirrors the agent architecture: the first row presents free-form source text on the left and a read-only preview of the generated diagram code on the right; the second row exposes general and diagram-specific flavor guidance; and the third row collects refinement instructions that are forwarded into the refinement phase. Below these inputs, controls allow users to request a diagram-type suggestion (invoking the selection capability) or to run the full workflow including validation and optional refinement.

The interface does not re-implement agent logic. Instead, it delegates execution to the existing command-line entry point by launching background tasks with the appropriate arguments. At run time, the GUI prepares the current source, flavor guidance, and (if present) refinement instructions as temporary artifacts, then invokes the pipeline with the corresponding parameters for source, specification location, flavor configuration, engine, language, and refinement mode. While the pipeline executes, the GUI monitors process output and shared logs, updates a live log pane, and displays an "agent chain" status indicator showing which components are currently active. When execution completes, the interface locates the generated diagram artifact (e.g., Mermaid, PlantUML, or Markdown), displays its raw code in the preview pane, and offers quick actions such as opening the rendered SVG, re-running validation, or triggering a focused refinement pass.

Key advantages:

- **Single source of truth** – All workflow logic (graph wiring, agents, validation, refinement, configuration) resides in the core pipeline; the GUI simply invokes that entry point, avoiding divergent implementations.
- **Consistent behavior** – Graphical and command-line runs exercise the same pipeline with the same parameters and defaults. Any change to the workflow automatically applies to both interfaces.
- **Simpler maintenance** – Fixes and features are implemented once in the shared pipeline. The GUI code focuses on layout, event handling, and process orchestration, not diagram logic.
- **Isolation of concerns** – The UI layer remains thin and does not depend on internal graph state or agent APIs; it prepares inputs, invokes the pipeline, and renders outputs and logs.

– **Easier testing and debugging** – The pipeline can be tested headlessly via the CLI (e.g., in CI) while the GUI acts as a thin shell. Logs and behavior remain consistent regardless of how the workflow is triggered.

Overall, the results demonstrate that the **multi-agent LLM pipeline is robust across a heterogeneous set of LLM back ends**, and that deterministic validation via Agent 7 effectively homogenizes quality: once wrapped in the same generate–validate–refine loop, even diverse models yield comparably reliable diagram artifacts, differing mainly in speed and first-attempt stability rather than in their ultimate ability to satisfy the diagram specification.

4. LLM Performance Evaluation for Text-to-Diagram Generation

We evaluated ten large language models (LLMs) on 30 text inputs: 20 short **easy** prompts (`easy001`–`easy020`) and 10 structurally more demanding **`complex`** specifications. For each model–prompt pair, the multi-agent pipeline executed Agent 5 (diagram generator) followed by Agent 7 (deterministic validator). Agent 7 attempted to render the generated diagram code up to three times and recorded diagnostics such as syntax violations, missing nodes/edges, and invalid attributes.

Each run is logged in `agent7_stats.jsonl` with: – the LLM identifier (`model`), – the proposed diagram type (`diagram_type_proposed`), – the number of validation iterations (`validation_iterations`), – whether the diagram passed on the first attempt (`passed_first_attempt`), – the final validation status (`final_validation_status`), – a structured error profile (`error_profile`), and – end-to-end latency (`latency_ms`). In total, the dataset contains **300 runs** (30 prompts times 10 models).

4.1. Overall model comparison

Table 1 summarizes validation performance and latency for all models across all prompts.

– **Final success**: fraction of runs where Agent 7 reported `final_validation_status = OK` within at most three iterations.
– **First-attempt**: fraction of runs that already passed in the very first validation call (no regeneration from Agent 5 required).
– **Mean iters**: mean number of Agent 7 validation iterations per run.

Table 1. Validation performance and latency for all models across all prompts.

Model	Runs	Final success	First-attempt	Mean iters	Mean latency (s)
anthropic/claude-sonnet-4.5	30	30	28	1.10	44.52
deepseek/deepseek-v3.2-exp	30	28	24	1.17	105.06
glm-4.6	30	25	19	1.13	100.87
google/gemini-2.5-pro	30	29	24	1.23	91.06
google/gemini-3-pro-preview	30	30	29	1.03	84.62
moonshotai/kimi-k2-thinking	30	25	21	1.03	613.12
openai/gpt-5	30	29	24	1.30	158.99
openai/gpt-5-codex	30	28	25	1.23	92.93
openai/gpt-5.1-codex	30	28	28	1.07	49.69
x-ai/grok-4-fast	30	28	23	1.27	29.48
x-ai/grok-code-fast-1	30	26	23	1.37	34.07

– **Mean latency**: average end-to-end time per run (Agent 5 generation + Agent 7 validation).

Overall, all models achieve high final success rates ($\geq 85\%$), but they differ in how reliably they succeed on the **first attempt** and in their **latency**. For example, some models trade slightly lower first-attempt accuracy for shorter runtimes, while others are slower but almost always produce validator-clean code on the first try.

4.1.1. Success rates

The bar chart in Figure 2 shows the final validation success rate for each model. All LLMs eventually produce valid diagrams for the majority of prompts, but models such as `openai/gpt-5-codex`, `anthropic/claude-sonnet-4.5`, and `openai/gpt-5.1-codex` reach **near-perfect final success**.

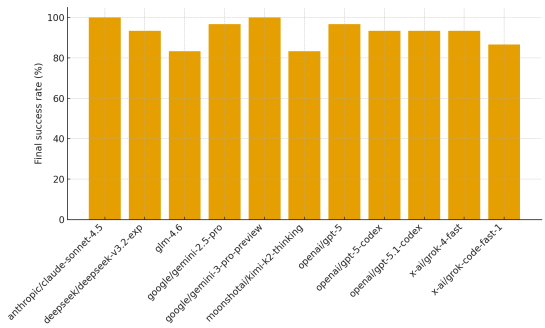


Figure 2. Final validation success rate per model

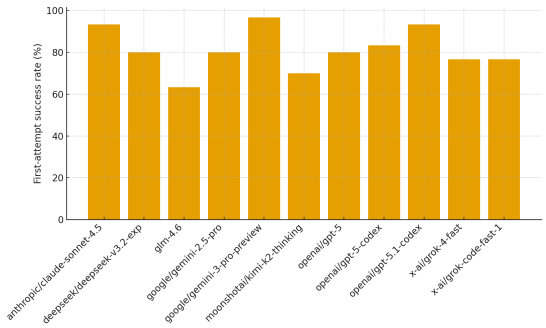


Figure 3. First-attempt validation success per model

Figure 3 focuses on the stricter metric of **first-attempt success**, i.e., whether thevery first diagram produced by Agent 5 passed validation. This metric is particularly relevant for interactive usage, where regeneration loops are undesirable.

4.1.2. Validation iterations

The mean number of validation iterations (Figure 4) is close to 1 for all models, indicating that the pipeline rarely needs to use the full budget of three attempts. Nevertheless, some models systematically require more retries, which suggests that they occasionally produce code with minor syntax or style issues that Agent 7 can catch and correct via regeneration.

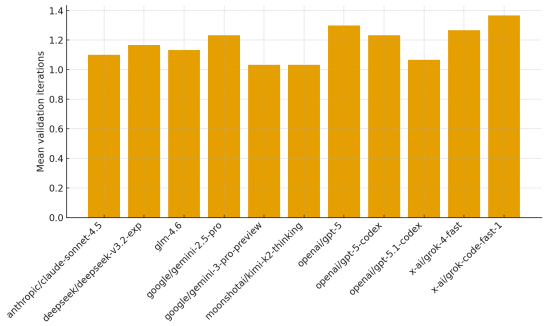


Figure 4. Mean Agent 7 validation iterations per model

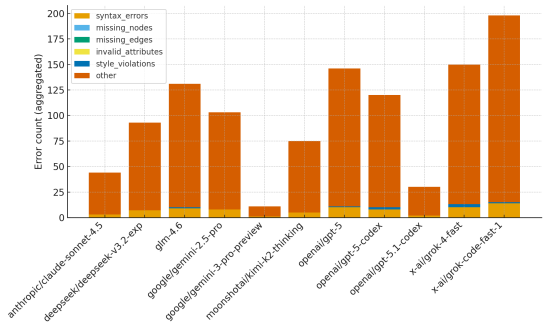


Figure 5. Error profile per model (Agent 7 diagnostics)

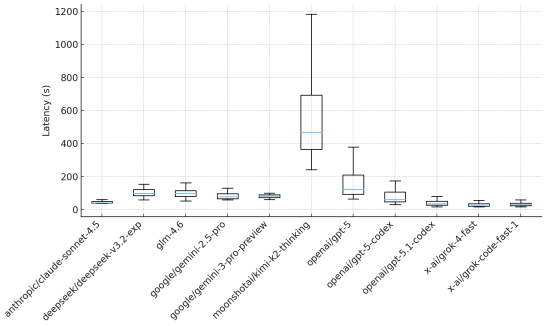


Figure 6. Latency distribution per model

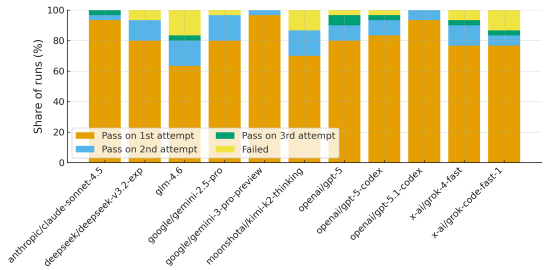


Figure 7. Cumulative success rates by validation iteration (Agent 7)

4.1.3. Latency

Figure 6 reports the latency distribution per model as a boxplot (median, quartiles, and central spread; outliers omitted for readability). We observe thatlatency varies significantly across providers. Some models provide **fast but slightly less stable** first-attempt behavior, while others are **slower but more consistent**. This highlights a practical trade-off when embedding LLMs into an interactive diagrammingtool.

4.2. Error profiles from Agent 7 diagnostics

Agent 7's structured `error_profile` field allows us to distinguish between different failure modes:

- `syntax_errors`: the rendering engine rejected the diagram due to malformed syntax.
- `missing_nodes`/`missing_edges`: references to undefined entities or incomplete relationships.
- `invalid_attributes`: invalid or ill-typed attributes for the given diagram language.
- `style_violations`: violations of style or flavor constraints (e.g., forbidden attribute formats).

– `other`: remaining engine errors and miscellaneous diagnostics.

Figure 5 aggregates these errors across all prompts for each model. The stacked bars show that most non-successful runs are dominated by **syntax errors** and a large residual `"other"` category. The absence of `missing_nodes`, `missing_edges`, and `invalid_attributes` in most models suggests that once the syntax is correct, the structural semantics of the diagrams are usually consistent with the specification.

The relatively small number of `style_violations` indicates that integrating **global** and **local flavors** into the prompt (Section 2.3 of the paper) is generally robust, and that LLMs rarely break purely aesthetic constraints once they have been learned.

4.3. Prompt complexity: easy vs. complex

To assess sensitivity to input complexity, we distinguish between the 20 short **easy** prompts (`easy001`–`easy020`) and the 10 structurally richer **`complex`** prompts. Table 2 reports success metrics for both groups.

Table 2. Success rates for easy prompts vs. complex prompts

Dataset	Model	Runs	Final success	First-attempt
easy	`anthropic/claude-sonnet-4.5`	20	100.0%	100.0%
easy	`deepseek/deepseek-v3.2-exp`	20	100.0%	85.0%
easy	`glm-4.6`	20	90.0%	65.0%
easy	`google/gemini-2.5-pro`	20	95.0%	85.0%
easy	`moonshotai/kimi-k2-thinking`	20	90.0%	80.0%
easy	`openai/gpt-5`	20	95.0%	85.0%
easy	`openai/gpt-5-codex`	20	100.0%	90.0%
easy	`openai/gpt-5.1-codex`	20	100.0%	95.0%
easy	`x-ai/grok-4-fast`	20	100.0%	85.0%
easy	`x-ai/grok-code-fast-1`	20	95.0%	85.0%
complex	`anthropic/claude-sonnet-4.5`	10	100.0%	100.0%
complex	`deepseek/deepseek-v3.2-exp`	10	100.0%	100.0%
complex	`glm-4.6`	10	0.0%	0.0%
complex	`google/gemini-2.5-pro`	10	100.0%	100.0%
complex	`moonshotai/kimi-k2-thinking`	10	100.0%	0.0%
complex	`openai/gpt-5`	10	100.0%	100.0%
complex	`openai/gpt-5-codex`	10	100.0%	0.0%
complex	`openai/gpt-5.1-codex`	10	100.0%	100.0%
complex	`x-ai/grok-4-fast`	10	0.0%	0.0%
complex	`x-ai/grok-code-fast-1`	10	100.0%	0.0%

On the **easy** prompts, most models achieve **final success rates of 95–100%** and high first-attempt rates, indicating that generating syntactically valid flowchart-style diagrams is a largely solved problem for modern LLMs in this pipeline. The more demanding **`complex`** prompts are represented by ten runs per model, so the numbers should still be interpreted with appropriate caution; however, they serve as an initial indicator that complex specifications can trigger additional failures or retries in some models, especially those optimized for speed.

4.4. Interpretation and implications for the architecture

From the perspective of the multi-agent architecture, the experiment supports three main observations:

1. **Agent 7 as a robustness amplifier.**
   Even for models with imperfect first-attempt behavior, the combination of deterministic validation and bounded regeneration significantly increases the final success rate. This confirms the effectiveness of the "generate–validate–regenerate" loop as a guardrail around high-variance generative models.
2. **Models differ more in stability than in raw capability.**
   All tested models are capable of producing valid diagram code for the majority of prompts. The main differences lie in how often they produce such code **without** needing retries, and how quickly they do so. For interactive settings, high first-attempt accuracy and low latency are more important than marginal differences in ultimate success rate.
3. **Error structure matters for prompt and flavor design.**
   The dominance of syntax-related errors and the scarcity of structural errors (missing nodes/edges) suggest that syntax templates and explicit grammar reminders in the prompts are crucial. Flavor-related violations are rare, indicating that stylistic guidance integrates well with the structural generation step and does not destabilize the pipeline.

Overall, the results demonstrate that the **multi-agent LLM pipeline is robust across a heterogeneous set of LLM back ends**, and that deterministic validation via Agent 7 effectively homogenizes quality: once wrapped in the same generate–validate–refine loop, even diverse models yield comparably reliable diagram artifacts, differing mainly in speed and first-attempt stability rather than in their ultimate ability to satisfy the diagram specification.

5. Summary

We surveyed the evolution of diagram generation from manual sketches and templates to formal model–driven pipelines and, most recently, LLM-based text-to-diagram systems. Building on these foundations, we introduced a modular, multi-agent architecture that operationalizes a reason–generate–validate–refine loop: Agent 3 selects the appropriate diagram type from a catalog; Agent 5 synthesizes a structured, executable specification while integrating global and local flavor guidance; Agent 7 deterministically validates and materializes artifacts; and Agent 9 performs optional, user-guided refinements with feasibility checks and minimal edits.

We implemented this pipeline end-to-end and provided a thin Tkinter GUI that delegates execution to the same CLI entry point, ensuring consistent behavior and simplified maintenance. In an empirical study across 30 diagrams (20 easy, 10 complex) covering 300 runs (30 prompts $\times$ 10 models), most models achieved high final validation success, while differing primarily in first-attempt stability and latency. Agent 7 served as a robustness amplifier by catching syntax issues and enabling bounded regeneration, and flavor constraints integrated cleanly without destabilizing structural correctness.

Future work will extend the diagram catalog and grammars, strengthen semantic grounding through DSLs and graph-transformation back ends, and deepen human-in-the-loop refinement (provenance, safety, and usability studies). We also plan to explore multimodal inputs and visual QA, broaden benchmarks for complex specifications beyond ER diagrams, and release artifacts (prompts, logs, and metrics) to facilitate reproducibility and community adoption.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data and code availability

We have shared our data in a archive available at https://zenodo.org/records/17741490.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sector

Declaration of AI assistance in the writing process

During the preparation of this manuscript, the authors used AI-based tools, including ChatGPT and Google Gemini, to support grammar checking, language refinement, and clarity of expression. After using these tools, the authors carefully reviewed and edited the text as necessary and accept full responsibility for the final content of the publication.

References

[1] E. Brito, R. Póvoas, and P. Providência, "Drawing in the engineering design process," in *Structures and Architecture*. CRC Press, 2016, pp. 871–878.
[2] A. Taraszkiewicz, "Freehand drawing versus digital design tools in architectural education," *Global Journal of Engineering Education*, Vol. 23, No. 2, 2021, pp. 107–112.
[3] K.D.D. Willis, Y. Li, Y. Sun, and T. Funkhouser, "Engineering sketch generation for computer-aided design," 2021, arXiv:2104.09621.
[4] R. Agrawal, K. Jagtap, and R. Kantipudi, "An overview of hand-drawn diagram recognition methods and applications," 2023, researchGate Preprint.
[5] L. Donnici, "Rethinking sketching: Integrating hand drawings, digital tools, and generative AI," *Infrastructures*, Vol. 9, No. 5, 2025, p. 119.
[6] Autodesk Inc., "AutoCAD software," 2024. [Online]. https://www.autodesk.com/products/autocad/overview
[7] Trimble Inc., "SketchUp software," 2024. [Online]. https://www.sketchup.com
[8] F. Luleci, F.N. Catbas, and O. Avci, "Generative adversarial networks for acceleration data augmentation in structural monitoring," *Journal of Civil Structural Health Monitoring*, Vol. 13, No. 1, 2023, pp. 181–198.
[9] G.C. Marano, M.M. Rosso, A. Aloisio, and G. Cirrincione, "Review of GANs in earthquake engineering," *Bulletin of Earthquake Engineering*, Vol. 22, No. 7, 2024, pp. 3511–3562.
[10] Z. Rastin, G.G. Amiri, and E. Darvishan, "GAN for damage identification in civil structures," *Shock and Vibration*, 2021, p. 3987835.
[11] Y. Feng, Y. Fei, Y. Lin, W. Liao, and X. Lu, "Intelligent generative design using rule-embedded GAN," *Journal of Structural Engineering*, Vol. 149, No. 11, 2023, p. 04023161.
[12] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea et al., "A prompt pattern catalog to enhance prompt engineering," 2023, arXiv:2302.11382.

[13] R.J.A. Buhr, G.M. Karam, C.J. Hayes, and C.M. Woodside, "Software CAD: A revolutionary approach," *IEEE Transactions on Software Engineering*, Vol. 15, No. 3, 1989, pp. 235–249.
[14] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, *Object-Oriented Modeling and Design*. Prentice Hall, 1991.
[15] G. Booch, *Object-Oriented Analysis and Design with Applications*, 2nd ed. Benjamin/Cummings, 1994.
[16] Object Management Group, "Unified modeling language (UML) specification, version 2.5.1," 2017. [Online]. https://www.omg.org/spec/UML/2.5.1
[17] A. Schürr, A. Winter, and A. Zündorf, "The PROGRES approach: Language and environment," in *Handbook of Graph Grammars and Computing by Graph Transformation*. World Scientific, 1999, pp. 487–550.
[18] R. France and B. Rumpe, "Model-driven development: A research roadmap," in *Future of Software Engineering (FOSE 2007)*, 2007, pp. 37–54.
[19] M. Mernik, J. Heering, and A.M. Sloane, "When and how to develop domain-specific languages," *ACM Computing Surveys*, Vol. 37, No. 4, 2005, pp. 316–344.
[20] T. Fischer, J. Niere, L. Torunski, and A. Zündorf, "Story diagrams: A new graph transformation language based on UML and Java," in *Theory and Application of Graph Transformation*. Springer, 2000, pp. 296–309.
[21] S. Ghosh, P. Mukherjee, B. Chakraborty, and R. Bashar, "Automated generation of E–R diagram from a given text," in *International Conference on Machine Learning and Data Engineering (iCMLDE)*, 2018, pp. 91–96.
[22] N. Rane, S. Choudhary, and J. Rane, "Integrating BIM with generative AI," *Buildings*, Vol. 14, No. 1, 2023, p. 220.
[23] L. Giray, "Prompt engineering with chatgpt: A guide for academic writers," *Annals of Biomedical Engineering*, Vol. 51, No. 12, 2023, pp. 2629–2633.
[24] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia et al., "Chain-of-thought prompting elicits reasoning in large language models," in *Advances in Neural Information Processing Systems*, Vol. 35, 2022, pp. 24 824–24 837.
[25] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths et al., "Tree-of-thoughts: Deliberate problem solving with large language models," in *Advances in Neural Information Processing Systems*, Vol. 36, 2023, pp. 11 809–11 822.
[26] X. Zhang, L. Iturburu, J.N. Villamizar, X. Liu, M. Salmeron et al., "LLM agent for structural drawing generation using ReAct prompting and RAG," 2025, arXiv:2507.19771.

Appendix A.

This section summarizes the outputs produced from a set of seven input applications for an IT Expert (complex007.txt). The figures illustrate qualitative differences across the evaluated language models.

Model glm-4.6 by z.ai generated a structured mind map highlighting key skill clusters (Figure A1). Model openai/gpt-5-codex by openai produced a tabular representation of the same information, emphasizing categorical organization (Figure A2). Model gemini-2.5-pro by google, in contrast, created a radar diagram capturing relative skill intensities (Figure A3). These differences demonstrate the variety of diagrammatic representations obtainable from identical textual input.



Figure A1. Mind map generated by model glm-4.6 for complex007 input

Candidate	Primary Focus	Signature Skills & Tools	Team Contributions
Alex Carter	Cloud infrastructure & reliability engineering	AWS, Azure, Terraform, Ansible, GitLab/GitHub CI, Prometheus, Grafana	Produces clear documentation, mentors engineers, embeds security and incident prevention
Priya Shah	Backend services & platform operations	Python, Java, Kubernetes, PostgreSQL, Redis, Kafka, observability tooling	Shapes APIs and SLAs with stakeholders, advances code reviews, streamlines workflows
Daniel Novak	Network architecture & IT security	Campus/DC networking, firewalls, VPNs, zero-trust design, SSO integration	Leads proactive monitoring, supports audits, enforces change management
Laura Meier	Enterprise integration & process automation	ERP/CRM connectors, REST APIs, message queues, data modeling	Bridges business requirements to designs, iterates delivery, establishes governance
Omar El-Sayed	DevOps & platform engineering	Docker, Kubernetes, Helm, Argo CD, IaC, blue-green/canary deployments	Builds self-service platforms, promotes blameless culture, optimizes cost and performance
Hannah Fischer	Data platforms & analytics infrastructure	Apache Airflow, Spark, cloud data warehouses, Python, SQL	Ensures data quality and lineage, secures access, operationalizes ML workloads
Miguel Alvarez	Endpoint management & collaboration services	Microsoft 365, Google Workspace, device management automation	Delivers user-centric support, documents processes, coordinates lifecycle management
Emily Zhang	Software architecture & quality engineering	Domain-driven design, event-driven systems, secure auth, automated testing	Guides design reviews, mentors on clean code, aligns agility with business goals

Figure A2. Table representation generated by model openai/gpt-5-codex for complex007 input

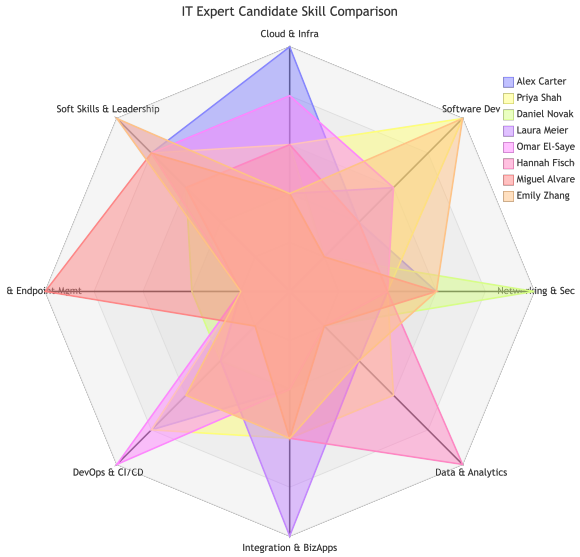


Figure A3. Radar generated by model gemini-2.5-pro for complex007 input

Authors and affiliations

Hartwig Grabowski
e-mail: hartwig.grabowski@hs-offenburg.de
ORCID: https://orcid.org/0009-0001-4300-2626
Institute of Machine Learning and Analysis (IMLA),
Offenburg University of Applied Sciences, Offenburg,
Germany

e-Informatica Software Engineering Journal
Volume 20, Issue 1, pages: 1–43
DOI: 10.37190/e-Inf260110
BibTeX

Onboarding Newcomers in Large-Scale Agile Projects: A Multiple Case Study

Pavithra Herath*, Muhammad Ovais Ahmad, Tomas Gustavsson
*Corresponding author: pavithra.herath@kau.se

Article info

Keywords:
onboarding
agile
large-scale agile
social debt
process debt
people debt
knowledge debt
organizational debt

Submitted: 31 Mar. 2026
Revised: 12 Jun. 2026
Accepted: 28 Jun. 2026
Available online: 2 Jul. 2026

Abstract

Context: Large-scale agile (LSA) is inherently characterized by socio-technical complexities (e.g., system dependencies, extensive cross-team coordination, and distributed organizational structures). Onboarding of newcomers is a critical challenge in LSA environments and it remains underexplored compared to small-scale agile settings.
Objective: Our study aims to investigate the onboarding processes and challenges within LSA projects.
Method: In this exploratory qualitative study, we conducted 41 semi-structured interviews across four Swedish software companies.
Results: We identified onboarding issues that result in negative outcomes, such as cognitive overload, psychological safety constraints, and cross-team collaboration hurdles. These factors create steep learning curves and knowledge deficiencies that contribute to long-term socio-technical debt.
Conclusions: We provide actionable recommendations for practitioners to improve their onboarding processes. In particular, we recommend designing competency-based rather than time-based onboarding plans, expanding social integration through structured cross-team exposure, and addressing the gap between textbook agile and real-world LSA practices through explicit expectation setting.

1. Introduction

Agile methodologies were originally designed for small, co-located teams, but their growing significance in practice led to their application in large-scale organizations which requires significant adaptations [1–5]. Software companies are increasingly adopting various Large-Scale Agile (LSA) frameworks (e.g. Scaled Agile Framework (SAFe) and Large-Scale Scrum (LeSS)) to manage projects involving dozens of people across multiple, interconnected teams [1]. In LSA, onboarding is a challenging task because new hires must integrate technically and socially into a network of teams, shared resources, and all required company meetings.

Successful onboarding establishes better relationships, increases satisfaction, clarifies expectations, and provides support to reduce unwanted turnover [6]. A few studies explore onboarding in agile teams [7,8], but are lacking in the LSA context. LSA projects experience unique challenges related to task complexity, skills alignment, and cross-team coordination. However, successful onboarding demands learning, confidence building, early mentoring, early task involvement, training and socilalization [7, 9, 10]. This highlights the need for

© 2026 The Authors. Published by Wrocław University of Science and Technology Publishing House.
This is an open access article under the CC BY-SA 4.0 license international.

a flexible and effective onboarding process in LSA projects. To bridge this research gap, our study aims to explore how LSA organizations manage the integration of new members. Our study aims to answer the following research questions:
RQ1: How do LSA projects onboard newcomers?
RQ2: How do onboarding challenges contribute to socio-technical debt in LSA projects?

This paper extends our earlier work [11]. We use Bauer's onboarding model [6] as a theoretical lens to map case-specific practices across two primary dimensions: (A) Onboarding Functions (including recruitment, orientation, support tools, coaching, training, and feedback mechanisms), (B) New Employee Adjustments (focusing on self-efficacy, role clarity, social integration, and cultural knowledge). We also identify specific socio-technical challenges connected to onboarding within LSA context. In this extended version, we provide a more in-depth analysis and comprehensive insights into the topic, and present additional empirical results focusing on onboarding functions and adjustments. We include "technical integration'' as an adjustment dimension in LSA context, extending beyond Bauer's onboarding model. Furthermore, we identify onboarding challenges, present a categorization of these challenges: organizational, individual and contextual and discuss how these challenges influence socio-technical debt in LSA projects.

The remainder of paper is organized as follows: Section 2 provides background to LSA and onboarding. Section 3 describes our research methodology, including research context, data collection and analysis procedures. Section 4 presents the results, followed by a discussion in Section 5. Section 6 offers recommendations, and Section 7 concludes the paper.

2. Background

In this section, we provide an overview of LSA and an introduction to onboarding. We present a detailed description of Bauer's model, explaining the onboarding functions and new employee adjustments.

2.1. Large-scale agile

Agile methodologies were originally designed for small co-located teams [1], but their success has led to their adoption in LSA contexts [1,2,12]. LSA projects are highly complex, and this complexity tends to increase with the project size [13]. Larger scope and high level of complexity introduce significant management and project integration challenges. LSA projects touch on more parts of the organization, posing a risk when something goes wrong. Missing business focus, technical complexity, unaligned teams, and lack of skills stand out among the causes of most project failures [1,14].

LSA comprises multi-team collaboration, typically involving 2–9 teams and 50 or more personnel who handle complex interdependencies [2]. However, LSA frameworks have "struggled to deal with the exponentially greater complexities and interdependencies of large-scale, organization-wide development" [1]. These complex inter-team interactions and organizational structures produce challenges that traditional methods struggle to address [1,2]. Furthermore, focusing excessively on adapting framework elements (such as SAFe roles and artifacts) often distracts the organization from its core business needs. Scaling agile to non-development units presents a persistent challenge across both the SAFe and LeSS frameworks. LSA organizations must cope with concurrent changes to team

structures, processes and tools [1]. At this level, standard knowledge-sharing approaches (e.g., co-located retrospectives) tend to break down at scale [2]. Consequently, LSA involves contingencies on how and when specific practices are applicable, which change with the project size, business domain, and team configurations [2].

Scaling out beyond traditional agile development requires extensive social and technical integration. The degree of complexity in inter-team interactions, development processes and organizational structures makes LSA projects highly susceptible to socio-technical debt. While technical debt (TD) occurs due to technical compromises in development that bring short-term benefits but create long-term system instabilities [15], non-technical debt (NTD) encompasses people, social, process and organizational debt [15–17] which arises from complex social interactions that subsequently accumulate TD. One specific source of NTD is "newbie free-riding" [15, 18]. When newcomers do not receive adequate support, guidance, or information, they may struggle to contribute effectively, placing an additional burden on team members and generating NTD in the form of knowledge gaps, social issues, and organizational inefficiencies.

2.2. Onboarding

"*Onboarding refers to the process that helps new employees learn the knowledge, skills, and behaviours they need to succeed in their new organizations*" [19]. Onboarding is also known as organizational socialization [6]. Onboarding helps "new hires adjust to social and performance aspects of their new jobs quickly and smoothly" [6, 19]. Several onboarding models are discussed in the literature, including Maanen and Schein model [20], Jones' model [21] and Bauer's model [6].

Maanen and Schein [20] define socialization as the "process by which an individual acquires the social knowledge and skills necessary to assume an organizational role" within their organizations. This ranges from self-guided trial and error to formal education and apprenticeship [20]. Maanen and Schein offered a theoretical explanation regarding role orientation in onboarding, showing six dimensions: (1) Collective vs. individual, (2) Formal vs. informal, (3) Sequential vs. random, (4) Fixed vs. variable, (5) Serial vs. disjunctive, and (6) Investiture vs. divestiture. Among these, three of which are particularly relevant to LSA:

- Formal vs. Informal: Formal onboarding isolates newcomers within structured programs (e.g., internships and apprenticeships), whereas informal onboarding has no clearly defined process to guide newcomers, and efforts are not made to distinguish them from experienced team members [7, 20].
- Collective vs. Individual: Collective onboarding occurs when a group of newcomers is recruited and faces a common set of experiences together. Individual onboarding occurs when newcomers face experiences separated from a group of newcomers [7, 20]. Each newcomer's onboarding journey is different from others, shaped by a "more or less unique set of experiences". Typical examples of individual onboarding include Apprenticeship programs, intern or trainee assignments and on-the-job training, where newcomers are expected to learn the knowledge and skills on their own accord.
- Fixed vs. Variable: Fixed onboarding occurs when there is a fixed timetable to complete set of activities. The newcomers are given precise knowledge of the time on how long the onboarding or any transition process will take. Various career paths within the same organization may have different time tables. In contrast, variable onboarding does not include any fixed timetable upon completion [7, 20].

Building on this, Jones model [21] categorizes onboarding as either institutionalized (structured orientation and mentoring) or individualized (learning norms and expectations "on the fly") [22]. Empirical studies have further characterized these experiences. Ju et al. [9] conducted a case study at Microsoft to understand and characterize the experiences and expectations of developers during onboarding in relation to the tasks they performed. Their findings identified three core themes in developer onboarding: learning, confidence building, and socialization [9]. Regarding technical integration, Yates et al. [23] conducted a study on programme comprehension during onboarding, based on in-situ onboarding sessions across eight organizations. They focused on the type of information that experts share with newcomers during onboarding and how valuable newcomers find that information. Their findings reveal that experts use four different views to explain the code: abstraction of the structure, algorithmic, temporal and rationale-based views. Among these, the rational-based view, which includes both function-based rationale and design-decision rationales, was particularly valued by newcomers [23]. Similarly, Fagerholm et al. [24] study onboarding in open source software projects on the role of mentors and project characteristics. Their results show that developers who receive deliberate onboarding support through mentoring become active at an earlier stage than developers who enter the project through conventional methods. In addition, project size and life time are significant factors that influence onboarding [24].

In the context of distributed agile teams Moe et al. [22] found that effectiveness of onboarding is affected by domain complexity, team type, and the availability of existing members. Additionally, it is important to note that "onboarding outcomes cannot always be predicted by management" [22]. Peggy et al. [7] extended Bauer's model for agile environments. Their findings indicate that while mentorship practices are universally helpful, both adapted general practices and unique agile techniques like pair programming are crucial for successfully integrating newcomers in teams [7]. To operationalize these concepts, Buchan et al. [8] identified eleven key goals for agile onboarding: (1) Understanding company culture, (2) Understanding team norms, (3) Knowing the responsibilities, expertise and authority of other team members, (4) Understanding what others expect from the role and meeting your responsibilities, (5) Understanding what tasks are assigned and when to do them, (6) Understanding the project structure, aims and implications, (7) Understanding how to code and test to meet team expectations, (8) Understand the teams' standards and quality, (9) Understand and gain the agile mindset, (10) Know how to use agile artifacts and techniques uses by team in software development process, (11) Understand the project domain knowledge and terminologies. However, achieving these goals in large-scale settings is challenging. Britto [10] investigated onboarding of software developers in LSA projects from Ericsson perspective. Their results show that onboarding performance was hampered by distant mentors, mismatched training, large early tasks, and team instability. Onboarding in such projects are requiring extensive advance planning and long-term mentoring by experienced members, which is very expensive [10].

Steinmacher et al. [25]found that newcomers in open source software projects encounter numerous barriers that hinder their initial contribution and lead to dropping out. They identified 58 barriers, which were grouped into six categories: cultural differences, newcomers' characteristics, reception issues, orientation, technical hurdles, and documentation problems [25]. Stol et al. [26] examined onboarding across settings on a hybrid work continuum to understand how onboarding shapes newcomers' sense of belonging. Their findings indicate that belonging develops through parallel interpersonal and intrapersonal adjustment processes, while the work setting acts as a critical factor and intentionality moderates

the extent to which organizational efforts are translated into employee adjustment [26]. Šmite et al. [27] examined resignation patterns among software professionals from Ericsson perspective between 2016 and 2025, focusing on how the transition from onsite work to fully remote and hybrid models influenced employee attrition. Their findings show that remote onboarding fails to foster organizational attachment and a sense of belonging necessary for long-term retention. Selective return-to-office practices for new hires and their teams are recommended for building social cohesion and reducing early attrition [27].

2.2.1. Bauer's onboarding model

According to Bauer [6] onboarding model, organizations must first determine whether they are best served by formal or informal onboarding. In informal onboarding, newcomers learn their new jobs without an explicit organizational plan. In contrast, formal onboarding process, newcomers receive assistance in adjusting to their new jobs both task-wise and socially. Additionally, organization provides a written set of policies and procedures to support them. Bauer's onboarding model tactics align with institutionalized and individualized onboarding, respectively. Newcomers who undergo formal onboarding with well-planned programs, including awareness on their roles and company norms, are more effective than employees who experience informal onboarding. Bauer identifies four building blocks of successful onboarding, known as the "Four C's": compliance, clarification, culture and connection. Onboarding has short-term outcomes associated with new employee adjustments, which organizations can use to assist newcomers in maximising their onboarding success. Short-term outcomes relate to four adjustments:

- *Self-efficacy* – The self-confidence in performing the job roles.
- *Role clarity* – Determines how well the employees understand their job roles and the expectations.
- *Social integration* – Starting to work with organizational insiders. This supports feeling socially comfortable and accepted by their peers and seniors.
- *Knowledge of culture* – Helps the newcomers understand the organizational culture and their roles within it.

Additionally, Bauer [6] presents six interconnected functions designed to facilitate newcomer support:

- *Recruiting process* helps newcomers to fit into the organizational environment and get to know the insiders and stakeholders. Recruiting is unrelated to onboarding plans. It is important to consider it as a step to begin onboarding.
- *Orientation* helps newcomers to understand important aspects of their job and organization, including its culture, values, goals, history and power structures. Orientation helps newcomers to feel welcome and introduces them to their co-workers and other individuals. This can be conducted either in person or online. Orientation includes discussions, lectures, videotapes, written material and online sources.
- *Support tools and processes* include written onboarding plans; a formal document that outlines the specific timeline, goals, responsibilities and support. Stakeholder meetings or "touching base " meetings help newcomers get the information they need on time. This includes onboarding online, and technology is used to follow the onboarding process.
- *Coaching and support* includes senior employees serving as role models. Mentors help newcomers learn about the organization, offer advice, assist with job instruction, and provide support in social and political contexts. Informal colleagues also provide adequate information. Coaching helps newcomers to follow a successful onboarding.

– *Training* includes hard skills, soft skills and onboarding skills. Training helps newcomers to support their own adjustments.
– *Feedback tools* provide newcomers with feedback and guidance. This is a two-way process during onboarding in which newcomers seek and receive information. Organizations vary in how effectively they use feedback tools.

The models proposed by Van Maanen and Schein [20], Jones [21], and Bauer [6] provide a comprehensive theoretical foundation for understanding how newcomers integrate into organizations. While these models identify what needs to be achieved (e.g., self-efficacy and culture knowledge), there is a gap in understanding how these functions are adapted within LSA context.

3. Method

We employ an exploratory qualitative approach to investigate onboarding within LSA context across four companies in Sweden. Qualitative research is suited for revealing underlying processes within individuals and teams, capturing not only what individuals experience but also how they interpret those experiences [28, 29]. The case study provides a flexible approach for a thorough examination of the complex LSA context [28]. Following an exploratory multiple case study design [28], this research focuses on four real-world projects in LSA context. Using multiple case studies enables broader exploration of research questions and theoretical elaboration. The evidence from multiple cases is often considered more compelling and robust, providing greater confidence in the overall study [30].

3.1. Research context

We conducted our study at Swedish offices of four global software and technology firms, hereafter denoted by these identification codes: *E*, *S*, *T* and *A*. (See Table 2). These cases provide a comparative view of different LSA frameworks, specifically SAFe (Companies E and S) and LeSS (Companies T and A).

3.2. Data collection

Interviews are a highly efficient way to gather rich, empirical data [31], and semi-structured interviews in particular allow direct contact with participants and enable real time data collection [28]. To ensure the depth and diversity of insights, we used purposive sampling [32] to select our interviewees, based on their LSA knowledge and experience. Engaging numerous knowledgeable informants who can view the phenomena from diverse perspectives across functional areas yields meaningful data [31]. We conducted a total of 41 semi-structured interviews: 7 from case E, 8 from case S, 15 from case T, and 11 from case A, representing a diverse range of experience levels and roles (see Table 1).

All interviews were conducted through Zoom, each lasting for 45–60 minutes. Each interview was recorded and stored in a secure repository. In our interviews, the guidelines covered a diversity of questions; the Interviewees' educational and professional background, LSA experience, teamwork practices, onboarding process and training processes. We cross-validated transcripts with the recordings for possible transcription errors. Moreover, we provided the interviewees with finalized transcripts to facilitate member checking.

Table 1. Interviewees background

Case E			Case S		
ID	Role	Professional experience	ID	Role	Professional experience
EP1	Integration specialist	1 year	SP1	Tester	7 years
EP2	Project manager	2 yeras	SP2	Developer	20 years
EP3	Scrum master	7 years	SP3	Tester	12 years
EP4	Developer	20 years	SP4	Project manager	5 years
EP5	Business owner	10 years	SP5	Designer	5 years
EP6	Scrum master	5 years	SP6	Designer	5 years
EP7	Analyst	3 years	SP7	Designer	2 years
			SP8	Designer	8 years
Case T			**Case A**		
ID	Role	Professional experience	ID	Role	Professional experience
TP1	Product owner	10 years	AP1	Developer	7 years
TP2	Security master	3 years	AP2	Team lead	25 years
TP3	Developer	1 year	AP3	Developer	23 years
TP4	Developer	10 years	AP4	Developer	4.5 years
TP5	Developer	20 years	AP5	Test lead	16 years
TP6	Developer	25 years	AP6	Developer	1 year
TP7	Project manager	20 years	AP7	System engineer	15 years
TP8	Product guardian	3 years	AP8	Project manager	14 years
TP9	Security master	1 year	AP9	Developer	8 years
TP10	Architect	16 years	AP10	Developer	12 years
TP11	Developer	1 year	AP11	Developer	21 years
TP12	Senior software engineer	7 years			
TP13	Developer	10 years			
TP14	Developer	12 years			
TP15	Tester	2 years			

Table 2. Case company descriptions

Case company	Description
E	Medium-sized software and technology company comprising cross-functional agile teams practising SAFe, specialized in Enterprise Resource Planning (ERP) and digitalization.
S	Large institution providing financial and insurance solutions, comprising cross-functional agile teams practising SAFe.
T	Large Nordic software and service provider that comprises cross-functional, self-organized agile teams developing a test tool for telecommunication operators practising LeSS.
A	Medium-sized software and services company consisting of X and Y project teams. Team X is developing a car rental software, whereas team Y is working on a retail chain with web management modules. Comprised of cross-functional agile teams practising LeSS.

3.3. Data analysis

We used thematic analysis [33], because it is recognized as a valid technique for a transparent and systematic procedure for producing codes and themes from qualitative data. Thematic analysis [33] follows a six-phase process (i.e., familiarising ourselves with data, generating initial codes, searching for themes, reviewing themes, defining and naming themes and producing the report) to systematically identify, analyze, and interpret patterns in data,

Table 3. Phases of thematic analysis

Phase	Description of the process
1. Familiarising ourselves with data	Repeatedly reading the data to search for meanings and patterns. Highlighting key information and taking the initial notes.
2. Generating initial codes	Coding interesting features of the data systematically across the entire dataset. Match code with data extracts that demonstrate the code and collated together within each code.
3. Searching for themes	Refocusing the analysis at the broader level of themes. Collating different codes into potential themes and collating relevant data extracts within the identified themes.
4 .Reviewing themes	Checking if the themes align with the coded data extracts (Level 1) and the entire data set (Level 2). Generating thematic mapping of the data analysis.
5. Defining and naming themes	Defining and further refining the themes that will be presented for analysis and analysing the data within themes. Identifying the essence of what each theme is about, and what aspects of data each theme captures. Identifying the story that each theme tells, and considering how it fits into the broader overall story that we are telling about our data.
6. Producing the report	Providing sufficient evidence of the themes within the data, e.g., using enough data extracts to demonstrate the prevalence of a theme. Choosing vivid data extracts as examples to capture the essence of the point being discussed. Relating the analysis back to the research question and literature.

Table 4. Themes, sub-themes and codes

Theme	Sub-themes	Codes
Onboarding functions	Recruiting process	Onboarding in recruitment process
	Orientation	Introductory meetings, system overview presentation, team-led grooming sessions
	Support tools and processes	Onboarding template/checklist, system documentation, contact support, standup meetings, tools accessibility, provisioning accounts, environmental setup
	Coaching and support	Dedicated mentors, scrum master, product owner support, informal peer support, innovation sprint, question-asking culture
	Training	Learning materials, online courses, task-based, case-based and competency-oriented learning, pair-programming, code reviewing
	Feedback tools	Continuous feedback, retrospectives
Adjustments	Self-efficacy	Adaptation to work
	Role clarity	Role expectations
	Social integration	Blending with teams, ad hoc-meetings, morning-talks, one-on-one rotating conversations, planning poker, team-building meetings
	Knowledge of culture	Agile training, team norms
	Technical integration	Technical meetings, tech fika/coffee time

moving between raw interview transcripts and coded extracts to prepare a description of the onboarding phenomenon (see Table 3).

Thematic analysis identifies and interprets the key features of the data guided by research questions and broader theoretical assumptions. The significance of thematic analysis lies in its flexibility to summarise the key features of large amounts of data, providing a "thick description" of the dataset [33]. Thematic analysis involves deep and prolonged immersion in the data, requiring researchers to engage in thoughtful and reflective analysis [34]. It helps to identify patterns within and across data relating to participants'

lived experience, views, perspectives, behaviour and practices, to understand what they think, feel and do [33]. Analysis involved a "recursive" process that constantly moves back and forth between the entire dataset, the code extracts of the data being analysed, and the analysis of the data produced [33]. Moreover, Braun and Clarke [33] propose fifteen criteria to ensure good qualitative analysis, highlighting the processes of transcription, coding, analysis, overall completion, and written report. We used NVivo (Version 14) software to systematically identify the patterns grounded within the data. We mapped generated codes to the onboarding functions and employee adjustments in Bauer's model (see Table 4).

4. Results

This section presents key findings related to onboarding in LSA context based on our qualitative analysis. This directly answers our research questions which are presented in Section 1. We mapped case companies practices with the lens of Bauer's onboarding model [6], containing six onboarding functions (i.e., recruiting process, orientation, support tools and processes, coaching and support, training, and feedback tools) and new employee adjustments (i.e., self-efficacy, role clarity, social integration, and knowledge of culture). Building on this model, we introduce "technical integration" as an adjustment during LSA onboarding. We mapped the presence of these onboarding functions with "Y" (Yes) indicating their presence in each case company (see Table 5) and adjustments in Table 6.

4.1. Functions

This section describes the onboarding functions of case companies, organized into six categories. Together, these functions explain the factors involved in the onboarding of newcomers in LSA projects.

4.1.1. Recruiting process

Recruiting process is the stage where newcomers receive their first impression and create expectations for their specific role. A well-planned onboarding process creates a motivating and captivating environment for newcomers. *"We have a structured onboarding process that is led by our recruitment person. Depending on when a new employee starts, they will have a tailored introduction plan"* (EP5). The recruitment process needs to be consistent and make adequate efforts to get the right person for the role, because new recruits in an established organization fill the gap left by an existing employee in a specific area of expertise. Therefore, a precise balance of technical expertise and personality must be considered during the onboarding process. *"There will be a gap that needs to be filled. I can either say I need someone with much experience or it's okay with someone directly from the university"* (TP1).

The careful consideration of a person's attitudes and behaviours is emphasised as a driving force to be considered, given their effect on team dynamics when working across multiple teams. *"You need to have the basic skills to join the team, but then personality is everything."* and *"They sit on the first meeting, speaking about techniques* [...] *and then make the first assessment. Is this a person that can fit into the team? Then the team meets this person. I [role in team] only ask about a kind of personal question to check what kind of person this is?"* (SP1). Personality tests are used as one of the evaluation criteria in

Table 5. Onboarding functions in four case companies

Onboarding functions	E	S	T	A
Recruiting process				
Onboarding in recruitment	Y	Y	Y	
Orientation				
Introductory meetings	Y			
System overview presentation				Y
Team-led grooming sessions		Y		
Support tools and processes				
Onboarding template/checklist			Y	
System documentation			Y	Y
Contact support				Y
Standup meetings	Y		Y	Y
Tools accessibility		Y	Y	Y
Provisioning accounts				Y
Environmental setup			Y	Y
Coaching and support				
Dedicated mentors	Y	Y	Y	
Scrum master support				Y
Product owner support				Y
Informal peer support	Y	Y	Y	Y
Innovation sprint		Y		
Question-asking culture			Y	
Training				
Learning materials	Y		Y	
Online courses			Y	
Task-based learning		Y	Y	Y
Case-based learning				Y
Pair-programming		Y	Y	Y
Competency-oriented learning			Y	
Code-reviewing	Y		Y	Y
Feedback tools				
Continuous feedback	Y			
Retrospectives			Y	

Note: "Y" indicates the presence of these functions in each case company. Each company represented with a code name: E, S, T and A.

hiring processes. *"We have a set of questions and we have different profiles and during those interviews, to ensure that we check the personality"* (SP2).

This indicates that the recruitment process is linked with structured planning, even before onboarding activities begin. The recruitment process does not end just after hiring newcomers, but continues until newcomers successfully integrate into their work environment both technically and socially.

4.1.2. Orientation

Orientation is an essential step in integrating newcomers into their designated roles and tasks. Case E has a comprehensive 100 days plan which includes two weeks focus on introductory meetings to help newcomers adjustment in their roles. *"Our onboarding process is quite rigorous. We have a lot of meetings for new employees to make sure they feel welcome"* (EP5).

Whereas case A has a system overview presentation, which is technical. Newcomers get high-level information on the system and its integrations, which are expected in their LSA projects. *"I take a brief overview presentation about the system, how things are connected and everything."* and *"We give them some overview knowledge about the systems"* (AP11).

Team-led grooming sessions are another strategy employed to facilitate newcomer learning. This provides the opportunity for the newcomers to engage with the team and understand their technical expertise and business values. This is seen as a team's collective effort in helping newcomers to integrate into their workflow. Case S has an onboarding process between three and six months. *"I think it's an important session for a new person to be together with the team in the grooming sessions"* (SP2).

4.1.3. Support tools and processes

Support tools and processes help newcomers gradually integrate into their tasks. Case T creates an onboarding template in Jira, including all the steps required for a formal introduction. This serves as a checklist for newcomers and includes information regarding tools, different accesses, training videos, e-learning and security rules. This onboarding story is cloned every time a newcomer joins. *"They have like an onboarding story in Jira, which they have gathered, like all steps on the introduction"* (TP2) and *"We have a template for an onboarding story. So it's a newcomer, I clone that story"* (TP1). Case T adopts a more intentional approach to integrating newcomers, improving their onboarding experience and making the process smoother and more effective for all involved. *"We realized that it would be really good to have a ready program that you can put in front of a new person"* (TP6).

In cases T and A, newcomers receive distinct system documentation for reference. This guides newcomers through the ways of working, provides a list of things to download, and initiates learning procedure. *"We have multiple documents and pages to guide them through the environment setup"* (TP12) and *"Here is the documentation page where you can read up on all the documentation"* (AP6). In case A, the information is documented on dynamic pages in Confluence, listing the information of team members whom newcomers can contact for support. Newcomers can contact this person to address their system-related issues. This is a dedicated role assigned with adequate time to guide newcomers on technical inquiries. *"This is the dedicated person, and they should have time to actually answer any questions you have regarding the system"* (AF11). The approach reflects active learning through real work, with adequate support available as needed. *"Here are people that you can ask for help"* (AP3). Moreover, this establishes a more practical approach than long, overwhelming lectures. The documentation emphasizes ways of working, including essential processes and practices, without providing extensive details. *"We are pretty good at documenting how things work, how you set up your environment, roadmap, and onboarding processes"* and *"I think we are trying to be as pragmatic as possible and, try to have a very condensed page in the Confluence specifying ways of working"* (AP2).

In cases E and A, daily standups are used for synchronisation and transparency, helping newcomers understand team's status and challenges. *"The best way to learn the system is through a big project to work with, but it is not always that easy. So we make it work by helping each other through standups"* (EP7) and *"Straight away, we have let them into these standup meetings. So they can get some understanding of how things would work"* (AP4). In case T, teams participate newcomers in standup meetings from the beginning, helping themselves immerse in the team's routine work while providing a regular checkpoint to monitor their onboarding progress. standup meetings function as a supportive mechanism,

allowing teams to identify and address any obstacles that newcomers encounter. *"We are checking how it is going with their onboarding, as sometimes they have problems like they don't get any access, get stuck somewhere and have code issues"* (TP5). standup meetings also offer a cooperative platform to discuss ongoing work, gather feedback, check on the progress of other teams, and understand and resolve technical discrepancies in greater detail. This is especially helpful for newcomers as they acclimate to their responsibilities. *"They discuss what they are doing and are now in this step, and they want feedback from another team"* (TP15). standups provide an interactive platform for distributed teams to connect and meet co-workers. This is a significant opportunity for new employees to integrate into the team. *"[…] Especially for new employees, it's a great way to get help and support in the beginning"* (EP6).

Newcomers should gain access to the required tools, configure their accounts and set up their working environment as soon as they get to know the team. Different cases have their own unique ways of integrating tools. In cases S and T, newcomers are first provided access to the tools, followed by an introduction to the areas of work or the domain. *"We have a program helping him first to have the necessary tools to be able to start up, then we would present the area or doing"* (SP2) and *"There are lots of tasks in them for setting up the environment and doing the first well, sandbox environments and so on"* (TP8). In case A, newcomers are initially provided with an overview of their system; their devices, accounts and environment are configured, and they are guided to start their learning. *"We will set up the computers and environments and all the accounts and access rights that they need, or at least put in orders for it"* (AP10) and *"We give them some overview knowledge about the systems and then we try to get their environments running up and they can start to work"* (AP11). Case A has implemented substantial technical improvements to its onboarding process, eliminating bottlenecks that delayed newcomers' ability to begin their work. *"We automated most of our installation processes and things that previously took one or two weeks just to get set up with the environment"* (AP9).

4.1.4. Coaching and support

Coaching and support is an essential element of successful onboarding. This ensures that newcomers are systematically integrated into their roles and teams. Each case assigns dedicated mentors to their newcomers. These mentors assist newcomers in understanding team routines, addressing emerging questions, and gradually helping them adapt to their work environment. *"Their [mentor] responsibility is just to support, and make sure that the new person has access to the team"* (EP1) and *"They are someone you can go to with any questions"* (EP6). Mentors invite newcomers to regular meetups, supporting them in assessing their progress as they settle into their roles. This practice continues until newcomers can confidently manage their responsibilities. *"I got a mentor when I started, and we had meetings once every two weeks at first. Now we don't have those meetings any more. We had them like the first six months"* (EP2).

When a new employee joins, both cases T and S ensure that they receive a dedicated mentor as the first point of contact for any clarifications. Assigning a mentor helps new employees feel supported and monitored as they transition into their roles. Mentorship not only assists them in their work but also helps build their confidence over time, which is a significant benefit of mentoring. *"Each new person gets a mentor in the team and they [newcomers] can come and ask questions, like the first person"* (TP2) and *"Mentors are supposed to help the newcomer to be a little bit more concerned"* (TP6). Mentors collaborate

with newcomers on all the cases they involve. Regular assistance and substantial attention enhance newcomer's personality and foster collaboration with peers. *"They are working on all the cases that he [newcomer] is doing"* (SP3) and *"The mentor is mostly like a buddy at your new work, someone who makes you feel welcome"* (EP6). The frequency of mentoring meetings is tailored to each individual's progression within the team and system integration. Mentoring fosters familiarity and engagement for newcomers within the team. *"The frequency of these mentor meetings is different for each person"* (EP4).

Case A employs a product owner and scrum master to provide coaching and support. In LSA settings, it is equally important to maintain both technical and business perspectives. *"We [teams] prolonged a meeting, and he [product owner] will go through his view of the system."* and *"He [scrum master] spends some time getting them on board and getting them up to speed, and introducing things, the applications to them"* (AP4).

In all cases E, T, S and A teams willingly support their newcomers in need. Team members collaborate closely, providing informal support to help newcomers overcome adaptation challenges, even in the presence of a formal mentor. *"We all try to help new employees in team and involve them in projects as well. The mentor doesn't have to learn everything for new employee, but we all try to help with the learning process"* (EP6) and *"You [newcomer] have a mentor in the team that is supposed to help newcomer to be a little bit more concerned. But we try to help each other in team"* (TP6). Teams provide ongoing support and assistance to help newcomers assume more responsibilities and integrate successfully. *"New member work together with someone else in the team, taking on a work item for himself with the support from the other person in the team"* (SP2) and *"They [newcomers] can do some parts of their own in some parts together with some of the older guys in the team, but we try to help all each other in the team"* (TP6).

Team members voluntarily stepped in to assist in the absence of a mentor. They support each other not because they are assigned to do so, but because the team values mutual support. The success of onboarding relies on an informal culture of helpfulness, with team members proactively filling gaps whenever needed. *"The newest guy, he got one mentor, and he was away. So I just took over because he needs help. I will help"* (TP2) and *"We were four new people in the group that came simultaneously. We worked on more minor bug cases, and had much help from the team"* (AP6). Team members also make an extra effort to physically meet and help new employees early on. Experienced team members adjust their physical presence to support newcomers despite their remote work preferences. Face-to-face interactions are recognized as important for building social connections. *"Often they are newly employed, and tend to be at the office more in the beginning, and then we [experienced team members] tend to be at the office more in the beginning as well, to be there to physically meet the person [newcomer] and to sit down with the person. But for me, we can choose from three variants"* (AP3).

Case S teams operate on a structured timeline known as "product increment". This is a twelve-week cycle, with the final two weeks dedicated to an innovation sprint. During this period, team members, including newcomers, are given several (three to five) days to work on their technical interests. This time is intentionally structured and time-bound. Newcomers who regularly participate in structured onboarding tasks gain a sense of freedom, which allows them to deepen their understanding, explore broadly and enhance their confidence. Even though newcomers might not expect to be involved in these activities early, team continues the practice because they believe it is important for learning and innovation. *"They [newcomers] ask: Well, what should I do?. You can do what you want! They will have three, four, five days that they can do whatever they want and it works good"* (SP1).

Case T teams encourage a question-asking culture for newcomers. Team dedicates a focused hour to addressing newcomers' questions, thereby reducing newcomers' anxiety, and minimizing potential distractions for experienced team members. *"If you have any questions, just write them down and we can have an hour when we talk about it."* (TP2) and *"They can ask questions or have a session with questions with someone in the team or the whole team."* (TP5). The structured question sessions serve a dual purpose. Beyond answering newcomers' questions, these sessions help maintain a constructive discussion and mutual understanding between newcomers and team regarding the tasks being performed. This reflects an active learning behaviour, as newcomers gradually grasp the context before delving into more complex works. *"Trying to start with minor commits and asking questions. I'm trying to get the idea. What is we really doing here? What am I doing? Really? What is this? Where? What is possible to do and so on?"* (TP15). Asking questions presents a fundamental team value. This reflects a conscious effort to build psychological safety by encouraging newcomers to feel comfortable, avoiding hesitations and embarrassment. *"We [team] always say that to the newcomers. It's always okay to ask questions. No question is too silly to be asked, and we always try to say yes"* (TP6). Team members openly acknowledge that even experienced team members can make mistakes, emphasizing the natural consequences of complex work, thereby encouraging transparency and continuous improvement. *"It is a complex system. So we do mistakes as well"* (TP6).

4.1.5. Training

Training plays an important role in the integration of newcomers into software and technology companies. Formal training (learning materials and online courses) provides a structured, systematic approach to integrate into workflows, while informal training (case-based learning) provides practical, case-based learning.

Cases E and T offer a package of learning materials to newcomers. This consists of recorded knowledge-sharing sessions and courses that newcomers may need to repeat several times. *"We have started to build up with some learning material and have a lot of knowledge sharing sessions that we record"* (EP6) and *"There are some recorded courses that we are able to look at, as part of the introduction"* (TP3).

Case T offers online courses to newcomers to acquire technical skills and knowledge. This indicates a more standard learning approach, which is consistent. *"There are a number of e-learning training courses that they [newcomers] need to participate in during the first weeks"* (TP1) and *"The onboarding process is very focused on setup and going through different courses online"* (TP15).

In cases S, T, and A, newcomers integrate into the team through task-based learning. In case S, newcomers work together with another team member and take responsibility for a task item while receiving guidance and support. *"[…], taking on a work item for himself with the support of the other person"* (SP2). In cases T and A, newcomers start with narrow tasks and progressively familiarise themselves with the workflow. *"They [team] assigned me some simple tasks and provided on-the-job training to get familiar with how we work in this team"* (TP13). Starting with manageable tasks helps newcomers build their basic competencies before they progress to more complex development work. *"The team members created some tasks for me to build up my basic competence to work in this team"* (TP13).

Focusing on a narrow task streamlines development, facilitating a smooth technical transition. This further reduces the cognitive load of newcomers by limiting their initial scope of understanding and preventing them from becoming overwhelmed by the system's

complexity. *"We [team] assign a task to the person that may be quite narrow tasks, so the person doesn't need to know everything, but just a part of the system"* (AP1). Case A views onboarding as a continuous process that gradually integrates into the domain and development environment. *"It's an ongoing process after that, starting with more manageable tasks"* (AP2). Newcomers are assigned to explore and implement a new tool that integrates learning with practical contribution, enabling them to engage early in projects. *"One person very recently joined. He's just stepped out of college and he's working on those scopes. We [team] kind of wanted to try that tool within the application"* (AP4).

Teams do not expect newcomers to have comprehensive system knowledge; however, they do expect responsible software development accompanied by an adequate contextual understanding. *"It's not mandatory for everyone to know everything. But they should at least know what they're doing and what they are affecting by their changes"* (AP2). Team progressively build newcomers' understanding through "small steps", providing sufficient context for each task. *"When we're specifying the issues or tasks to be done, I try to fill in a little bit more about how it works in small steps"* (AP2).

Given the importance of understanding internal and external dependencies and architecture perspectives in LSA, case A emphasises a case-based learning approach. Such tasks are given as an assignment, enabling newcomers to learn and expand their domain knowledge while moving forward in the workflow. *"There is a need to explain the dependencies between that application and other components, internally or externally, on the architecture"* and *"We start from there, a narrow part of the system, an assignment, a task"* (AP3). Team strategically assign newcomers to a safe learning environment. *"We have identified one product that is low-hanging fruit. So to say it's not used that much. We've identified that and slightly introduced this person to the code and the actual code base, and how all the systems are kind of interconnected"* (AP10). The term "low-hanging fruit" refers to the team's effort to create opportunities for newcomers to learn and experiment in a low-stakes environment. This approach allows them to gradually become familiar with system architecture. By offering learning opportunities that have a real impact, newcomers can enhance their competence and confidence before moving into more complex systems.

Developing competency-oriented ramp-up program to serve and adjust the competencies of every newcomer is a beneficial approach. *"Some people need a good foundation and ramp-up very fast, while others can't do long studying and need to be integrated into the development cycle faster"* (TP12). Case T implements competency-building sessions, demonstrating a formal commitment to continuous professional development. Newcomers are being supported through this initiative to address their competency gaps. *"We [team] can spend one hour on anything, any competence that we want to build up or learn something"* (TP15) and *"I think everyone is lacking competence in general. But if you compare me with other people, I lack competence because I'm very new"* (TP14). By facilitating more opportunities, case T connects newcomers with employees who have recently completed their onboarding tasks. This allows newcomers to receive guidance and support based on the experiences of those who have recently gone through the onboarding process. *"What we are trying to do is to direct them to someone that has recently done it [onboarding]"* (TP5). Experienced employees are willing to assist newcomers with tasks that align with their specific experience. *"Everyone has a different background. Maybe the newcomer is more experienced than me. So, if I encountered a problem during onboarding that the newcomer did not face, then I would suggest that they refer to my onboarding process"* (TP13).

Pair programming is a structured practice in agile, but learning emerges naturally through collaboration and communication. In case S, T, and A newcomers paired with

co-workers. Newcomers observe the works of experienced colleagues, collaborate with them and receive guidance until they are capable of working independently. *"First sitting together with another one with the pair programming"* (SP2), *"We sit down with this new person a lot at the beginning and pair programming* (AP3)", *"The new person will do pair programming, shadowing with someone who is developing until they are ready to develop independently"* (TP12) and *"Always have someone they can code with, the newest member of the team, he was paired with him [senior] for a little while"* (AP9).

Case A identifies code reviewing as a dual-purpose mechanism intended to enhance newcomers' learning while promoting knowledge sharing within the team to support high-quality code development. *"We need to do more coding reviews because the other two people in the project are quite new. So for them, it's more important to have coding reviews"* (AP1). Instead of assigning code review to newcomers during their onboarding tasks, case A ensures that newcomers receive reviews from the entire team. This approach helps newcomers improve their coding skills and understand the standards and practices effectively. *"You [newcomer] won't be code reviewing for a while. So many people should code review your code"* (AP6). Similarly, case E views code reviewing as a useful approach and is designed to be implemented, given the benefits of learning from each other and providing a platform to review newcomers' code more concisely. *"It might be a good idea to start doing code reviewing. When we have a new employee, I can review his code in the beginning and learn from each other"* (EP6).

Moreover, case T employs a structured code review process during development, prior to deployments. Multiple reviews are encouraged, allowing developers to invite their team to review their code. *"I was developing this code with my colleague and was reading through this code many times, and then we say to the team, you should also review it"* (TP3). Team identifies code reviewing as a strategic approach to enhance analytical skills and improve understanding of each other's code. *"Code reviewing will help us learn how other people approach a problem and how we approach a problem"* (TP15). Even though code review seems challenging for newcomers, case T encourages them to actively participate in order to improve code quality. *"I find the code review process to be a challenge, but it's necessary for maintaining code quality"* (TP15). However, feedback from experienced developers assists newcomers in producing clearer and more readable code. *"Some more experienced people can give very good comments about the code. For example, our new team member S has less experience, and his code needs more improvement. We will leave comments about how to write the code in a better and more readable way, and he will fix the comments"* (TP13).

4.1.6. Feedback tools

Feedback tools determine whether continuous assistance is required. From the very beginning, newcomers engage in regular meetings with their mentors. These meetings function as a feedback loop, ensuring both learning and adaptation. *"We had them in the first six months, and then we both felt the need. Like, don't need them more, and I would contact her if I have any questions"* (EP2). Teams often collaboratively integrate newcomers, being approachable and open to requests for clarifications and guidance. *"We try to help each other in the team, so they [newcomers] can just reach out to anyone if they have some questions"* (EP6).

Case T uses retrospectives as a platform for sharing collective feedback at the team level. Retrospectives facilitate collective reflection and process improvements while providing a psychologically safe environment for newcomers to ask questions and suggest improvements

within teams. *"We get feedback during our scrum meetings, we have the retrospective where we give feedback, but that's more on a team level"* (TP11) and *"But we do have team members who have their work shared or at least tested by some other teams. In that case, they do write the feedback in the retrospectives"* (TP15).

4.2. Adjustments

This section describes the employee adjustments in case companies, organized into five categories, along with "technical integration" as an adjustment in LSA. Together, these adjustments describe how case companies support newcomers in maximizing their onboarding success. We mapped the presence of these onboarding adjustments with "Y" (Yes) indicating their presence in each case company (see Table 6).

Table 6. Onboarding adjustments in four case companies

Onboarding adjustments	E	S	T	A
Self-efficacy				
Adaptation to work	Y		Y	
Role clarity				
Role expectations		Y		Y
Social integration				
Blending with teams			Y	Y
Ad hoc-meetings			Y	
Morning-talks				Y
One-one-one rotating conversations			Y	
Planning poker				Y
Team-building meetings	Y		Y	Y
Knowledge of culture				
Agile training	Y	Y	Y	Y
Team norms		Y		
Technical integration				
Technical meetings			Y	Y
Tech Fika/ Coffee time			Y	

Note: "Y" indicates the presence of these adjustments in each case company. Each company represented with a code name: E, S, T and A.

4.2.1. Self-efficacy

Self-efficacy focuses on newcomers' confidence to learn and adapt to the team workflow. Successful onboarding creates a supportive and adaptable work environment, fostering competence. *"The department manager and I still have talks every two weeks to discuss all my development and my plan forward"* (EP3) and *"When I started working in this team, it was relatively smooth. It took about one or two months to get familiar with development environment and team"* (TP13). Moreover, case T Team helps its members set goals. Every member is provided with an allocated time to focus on skill development, improve their expertise, and pursue their interests. *"You can focus your competence development on this [allocated time], and use that for going in that direction, or if it's something that you think is interesting, then you can go in that direction"* (TP11).

4.2.2. Role clarity

Role clarity assesses newcomer's role, centered around their knowledge, skills, workflow integration and teamwork spirit. *"We get a new team member, and I can see how good teamwork is flowing and the level of skills the person has"* (AP1). This highlights the team's expectations and desires about the newcomer's performance and their role within the team. Additionally, Case s team expects new employees to value cultural expectations as much as technical expertise. Roles are defined by how newcomers should engage and contribute. Newcomers are expected to pursue their defined responsibilities with appropriate attitudes and behaviours. *"You have to be able to work in this type of team, you need to be curious, and want to develop yourself, want to take on a new assignment, and be interested both in the business and in what the team do with you"* (SP2).

4.2.3. Social integration

Social integration incorporates newcomers into teams, facilitating collaborative participation. Blending newcomers with teams brings higher engagement and motivation. *"If you get a lot of new people in, you can not just put a new team with all new people. It is better to spread them out and then form a new constellation"* (TP8) and *"When I wasn't quite familiar with debugging errors when running test cases, my team members helped me"* (TP13). A supportive team culture fosters understanding, respect and patience, enabling newcomers to integrate smoothly. This integration enhances the effectiveness of formal onboarding activities. *"I think it comes down to everyone respecting each other, giving each other additional time, and being patient"* (AP9).

Ad-hoc meetings serve as an excellent opportunity for newcomers to connect. These informal gatherings foster relationship-building and cultivate a sense of belonging, both of which can significantly enhance collaboration among team members. *"Unplanned, informal meeting just when bumping into each other"* (TP1).

Case A values personal connections as much as professional collaborations and establishes a daily routine to reflect this priority. Teams intentionally allocate time for team mingles called morning talks, fostering a psychologically safe environment that helps newcomers overcome the fear of embarrassment when asking questions. *"Every morning, we join Person xyz, and talk about anything that is not work-related"* (AP10) and *"We have morning talk 15 minutes every morning before the standup and you cannot talk work-related stuff there"* (AP3).

Moreover, case A, T and E organize frequent team-building activities to provide regular breaks in ongoing work, fostering a relaxed and exciting environment where employees can enjoy and get to know one another. Team members, along with newcomers, engage in fun activities such as online games, scribble drawing, and identifying geographic locations. These activities have been appreciated by teams for making regular work more effective. *"We also have added team building meetings each week just to get a break in the work and talk about other stuff"* (AP11). Cases T and E involve various physical team-building activities such as team lunches, after-work drinks and team outings like bowling or mini-golf in regular intervals outside of their formal routine. *"We are going to places, eat something, and there are more interactions"* (TP12) and *"We do a lot of fun activities and after work together"* (EP6). In addition to these activities, case T teams also maintain a digital social platform, such as team chat applications, where team members share entertaining discussions. These informal interactions foster a sense of belonging, contributing to a cohesive team culture.

"We also have a team chat where we can socialize and share non-work-related things like funny memes or interesting articles" (TP9).

Case A teams use planning poker as a collaborative tool, allowing each member to contribute their insights into the task planning process. This method not only promotes a shared understanding among the team members, but also cultivates a sense of collective responsibility. *"We try to always do some activity that engages everyone. So, for estimation meetings, we do Scrum poker"* (AP11).

Case T organizes one-on-one rotating conversations. This in-person event provides participants with the opportunity to engage in discussions with one another. Newcomers are welcomed and encouraged to share their insights with the team, as they will have the opportunity to conduct a five-minute discussion with each attendee. Although the limited time can create stress, this event is still considered a valuable opportunity for newcomers to connect with the team members with whom they have not interacted. *"They got call from each other for 5 minutes and then it was a new person that they should call, and so on* [...]" (TP5).

4.2.4. Knowledge of culture

Knowledge of culture is necessary for shaping newcomers' adaptation into teams and workflow practices. For instance, onboarding should communicate why specific practices are adopted and how newcomers use a specific workflow. Using agile practices helps to improve collaboration, feedback loop and adaptability. Case S conducts flexible SAFe training, in the form of a continuous program called "Leading within SAFe" which lasts for several days or a week. Newcomers joining the teams are encouraged to participate in the SAFe training from the outset. *"All new people who come here come to work, they will have some kind of education [SAFe]"* (SP1) and *"They [newcomers] will get this SAFe training upfront when they join"* (SP2). Case E conducts introduction sessions in Agile, explaining how the team adopt agile way of working. Agile culture fosters a sense of innovation and motivation among employees. *"As a part of the introduction, there are advanced courses in Agile"* (EP1) and *"I love the Agile process because it forces me to think about things I never thought about before"* (EP4).

Moreover, scrum master, in case A team also assists newcomers in conducting agile scrum ceremonies, *"About how the sprint works"* (AP4). Employees have gained experience and learned the agile way of working over time. *"When we started, we had no experience. So, it started with baby steps"* (AP1). Case T conducts agile courses in which all its employees participate. The full-day course explains the standard agile practices and how the team utilizes them. *"You went through how to use and how we are using Scrum"* (TP3). Though it takes time for new employees to adapt to agile, case T appreciate the agile way of working. Implementing agile has enabled the teams to compress their tasks into flexible sprints instead of long-yearly planning. *"I think it's nice to get a better overview of all the work"* (TP11).

Team norms play a significant role in shaping team practices, and newcomers are expected to adhere to certain ethical guidelines and boundaries. These team practices are negotiable and flexible, provided that there is clear communication. This approach helps newcomers develop trust, accountability and respect for team norms, facilitating their successful integration into the teams. *"I have to get them to adopt the rules and accept them because you're not allowed to look into your mobile phone during a meeting. But if*

they tell the others in the team, then it's okay, but it's not okay if you don't communicate it" (SP2).

4.2.5. Technical integration

This is an emerging theme that we identified beyond Bauer's model, as the complexity of large-scale systems creates challenges that neither role clarity nor cultural knowledge can adequately address. Technical integration is particularly important in an LSA environment, where merely defining role expectations, adapting to team workflows, and relying on self-directed learning are insufficient to support newcomer integration. Technical integration addresses how newcomers develop a sufficient understanding of interconnected systems that even experienced members do not fully master.

Technical integration involves regular team discussions and continuous updates on system know-how. Case A teams participate in technical meetings, report on and share information on what they implement. These open meetings are conducted regularly to keep all team members updated and provide an overview of their ongoing tasks. *"We can take up things that maybe we have made a new tool that we know you also should do"* (AP1).

In case T experienced team members lead technical discussions, which function as a platform for newcomer knowledge sharing. *"Explain things deeply for our team because we have different backgrounds"* and *"It's a very specific area to get newcomers into team"* (TP5). Moreover, in case T, all team members participate in "cluster meetings" in which they discuss the achievements made in different areas of their sub programmes. The team also hold regular informal domain knowledge sharing sessions known as tech fika or tech coffee time, in which they share information on specific topics. *"It could be that one person reads up on the subject and tells the other about it"* (TP8). Although these sessions proactively produce knowledge, newcomers perceive them as demanding because they require preparation for active participation. *"It pushes me a bit because I also have to read up before and actually have something with me"* (TP11). However, they also recognize this as a motivation for learning. *"That kind of motivates me as well"* (TP11).

4.3. Onboarding challenges

This section presents a systematic classification of onboarding hurdles into organizational, individual and contextual challenges, describing the difficulties encountered during the onboarding of newcomers (see Table 7).

4.3.1. Organizational challenges

Lack of an organized onboarding process: While newcomers anticipate a formal introduction to team members and other relevant stakeholders, a lack of a structured onboarding process with adequate orientation limit their access to people and information, thereby hindering their early engagement. *"I don't think there's a formalized way of introducing people, and it was a lot up to me trying to find information"* (AP5).

Case E considers it is valuable for newcomers to be included in the projects from the very beginning, as this provides them with a gradual learning process and an opportunity to learn through observation. However, this approach can be challenging, as learning the system through a larger project is not always straightforward. *"It's all about timing. The best way to learn the system is through a big project to work with, but it's not always*

Table 7. Onboarding challenges

Onboarding challenge	Sub-challenge	Description
Organizational Challenges	Lack of an organized onboarding process	(1) Lack of a formal introduction; inadequate access to people and information. (2) Learning opportunities depends on availability of newcomers and ongoing projects. (3) Lack of a common understanding on onboarding approaches among team. (4) Sprint time not deliberately protected for onboarding. (5) Lack of an organized knowledge-sharing approach across applications. (6) Lack of a flexible onboarding approach to accommodate individual needs (7) Lack of clarity regarding prioritization and decision-making. (8) Lack of the supervision of a mentor.
	Recruitment difficulties	(1) Hiring skilled personal from diverse backgrounds. (2) Finding recruits with combined (technology and domain) skills. (3) Recruitment not aligned to team requirements.
	Team-facing onboarding challenges	(1) Lower availability of experienced team members vs higher availability of newcomers. (2) Team instability due to frequent newcomer onboarding. (3) Assigning inexperienced newcomers to new customer projects. (4) Newcomer training amid align rapid organizational growth. (5) Newcomer's integration pace affect team effectiveness. (6) Team cost effort, time and energy for training newcomers with little experience on projects.
Individual Challenges	Cognitive overload	(1) Sheer volume of information presented in tightly scheduled introductory meetings. (2) Maintaining a working pace parallel to peers. (3) Frequent switching between tasks, projects and meetings.
	Barriers to psychological safety	(1) Newcomers been observed and reported to progress to management. (2) Newcomers' limited ability to self-advocate unlike experienced team members. (3) Newcomers rarely engage in team discussions. (4) Newcomers remain silent: Lack knowledge to ask questions. (5) Newcomers feeling comparatively inexperienced.
	Knowledge gap and retention	(1) Gaps in domain knowledge (2) Difficulties in integrating highly specialized/narrowly focused expertise into teams.
Contextual Challenges	System complexities	(1) Domain complexity. (2) Insufficient system understanding due to limited exposure to components and processes. (3) Numerous application and unclear naming conventions. (4) Difficulties in acquiring input from experiences team members. (5) Multiple team involved in development.
	Remote onboarding	(1) Difficulties to integrate with cross-team collaboration and planning. (2) Gaps in availability and visibility among team members.
	Cultural sensitivity	(1) Adaptation to different working styles. (2) Adaptation to different organizational structures; hierarchical vs flat. (3) Adaptation to different decision-making styles.
	Agile-related issues	(1) Gap between textbook agile values and actual agile practices. (2) Transparency induce performance expectations.

that easy" (EP7). A lack of adequate projects limits newcomers' opportunities to learn and contribute. *"Sometimes we don't have so many projects going on. So, I think it's harder for them when they start to like, join and learn and be a part of something"* (EP1). This indicates a lack of an organized onboarding procedures, where newcomers' learning depends on being available at the right time and having access to projects. A lack of such organized tasks creates tense situations, causing newcomers to feel stressed. *"We don't have a specific task for them to work on right away, and that can be stressful for some"* (EP3). Moreover, in the lack of a common understanding of onboarding approaches across teams, each developer follows their own personal onboarding approach. *"Everyone has their own sort of structure"* (AP11). Although a default structure theoretically existed, it was not consistently applied, and employees loosely followed it. *"Tries to follow the default one, and just that it was sort of cluttered"* (AP11).

In case A, team contributes to setting up the environment early and allocating sprint time for onboarding, thereby avoiding overloading sprints. However, they lacked a structured onboarding process where time was not deliberately protected for the onboarding process, despite their sprint commitments demanding time. *"We haven't really had a good plan, and we didn't have time to actually create an onboarding process"* (AP10). Although the teams had good intentions, they lacked a proper strategy. Teams acknowledge their need for change to accomplish a dedicated onboarding approach. Moreover, providing adequate knowledge of the most frequently used applications enables individuals to become familiar and adapt effectively. However, the lack of an organized knowledge-sharing approach prevents the intended knowledge from being effectively conveyed. *"I think there should be a better process for not giving them knowledge about every application because I have myself felt that I haven't gotten that knowledge from the beginning"* (AP11).

Although newcomers are given time to learn the system, their understanding remains limited. They lack clarity regarding prioritizations and transparency in decision-making, indicating a need for a more organized onboarding approach. *"I have only been six months at S, but for me, it's not clear how everything is being prioritized when it comes to planning"* (SP8). Environment setup remains an unresolved challenge and is acknowledged as an overhead which requires improvements. Individuals learn and integrate differently, with differences in their onboarding needs. The onboarding process should be flexible enough to accommodate these differences concerning newcomers' individual needs. *"Flexible onboarding planning, the environment setup is an overhead that could be optimized, but it takes time. Some people need a good foundation and ramp up very fast"* (TP12).

Moreover, working without the supervision of a mentor is challenging for newcomers. *"He has been working by himself"* (SP3). While being involved in meetings is beneficial for team cohesion, a lack of clear direction and motivation hinders effective newcomer integration.*"He has a tough journey"* (SP3). Self-directed onboarding processes place excessive responsibility on newcomers. When integration into the system is treated as the individual's responsibility, with limited structured support, it may result in stress and reduced confidence. Getting to know and work with multiple teams requires timely learning. *"Responsibility sometimes. Ask a colleague, how can I join you on this? Can I join you on this and just be a part of it?"* (EP1) and *"It's not just one person they are working with, but need to know those teams, and they are rotating"* (SP3).

Recruitment difficulties: Recruiting skilled personnel with required competence is a significant challenge for software companies. Most of skilled personnel are hired from a diverse culture background. This brings a constraint from business-related processes, rather than technical skills. *"You can't recruit good people. It's hard. It's too long from X.*

It's too long from Y. It's hard to find the right competence, and they have struggled with recruiting people for ideas and for my product" (AP8). Moreover, a noticeable knowledge gap exists, as candidates have computing backgrounds but lack domain expertise, making it difficult to find recruits with combined skills. *"I have noticed that it's very difficult to find such persons who have both [domain knowledge and programming], these two skills"* (TP4) and *"It is not programming, and it is the business requirements, and then we have a scale-out issue instead"* (AP8).

Attrition over the past few years further indicates that recruiting and retaining the required competence has been a persistent challenge. Whereas, replacing a "head" does not immediately replace the knowledge, contextual understanding and team commitments. *"We hired new people, but you can't replace heads, but you can replace experience in a certain project"* (AP3). Additionally, attrition of long-standing employees makes it more difficult to retain critical competencies in the short time. *"I would say our largest problem is that we lose people and replace them with newcomers"* (TP6). and *"For a couple of years, our main challenge has been knowledge about the systems because the architecture is so vast and, at the same time getting new persons up to speed on the project"* (AP3).

Recruitment should be aligned with the team's requirements. Focusing solely on hiring to achieve organizational goals can create challenges if teams do not acquire the appropriate competence and experience required to support their work. *"They are always trying to get new people from the right, directly from the university, but it's always the people that have studied, like computer engineering and stuff, and maybe that's not enough"* (TP3).

Team-facing onboarding challenges: Team faces a challenge regarding the availability of experienced team members. Although there are many new team members, attrition causes low availability of experienced team members. This type of structure reduces their time for questioning, and the team requests expertise in order to better establish learning opportunities. *"We have quite many young people in my team or new people* [...] *Have a little more time to ask questions or something more personal to take list problems with"* (EP1).

Similarly, team struggles with frequent onboarding of newcomers, which disrupts team stability, making it difficult to proceed. *"It's only been half a year now, like in a couple of months. I would say that the project is suffering from too much changes in people and in teams setups"* (TP5). In addition, case T encounters challenges in allocating team members to new customer projects, because of the limited experience of the newly hired employees. *"It is not possible to put these inexperienced people right into the program of a new customer. So I will lose some of my best people"* (TP1). Newcomers' pace of integration into the system affects the team's effectiveness in pursuing novel solutions. Team relies on the knowledge of experienced team members drawn from previous projects. *"It's always like a newcomer coming instead. That is, of course, something that at least for a while, slows down your team"* (TP6).

Rapid organizational growth brings onboarding challenges. Senior consultants in case E are responsible for training newcomers in their roles while still facilitating rapid organizational demands. This type of situation creates stressful conditions, even for experienced employees. *"This puts a lot of responsibility on the senior consultants to train the new ones and also do their job"* (EP3). Moreover, when a new employee with little experience of the functionalities and the product joins a team, supporting and training the newcomer requires additional resources from the existing team. *"The existing team needs to take care of that new person and train him or her, which will cost effort and time and energy"* (TP1).

4.3.2. Individual challenges

Cognitive overload: The sheer volume of information presented in tightly scheduled introductory meetings creates tense situations for newcomers, resulting in mental fatigue and minimal reflection. *"We had two and a half days full of onboarding classes or introduction meetings where we got to learn everything. It was very stressful, a very tight schedule"* (EP2), *"I thought maybe it was quite hard because we have a specific list of things that we can do the first couple of weeks or several weeks"* (TP3) and *"But it's pretty much information overflow"* (AP2). Newcomers anticipate a smoother transition from onboarding training to their actual job tasks. *"From those tasks to doing the real job or some real task, maybe I should have a smaller step in between that and then"* (TP3).

For newcomers, it is challenging at first to maintain a working pace comparable to their peers. Over time, newcomers are introduced into a network of teams and artifacts that they must continuously engage with to accomplish their work. In a LSA project, this naturally develops work pressure for newcomers. *"I feel more pressure, because everybody in our team is working very hard. I try to catch up as a new team member, and there are a lot of things need to learn. So maybe have some pressure"* (TP13) and *"It's challenging for me because I had my own tasks to finish, and I think it's a priority"* (TP14). Ensuring successful onboarding is a challenge in LSA because it involves multiple teams, scaling of agile practices, and integrated software development. *"Onboarding is a big issue that we have to work on and maintain a nice spirit even as the company grows bigger"* (EP3).

As newcomers enter working life, they gradually learn to manage tasks and navigate through system complexities. However, frequent switching between different tasks, projects or meetings is mentally demanding and challenging for newcomers to manage. *"I've only been in the work life for one year, I'm getting used to a lot of things and the context switching can be a bit difficult"* (TP9).

Barriers to psychological safety: In case A, newcomers are particularly observed by experienced team members who report their progression to the management. Though this is done to inform newcomers' continued presence within the team, this can consequently create a stressful atmosphere for them. *"I give feedback to the management, on how things are going, whether we can keep this person in, or the person should be replaced"* (AP1). Additionally, newcomers are less able to advocate for themselves on questions or decisions compared to experienced team members. *"I [a senior employee] have no problem telling the manager or the customer that I want to do this"*. It is not that they are incapable, but rather they need time to build confidence. The confidence that experienced employers have built up over years of work is difficult for recent graduates to achieve. *"A lot of new people are directly from school, and it's not easy for them to do the same"* (AP3).

Team members tend to notice that newcomers rarely engage with team discussions and share their perspectives. This lower level of engagement creates confusion regarding the tasks these individuals are performing, as the team remains unaware of their conditions. *"I can notice when there are new ones or people with lower energy personalities. They don't want to ask for help"* (AP6). In a highly technical or specialized environment, newcomers remain silent because they lack the knowledge needed to raise queries. *"It's hard to know when you don't know what they're not asking"* (TP8). This substantial knowledge gap creates uncertainty. Even in a psychologically safe environment, newcomers struggle to grasp the context without their confidence in technical understanding. *"Once you've become used to the forums and feel a bit more self-sure of yourself, then I think most people would dare to speak up"* (TP8). In addition, newcomers recognize that their co-workers are more

experienced and feel comparatively inexperienced. This makes it difficult for them to follow the discussions, and scared to admit their confusion. This creates tension between their rational thinking and emotional states, making them emotionally insecure even though the work environment is safe. *"No one is going to be angry or something, because I don't understand, I'm just scared sometimes"* (TP3).

Though team needs to maintain a balanced environment in which everyone feels comfortable in asking questions, this can be challenging. Because the information should not be presented in a way that makes it seem acceptable, nor should it make newcomers feel intimidated. *"We tend to have a friendly atmosphere, but it can sometimes be a bit blunt, no need to sugarcoat things, but no need to make them inadvertently aggressive either"* (TP9). Nevertheless, case T teams encourage newcomers to ask questions wherever they encounter issues. Although newcomers face difficulties and make mistakes within a complex system, this leads them to improve and feel more comfortable in a psychologically safe environment. *"No question is too silly to be asked and we always try to say yes"* (TP6).

Knowledge gap and retention: Although case A comprise strong expertise in certain areas of technology, it has identified domains in which their competencies are lacking. These gaps should be considered during the onboarding process to ensure that newcomers acquire the necessary knowledge and help reduce existing knowledge deficiencies. *"[...], There is one area where the competence is non-existent and there is no one out there that knows it either"* (AP2) and *"We do have areas in which we are severely lacking competency and we definitely have a people debt in that sense"* (AP3). Therefore, team strives to hire newcomers into specialized areas to make the experience more engaging. *"I'm sure that's very important. It's a very specific area to get newcomers into team"* (TP5). In addition, experienced team members expect new starters to familiarize themselves with the ongoing project works. *"More fresh new knowledge from school on how you're supposed to code and but does not have this [experience]. I mean, they will gain it"* (TP5). For instance, an experienced developer who understands the entire code base will feel satisfied when they remove redundant code. In contrast, a newcomer might feel satisfied when they add more code. Therefore, newcomers are introduced to a code base that requires careful handling. *"When you are to go into a test or a piece of code and improve something, make a piece of code more readable"* (TP8).

Apart from this, integrating highly specialized expertise into a complex system is challenging. *"It's challenging when someone has 15 years of experience in a bit piece of software and no expertise whatsoever in any kind of other software"*. To utilize such resources, the organization needs to invest further in their development. *"To utilize that kind of resource, you either need to educate them, and he needs to be open to being educated"* (AP7).

4.3.3. Contextual challenges

System complexities: The complexities in the system present challenges in certain applications where even experienced team members lack prior exposure. This hinders the learning process of newcomers and slows their integration into the team. *"Sometimes we receive an error in an application that we haven't even heard about, new people come into the project. It's more complicated than it sounds"* (AP1). Working within a large, complex system does not lead to a full understanding without sufficient exposure to its components and processes. *"I am still pretty new, and it's an extensive system with many applications. So it could just be that I haven't gotten a clear grasp of it yet"* (AP6). These systems include numerous applications and unclear naming conventions that newcomers are not sufficiently

familiar with. This creates a lack of clarity in accountability when a problem occurs. These types of situations are challenging for a newcomer to encounter. *"When problems occur, it could be a little hard to see where the responsibility lies"* (AP6).

The complexity of the domain poses substantial challenges for newcomers. Although their onboarding agenda includes initiatives aimed at improving domain knowledge, newcomers perceive that their understanding is insufficient. This calls for input from experienced team members. *"I think there might be some people who really grasp the code and the whole architecture of the system to the level where the newbies and some junior people might need some more input from those people. It is a problem, being the domain is so complex"* (AP2). Additionally, in case T, newcomers are expected to contribute as regular team members once their onboarding process is completed. Nevertheless, team proposes involving experienced employees with strong domain knowledge to train newcomers and junior employees. However, given the system complexities and responsibilities of senior employees, implementing such an approach is difficult in practice. *"We have some people who have 20 years of experience in this domain. How can we use them to train people that have one or two years of experience? But I think it's challenging to make it"* (TP1).

Beyond that, system complexity and barriers to domain knowledge acquisition create steep learning curves for newcomers. Even though newcomers receive introductions to tasks, they require adequate time to adapt to complex systems. *"I have not done anything code-wise during this time. Take time for newcomers to adapt to complex systems"* (TP3) and *"It will be very difficult because the procedures and the way of working is entirely different in the systems team and to get familiarized with all these things"* (TP4). LSA organizations comprise many teams whose members collaborate with each other to perform their tasks. Newcomers require adequate time to familiar with different project teams, as they are primarily assigned to a single team. *"It takes time, to get into the project because first you focus on only your own team on the tasks there"* (TP5) and *"In my experience, which is very little, like five months, I have never had something that was needed from another team or manager"* (TP15).

Remote onboarding: In a hybrid work setting, team members can split their time between remote work and on-site work. It is challenging for remote newcomers to integrate into cross-team collaborations and synchronised planning. *"It would probably have been easier to have something more formalised, especially since we're remote, or at least I am remote. Some of the other guys are sitting together in X, but I am in Y, and it is all between teams"* (AP5).

A remote work setting creates gaps in availability and visibility among team members. Communication through MS Teams platform can reduce the opportunities for in-person social interactions. This absence of frequent interpersonal contacts creates psychological distance among the team members. *"If we don't hear from each other on MS Teams, maybe it feels a bit isolated"* (AP1). As a result, newcomers face difficulties in building strong relationships with their co-workers and are unaware of whom to reach out to, particularly in the absence of a structured communication process. and *"There's also some informal information going on there that you miss out on"* and *"It is maybe not that obvious how to contact other teams and when and who in the other groups"* (AP5).

Cultural sensitivity: Welcoming newcomers from diverse cultural backgrounds requires careful consideration because working cultures and norms differ from region to region. Without careful individual assessment and planning, newcomers experience isolation and find it difficult to integrate into their teams. *"I faced some difficulties during the initial days when I started here. I have heard a lot from others that they are not providing enough support*

during onboarding" and *"It is mainly because of the Swedish culture that they think that the other guy knows more than I do"* (TP4). Some working cultures practice self-learning onboarding approaches, whereas others follow well-defined onboarding procedures, providing a sequence of instructions. New employees from diverse cultural backgrounds embrace these changes differently. *"I know that you have to learn it yourself and it depends on. In Sweden, there is no such formalities. But in some other countries"* (TP4).

Different cultural backgrounds associate with distinct working styles. Newcomers from other countries who integrate into a different cultural environment observe variations in working pace, time management, and team practices, which highlight need for careful team discussions and task management. *"Some people, they were against calling for a meeting on Friday afternoon. There are a lot of problems"* (TP4).

Employees from diverse cultural backgrounds are accustomed to different types of organizational structures. Some have experience in hierarchical systems, where authority flows from top down through clear chains of command. *"If they are managers, Indians, trainees, their way of working is different. It's purely hierarchical"*. Others are more familiar with flat structures, characterized by more autonomy and collective decision-making. The statement *"They always force us to do something"* reflects a cultural mismatch resulting in tension arising from differences between top-down management and flat organizational structure. New hires are particularly affected by such structural constraints. *"They are employing more Swedish, Chinese and Indian managers. Because Swedes cannot force others"* (TP4). Moreover, in a flat organizational culture, a new employee being productive is not only possible but also acknowledged and accepted. *"A junior employee will be more productive than a senior employee"*. In such an environment, performance and output are valued more than seniority. *"I'll get senior to junior. I haven't seen anything like that"* (TP4).

A team composed of members from diverse cultural backgrounds also face challenges in both decision-making and execution. *"They are not used to Sweden is quite special when it comes to like decision making"* (TP6). Although newcomers are proficient in English, differences in their decision-making styles still create misunderstandings. In Sweden, where orgnizations are embedded in a flat structure, employees make their own decisions once they feel confident in their competence, while seeking clarifications when necessary. However, newcomers from more hierarchical backgrounds are used to following instructions from their managers. *"They are not at all used to that. They are more used to managers actually telling them what to do"* (TP6).

Furthermore, long-term exposure to top-down hierarchical structures leads junior employees to expect those in higher positions to take the lead in discussions and apparently seek only their specific support. As reflected in the statements *"It's always like it's directed to the person higher up in their hierarchy"* (TP11) and *"I'm not a native English speaker, and I mean in the education in X, you must follow the teacher, right? So that's the challenge for me in some meetings"* (TP14), communication tends to flow upward rather than being shared collectively. This creates challenges within teams, as not everyone feels comfortable expressing their insights during meetings. Consequently, this hinders collective decision-making, which is essential for a collaborative team environment. *"There could be some like hierarchy issues, like people aren't really speaking up because they expect the person higher up in the hierarchy to speak up"* and *"We're not used to speak out or myself or even didn't think about it. But we have an opinion, so that's the challenge"* (TP11).

Agile-related issues: Newcomers perceive a contradiction between the intended agile mindset and the formal, task-intensive reality of daily agile practices. These process-related

complexities can be overwhelming. *"There has to be a place for innovation and creative thinking and new ideas as well. But maybe it contrasts against the very formal reality of the Scrum Board, which is a little bit crazy because, if you're looking at the agile mindset and what that represents, it should not be that kind of a hassle"* (AP3). While transparency in agile supports task accountability and coordination, it also exposes workload saturation and limits the opportunities for innovation. *"But if it's transparent that we should do these things, it shows that we don't have time to do other stuff"* (AP3).

Integrating into a physically co-located team with a limited development focus and immediate support is straightforward for newcomers. *"I would say, easier for a newcomer coming in because then it was a little bit more limited scope"* (TP6). However, newcomers face significant complexity with LSA projects, involving multiple interconnected systems with broader functionalities. "Higher threshold coming in now" indicates that newcomers need comprehensive knowledge for effective system integration, which is challenging. Agile development also involves trade-offs that are not always acknowledged *"When you have agile teams that are covering so much more functionality"* (TP6).

5. Discussion

Our findings expand understanding of newcomer onboarding in LSA projects. First, while Bauer's organizational socialization model [6] provides a relevant and broadly applicable framework, its direct application to LSA contexts is insufficient without augmentation by agile-specific practices that are structurally embedded in how LSA teams work. Second, the complexity of LSA systems generates a distinct onboarding dimension (i.e., technical integration) that existing onboarding frameworks do not recognize and that social integration practices alone cannot address. Third, the challenges that characterize LSA onboarding are not merely experiential difficulties for individual newcomers. They are organizational compromises that accumulate as forms of socio-technical debt. Together, these three findings suggest that LSA onboarding must be treated not as a peripheral HR activity but as a core risk management function.

Successful LSA onboarding is not a discrete event but begins conceptually from the recruitment phase, requiring a structured and coordinated integration plan. This active approach necessitates connecting newcomers with relevant stakeholders early to ensure they fit LSA environment, aligning well with the initial phases of Bauer's model [6].

Formal orientation and support are crucial in LSA to manage complexity; these practices include mandatory introductory meetings and detailed system overview presentations. These are critical for grasping LSA system integrations. We identified different orientation techniques, including introductory meetings, system overview presentations, and team-led grooming sessions, which are either socially or technically oriented. Using onboarding templates or checklists standardizes timelines and access procedures. Providing such templates that specify the required steps is considered a best practice in onboarding [6]. Companies consider institutionalized (formal) onboarding to be a successful approach [7, 10] and recommend that "most onboarding activities should be required and offered formally" [35]. Reference to the system documentation provides an efficient and coherent way of familiarizing oneself with the tools and technologies, the system domain, feature dependencies and security measures, which is particularly important in LSA development. Moreover, our case companies employed all or some of the techniques, such as providing

access to tools, provisioning accounts, and assisting with system setup, to facilitate newcomer integration into the workflow.

Training is often a hybrid approach, combining formal learning (materials, online courses) with individualized practical methods like task-based and case-based learning. Our results support the finding of [6, 7] that training can be formal or informal. The hands-on methods are particularly supportive in LSA as they help newcomers gradually navigate system architecture, complex dependencies, and essential domain knowledge. This is essential for minimizing subsequent rework caused by incomplete understanding. By conducting code reviews individuals familiarize themselves with the project's codebase, standards and conventions, while promoting knowledge transfer, team awareness and transparency, particularly among junior developers [36].

Furthermore, coaching and support effectively leverage mentors, peers, scrum masters and product owners, demonstrating a necessary institutionalized onboarding approach characterized as formal and sequential. All four of our case companies position mentors to facilitate a smooth technical and social transition for newcomers. Newcomers who receive mentoring-based education report being active earlier than those who undergo conventional onboarding [24]. Mentoring is claimed to be a central onboarding approach facilitating newcomers' integration into their team, technology, agile ways of working, and occurs informally through peer interactions. Mentoring helps newcomers enhance their skills by providing someone to ask questions [7,37]. Pair programming provides informal peer support by integrating newcomers into daily collaborative work alongside experienced colleagues [7, 37]. The team being approachable to questions from new employees is considered an effective onboarding practice, as it allows newcomers to seek help from any team member [37]. Additionally, seeking ongoing feedback is a recognized onboarding practice that helps newcomers to actively gather information to improve their performance and adapt to teams [6, 19]. The incremental and iterative nature of agile [38] treats feedback as a continuous and essential practice, both within teams and across other stakeholders. In LSA settings, however, feedback extends beyond a team practice to serve as a critical coordination and transparency mechanism [1], becoming increasingly challenging to maintain as the number of teams and organizational layers increases.

Our findings align with adjustment dimensions [6] which include self-efficacy, role clarity, social integration, and knowledge of culture. Moreover, we propose "technical integration" as an adjustment for LSA, extending Bauer's model by emphasizing the importance of smooth technical integration. Self-efficacy, reflecting an individual's belief in their ability to perform their responsibilities effectively, is linked to positive workplace outcomes, including organizational commitment, job satisfaction and reduced staff turnover [19]. This is particularly true for newcomers who feel capable of mastering their work, as they tend to adapt and integrate successfully. A higher level of role clarity is indicated when new employees have a clear understanding of their roles and responsibilities. In two of the cases we studied, teams expect newcomers not only to understand the cultural and technical expectations of their job roles but also to engage actively and contribute meaningfully. Role clarity is identified as one of the most important predictors of job satisfaction and organizational commitment [19]. It is also important that newcomers feel socially comfortable and are accepted by their peers and supervisors [19]. Case companies organize various social integration activities to help newcomers feel settled and psychologically safe within their work environments. A psychologically safe environment allows the belief to emerge that the context is safe for interpersonal risk-taking, in which speaking up with ideas, questions, concerns, or mistakes is welcomed and valued [39–41]. Whereas "fear"

undermines cognitive diversity, including analytical skills, creativity and problem solving and further reduces learning behaviours such as asking questions, seeking help, and openly discussing mistakes [39].

Beyond emphasizing people and culture-related adjustments, the complexity of the LSA context calls for "technical integration" as a new adjustment dimension. Case companies facilitate technical integration through various means, such as technical meetings and tech know-how sessions. Providing an opportunity to learn technologies while enjoying a coffee together with peers seems to be beneficial. Specifically, "tech fika" provides an opportunity to learn and raise questions regarding technical and domain knowledge in a relaxed, discussion-friendly setting. These technical meetings become important when the team has more new technical developers, as they can spend more time on learning and understanding system complexities [9].

Notably, onboarding functions are fundamentally applicable to LSA projects when augmented with specific agile practices. Daily standup meetings are a short, time-boxed meeting widely used by agile teams and are particularly useful for newcomers [42]. Daily standup meetings improve communication and coordination within the project, providing an important opportunity for the agile team to share information and increase team awareness. Junior developers tend to view daily standup positively, as they find these meetings valuable for receiving information and getting support [42]. Majority of project teams use pair-programming for a defined period as an initial approach to technical integration. Pair programming provides useful support for newcomers to familiarise themselves with the code base [7]. However, the relative benefits of pair-programming compared to solo programming are not precise, and it depends on the programmer's expertise, complexity of system, and tasks performance [43].

Innovation sprints provide newcomers with hands-on experience in developing innovative thinking. No work is planned during the Planning innovation (PI) iteration in PI Planning. Instead, this time is set aside, allowing teams to be creative [44]. Retrospectives are one of the other standard practices developed for agile teams, in which they reflect and improve their work process [7,45]. Retrospectives involve discussing team obstacles, sharing feelings, action points, root causes and future actions [45]. Retrospectives provide a platform for newcomers to exchange feedback and receive guidance on their progression, opportunities and weaknesses. Agile training is included in introduction sessions, where newcomers are encouraged to participate and tailored to their specific roles. Agile training helps to bridge the gap between traditional and agile practices. It allows the agile teams to absorb and retain the agile values and principles [46].

The challenges identified in LSA onboarding expose the projects to various forms of socio-technical debt (e.g., people, social, process, organizational, knowledge, documentation and code). (see Figures 1, 2 and 3 for the organizational, individual and contextual challenges and their association with socio-technical debt accumulation). These associations between onboarding challenges and socio-technical debt are interpretive propositions, inferred from the qualitative data.

The lack of an organized onboarding process slows the integration of newcomers. In particular, limited access to people and information is associated with social debt, requiring newcomers to seek information and establish relevant connections. Consequently, integrating newcomers into projects solely on newcomers' and project availability slows their growth and performance, interpreted as potentially leading to people debt. The presence of people debt is directly related to productivity and employee satisfaction [18]. Furthermore, failure to allocate dedicated time for onboarding within working sprints results in inconsistent

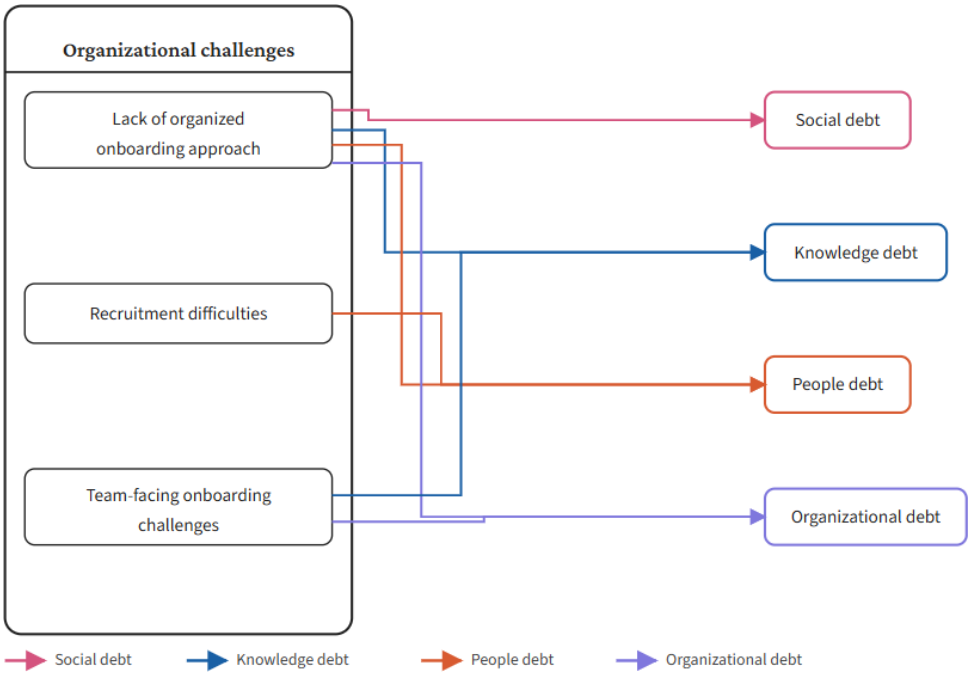


Figure 1. Organizational challenges associated with socio-technical debt

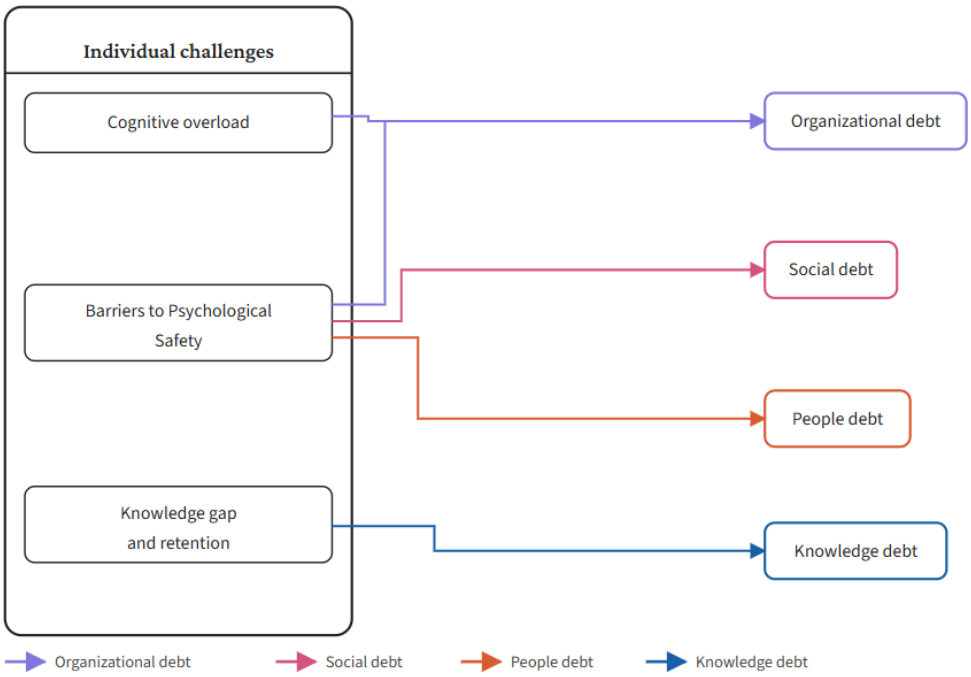


Figure 2. Individual challenges associated with socio-technical debt

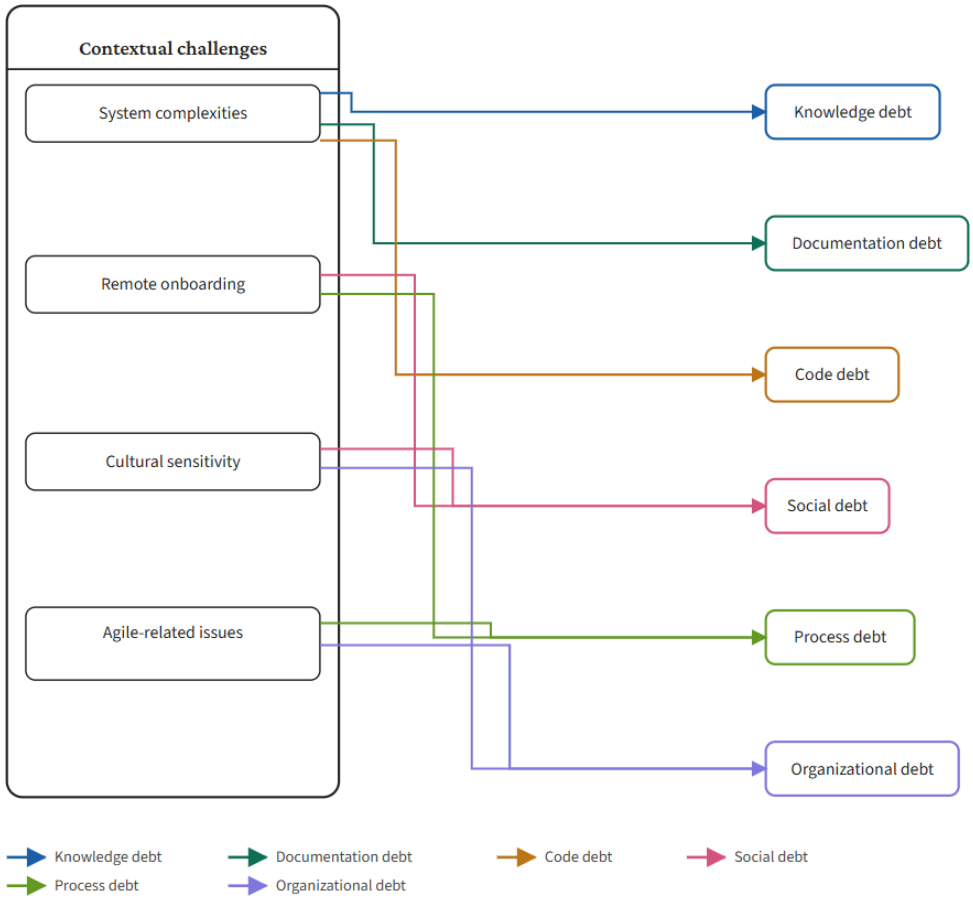


Figure 3. Contextual challenges associated with socio-technical debt

onboarding practices and stressful situations for newcomers, particularly associated with the emergence of organizational debt. Organizational debt represents an accumulated burden that prevents people from achieving exceptional results [15].

The absence of an organized knowledge-sharing approach for newcomers is interpreted as potentially contributing to knowledge debt [47], whereas insufficient knowledge distribution represents another dimension of technical debt [47]. Allocating time for system learning alone is not merely sufficient. Newcomers also require adequate knowledge and skills in planning, prioritisation and decision-making; otherwise, knowledge debt may be incurred. Insufficient time allocated to developing support tools and processes tailored to individual needs and learning pace is interpreted as potentially contributing to organizational debt, reflecting the "things that should've been done to ensure health efficiency, but weren't [15]." A newcomer who does not have the opportunity to be led by someone towards other teams and their structures finds it difficult to perform and get along with team members, associates with organizational debt [15]. Assigning challenging tasks during the early stages of onboarding has been linked to lower confidence and anxiety among newcomers, as reported in a study of onboarding in Microsoft software teams [9]. Similarly, finding the right piece of code to work with and a task to start with has been noted as a challenge in open source software projects [25]. These challenges become particularly important in LSA environment, where domain complexity makes the selection of suitable onboarding tasks even more difficult.

Recruitment is challenging due to inherent knowledge gaps and complex business requirements. In particular, recruiting employees with a combination of IT and business knowledge, especially domain experts can help resolve the root causes of the people-related issues [46]. When recruitment and onboarding are poorly executed, it is associated with people debt [18]. People debt stems from insufficient expertise or lack of effective onboarding support [18].

In addition, attrition of experienced employees, coupled with the simultaneous onboarding of newcomers, changes the teams' structure and task management. Insufficient learning opportunities thus created are interpreted as potentially contributing to the accumulation of knowledge debt [47]. Frequent team switching, challenges of handling new customer projects with inexperienced employees, and training of newcomers in a dynamic environment require significant effort and time. In LSA environments, these challenges are often exacerbated by the coordination and cross-team dependencies. Consequently, these challenges that teams encounter during the onboarding process are associated with organizational debt, which reduces organizational effectiveness and slows adaptation [15]. To address these challenges, organizations could adjust team composition by including more experienced members and fewer newcomers, creating a balance that enhances the team's ability to continuously improve [48].

The issue of cognitive overload, resulting from tightly scheduled introductory sessions and fast-paced development, creates significant mental fatigue and pressure on newcomers. Cognitive overload occurs at a point where the amount of information or the ability to process information exceeds an individual's capacity [49]. Task completion urgency builds pressure, reducing engagement capacity, flexibility and innovation [15]. Characteristics present in contributing files such as branching workflows, hidden dependencies, scattered instructions, and project conventions increase the cognitive burden for newcomers in open source software projects [25, 50], particularly amplifying the impact on systems developed by multiple teams. This highlights the trade-off between culture and management practices interpreted as potentially contributing to organizational debt. "Organizational debt is all the people/culture compromises made to "just get it done" in the early stages of a startup [15]. Introducing a gradual stepwise transition would help newcomers' adaptation to the real tasks.

We identified both individual and structural barriers to psychological safety. Human instincts can prevent people from working productively in a dynamic environment [39]. When individuals experience high psychological safety and high performance standards, they are more likely to operate in a learning zone [39, 51]. Cultivating psychological safety during onboarding can be supported through clear expectations, encouraging open dialogue, and providing continuous feedback within the first 90 days [52]. This is particularly important in the LSA environment, where newcomers navigate across multiple teams. Newcomers who rarely participate in team discussions exhibit a low level of engagement and awareness, which is interpreted as potentially contributing to social debt [16]. Fear of asking questions and lower confidence arising from experience gaps are associated with people debt [18]. Newcomers face organizational-level challenges that can reduce their confidence and perceived comfort within the workplace. To prevent the accumulation of potential organizational debt [15], teams need to invest substantial efforts in creating an environment where newcomers can thrive, ensuring that issues are neither sugar-coated nor expressed aggressively. Moreover, the inherent complexities of legacy systems make it challenging to achieve a comprehensive understanding of the system is potentially interpreted as contributing to knowledge debt [47].

Learning a large legacy codebase makes onboarding challenging for newcomers, requiring a significantly longer period of mentoring and coaching compared to a typical software project. Furthermore, code written by former team members complicates knowledge transfer [53]. These challenges are particularly pronounced in a large-scale development environment. Moreover, inadequate mentoring on the system and workarounds subsequently associate with code debt [54]. The presence of a large, extensive system with many applications and unclear naming conventions suggests that standards are needed to prevent the accumulation of documentation debt. Although companies maintain an intranet, wikis and production documents, these resources could be outdated, incomplete and insufficiently detailed. Consequently, onboarding becomes challenging and requires newcomers to engage in a continuous interaction with mentors to acquire knowledge [53]. Accessing a complex domain requires expert support. Failure to embed domain knowledge through documentation and knowledge-sharing sessions is interpreted as potentially leading to knowledge debt and documentation debt [47].

Similarly, difficulties with remote onboarding complicate newcomers' seamless integration into cross-team collaborations and planning activities. Developing shared organizational values, skills, attitudes, abilities, behaviours, and relationships becomes challenging in a virtual onboarding setting [55]. Limited informal communications between newcomers and their reduced visibility within the team result in newcomers being excluded from discussions [56]. Such challenges are prevalent in LSA environments, where coordination across multiple teams exacerbates communication and availability gaps. When the needs and expectations of these remotely working newcomers are unmet, the resulting lack of effective social and process integration constitutes a compromise in social processes [7, 15]. These results may potentially contribute to the accumulation of social debt and exacerbate team isolation. Process debt arises when a discrepancy exists between a job role and the responsibilities defined in the organizational structure [17].

The issues related to cultural sensitivity are interpreted as potentially contributing to social and organizational debt. A failure to foster an inclusive and collaborative learning culture undermines the project network and requires future organizational effort to rectify [15, 16]. For instance, cultural misunderstanding arising from a failure to respect and adapt to different working styles, organizational structures, decision-making strategies and communication styles associate with potential exacerbation of social and organizational debt.

Lastly, issues related to agile development are associated with process and organizational debt [15]. Process complexities across both ideal and real-world agile practices, coupled with insufficient time for innovations, pose challenges for newcomers. These challenges are further reflected in difficulties in conveying agile principles, as established team members hold high expectations of newcomers and struggle to communicate them effectively [48]. A workflow refers to a series of repeated activities carried out to complete a particular task while a process consists of repeatable activities performed to achieve organizational goals [18]. "Inadequate adaptations" within the organizations leads to ineffectiveness [15].

We offer insights for practitioners to improve LSA onboarding by integrating agile-related practices and adopting a practical training approach that systematically combines formal, task-based, and case-based learning. In particular, we recommend incorporating technical integration to facilitate a psychologically safe adoption of technical systems and to address organizational compromises that extend beyond individual experiences. Our goal is to guide a process that is structured, flexible and context-aware, helping to reduce the accumulation of socio-technical debt.

Threats to validity and limitations

To evaluate our study limitations, we apply validity criteria defined by Runeson and Höst [28].
Construct validity refers to the extent to which our study measures what it is intended to measure. This is ensured by forming an interview guide to ensure that the data produced through interviews answers our research questions. The researcher's perceptions, along with the interview guide, were discussed with company contacts to potentially build meaningful insights and prevent any misinterpretations. Throughout the study, debriefing sessions supported researchers in avoiding potential bias in data collection and interpretation. The interviews were recorded so that they could be transcribed later and all researchers could have the same understanding of what was said. Spending sufficient time with the cases and engaging in prolonged interaction helped researchers to build trustworthy relationships with interviewees, reducing the likelihood of them providing any biased information.
Internal validity concerns the credibility and robustness of our findings. We conducted a multiple case study using interview data collected in different organizational contexts. We selected participants based on their differing roles in order to capture diverse perspectives. We sent the transcripts back to the interviewees for member checking. Additionally, we produced mind maps during the qualitative analysis stages to ensure the accurate interpretation of the interviewees' perspectives. The researchers collectively reflected on initial coding and assigned it to themes, potentially reducing the bias of the individual researchers when analysing the data. In relation to data adequacy, data saturation was not formally assessed as a distinct step within the methodology. However, with 41 interviews conducted across four different companies, data analysis continued until no substantially new themes were emerging in the later part of analysis. Nevertheless, we acknowledge this as a methodological limitation. In the Methods section, we provided a detailed description of our findings, supported by quotations from the data, and maintained audit trails comprising the interview methods, transcripts and data collection.
External validity concerns the generalizability of findings beyond the studied cases. Our four case companies represent a range of organizational sizes and domains, which strengthens transferability. Regarding data collection, participants' experiences reflect their overall professional experience and not their tenure at the specific organization studied. The participants also included a range of career stages across all four companies. Furthermore, most of our participants are team members holding a variety of roles, including developers, testers, designers and architects. However, we acknowledge that the onboarding experience encompasses multiple perspectives across working levels, including mentors and management. Beyond that, all four of our companies are located in the same geographic region, which introduces cultural homogeneity and limits the applicability of findings to organizational cultures. Additionally, as a qualitative case study, our work does not permit causal claims. The associations we draw between onboarding practices, challenges, and socio-technical debt accumulation are interpretive and theoretically grounded, but not empirically tested in a quantitative sense. Our findings should be treated as propositions that needs further validation rather than established generalizations.
Reliability concerns the repeatability of research process. To uphold reliability, we used the thematic mapping approach by Braun and Clarke [33], showing how we coded collected data. Our methodology includes detailed reporting on case context, data collection techniques and data analysis, ensuring consistency. The interview guide was discussed in detail, and any unclear questions were avoided to ensure that interviews were clear.

6. Recommendations

We propose five key recommendations for practitioners to improve LSA onboarding. The goal is to provide guidance toward a process that is flexible, context-aware, and structured to mitigate the accumulation of socio-technical debt.

R1: Design competency-based onboarding plans rather than time-based ones: Across cases, a recurring challenge was the mismatch between fixed onboarding schedule and variable prior experience of newcomers. Rather than prescribing a uniform, fixed-schedule process, onboarding duration and content should be customizable based on newcomer's existing domain knowledge and technical skills. This approach ensures that experienced hires are not over-burdened by redundant training, while those requiring more foundational support are given a stronger, customized learning path. This will support long-term retention and reduces risk of incurring people debt [18] from misaligned roles.

R2: Integrate agile ceremonies into onboarding from day one as structured learning mechanisms: Our findings show that agile ceremonies were among the most consistently reported and effective onboarding tools across all four cases. However, their use was often informal and dependent on team initiative rather than deliberate design. In Case E, standup meetings were explicitly used to monitor newcomer progress and surface blockers. In cases S, T, and A, pair programming provided technical scaffolding during critical early weeks. Where these practices were structured and newcomers reported faster integration; where they were ad hoc, newcomers were more likely to remain passive observers. This means practical training approach that systematically blends formal instruction with hands-on learning. This method should leverage essential Agile practices (e.g., pair programming, daily standup meetings) from the first day to provide immediate technical and social exposure. Furthermore, teams should deliberately employ task-based learning for simple, focused assignments alongside case-based learning for assignments that require newcomer to explore complex system architecture, multiple dependencies, and demanding business requirements. This deliberate progression from simple to complex tasks ensures gradual increase in cognitive load while building comprehensive system understanding. This results in reducing likelihood of committing errors that contribute to technical debt.

R3: Expand social integration beyond the immediate team through structured cross-team exposure: A distinctive challenge in LSA onboarding, observed across all cases, is that newcomers must integrate not into a single team but into a network of interdependent teams. Cases T and A shows that newcomers who were confined to their immediate squad during onboarding struggled with cross-team collaboration and had difficulty identifying right people to contact for system-related support. This produced social debt [15, 18] in the form of communication breakdowns and duplicated effort. Social integration requires structured effort. We recommend that onboarding plans in LSA contexts explicitly include structured cross-team exposure from early in the onboarding period. For example, newcomers should be spread out. Avoid grouping newcomers all together. Place them on existing, established teams. This maximizes their exposure to diverse organization units and helps them to gain peer knowledge. Robust support must be guaranteed which needs both internal and external stakeholders. Internal support can come from a dedicated mentor or scrum master. External support from the product owner is required to ensure clear guidance on business priorities. This is a common blind spot for newcomers and is noticeable in LSA context.

R4: Address gap between textbook agile and real-world LSA practice through explicit expectation setting: Across all four cases, newcomers consistently encountered

a gap between agile principles they had learned through training or prior experience and actual practices of their LSA teams. Fourthly, bridging the gap between textbook Agile values and actual Agile practices requires cognitive and cultural sense-making. For example, implementing a customised agile at the organizational level reduces the sense of being overwhelmed by newcomers due to process complexities, thereby enhancing understanding process orientation and expectation setting. In addition, a limited scope for innovation can demotivate newcomers. For instance, if newcomers expect creativity but encounter time pressure and a focus on delivery, they may feel less engaged. Therefore, realistic innovation opportunities should be facilitated. Transparency within agile reveals workload intensity and performance expectations, which can generate stress and fear of underperforming among newcomers. Therefore, newcomers require a gradual workload ramp-up and an acceptable learning curve to avoid overwhelming them during these critical periods.

R5: Restructure orientation to prevent cognitive overload through phased information delivery: Cognitive overload was one of the most frequently cited individual-level challenges across cases. Such overload stems from a high volume of initial information, tight scheduling, expectation to maintain peer-equivalent working pace, and frequent switching between onboarding tasks, meetings, and project work. In LSA contexts, this challenge is structurally amplified: the system complexity, number of teams, and breadth of domain knowledge required, which means that the information load placed on newcomers is inherently greater than in smaller projects. When unaddressed, cognitive overload generates organizational debt by eroding newcomers' capacity for reflection, reducing code quality, and increasing the likelihood of early disengagement [15, 49]. This is best addressed by restructuring initial information cascade into a phased and gradual approach to delivery. Orientation materials and introductory sessions should be broken down into smaller, digestible modules and intentionally spread across first few weeks, rather than compressed into first few days. By establishing a clearer, stepwise transition from abstract onboarding training to actual job tasks, organization reduces immediate pressure on newcomer. This allows adequate time for critical reflection, assimilation, and sustainable adaptation to true working pace of LSA environment.

7. Conclusions

This study investigated onboarding processes for newcomers in LSA projects and assessed Bauer's model applicability in this context. Through a qualitative multiple case study spanning four Swedish companies, we have established three principal contributions.

First, Bauer's model is broadly applicable to LSA onboarding but requires augmentation. Crucially, our findings reveal that the model must be augmented with specific agile-related practices and case-based learning practices to be effective. We identified system overview presentations, team-led grooming sessions, and use of system documentation as meaningful techniques for LSA onboarding. Any application of Bauer's model to LSA must incorporate these agile dimensions to be analytically complete. Second, technical integration constitutes a distinct onboarding dimension specific to LSA that is absent from existing onboarding frameworks. The scale, complexity, and multi-team architecture of LSA systems create a category of onboarding challenge (i.e., learning to navigate a large, interdependent, and continuously evolving technical environment) that social integration activities alone cannot address. The emergence of technical integration suggests that LSA teams have organically recognized this need. We propose that technical integration be formally recognized as

an extension of Bauer's model when applied to LSA context. Third, the most significant contribution of this work lies in identifying the LSA-specific onboarding challenges and establishing their link to various forms of socio-technical debt. We demonstrated how issues like cognitive overload from overly dense schedules, large-scale system complexities and difficulties of remote integration become compromises that hinder long-term team efficiency and cultural stability. Successful LSA onboarding is not merely transactional; it is an integrated process that begins with rigorous recruitment and requires a flexible approach to training and integration. This framing reposition onboarding from a peripheral HR concern to a core technical and organizational risk management activity. Poor onboarding is not merely costly in the short term; it creates accumulated deficits that must be repaid with interest as projects scale.

We provided five recommendations to offer practitioners a framework for restructuring LSA onboarding: competency-based planning, agile-integrated learning, structured cross-team socialization, explicit expectation setting around real-world agile practice, and phased orientation delivery. These recommendations will not only improve newcomer experience but to proactively prevent socio-technical debt that accrues when onboarding is treated as an afterthought.

Building upon our findings, we propose several future research directions for LSA environments. First, the associations between onboarding challenges and socio-technical debt require quantitative validation. Future studies should focus on developing and applying metrics to measure the accumulation of specific types of debt in projects that employ varying onboarding strategies. Second, Future research should adopt a longitudinal, multi-perspective approach to interview mentors and management in addition to the team members. Third, longitudinal studies are needed to trace how onboarding quality at individual level translates into team performance over time. Fourth, the effectiveness of task-based and case-based learning relative to other training approaches highlights controlled investigation. Finally, as LSA frameworks continue to evolve and hybrid work becomes the norm, comparative research is needed to explicitly analyse onboarding in different scaling frameworks (e.g., SAFe, LeSS) as well as study specific socio-technical debt implications of fully remote versus hybrid LSA teams.

CRediT authorship contribution statement

P. Herath: Writing original draft, review and editing, visualization, methodology, investigation, formal analysis, data curation, conceptualization.
M.O. Ahmad: Writing original draft, review and editing, visualization, supervision, project administration, methodology, investigation, formal analysis, data curation, conceptualization.
T. Gustavsson: writing original draft, review, validation, investigation, conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

This research was carried out within the NODLA project [grant no. 20200253] funded by Knowledge Foundation Sweden, Helge Ax: son Johnsons Stiftelse and Swedish Foundation for International Cooperation in Research and Higher Education (STINT) [IB2020-8720].

References

[1] H. Edison, X. Wang, and K. Conboy, "Comparing methods for large-scale agile software development: A systematic literature review," *IEEE Transactions on Software Engineering*, Vol. 48, No. 8, 2021, pp. 2709–2731.

[2] T. Dingsøyr and N.B. Moe, "Research challenges in large-scale agile software development," *ACM SIGSOFT Software Engineering Notes*, Vol. 38, No. 5, 2013, pp. 38–39.

[3] M. Kowalczyk, B. Marcinkowski, and A. Przybyłek, "Scaled agile framework. Dealing with software process-related challenges of a financial group with the action research approach," *Journal of Software: Evolution and Process*, Vol. 34, No. 6, 2022, p. e2455.

[4] A. Joskowski, A. Przybyłek, and B. Marcinkowski, "Scaling Scrum with a customized Nexus Framework: A report from a joint industry-academia research project," *Software: Practice and Experience*, Vol. 53, No. 7, 2023, pp. 1525–1542.

[5] M.O. Ahmad, "5G secure solution development and security master role," in *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering*, 2024, pp. 629–633.

[6] T.N. Bauer, "Onboarding new employees: Maximizing success," *SHRM Foundation's Effective practice guidelines series*, 2010. [Online]. https://www.shrm.org/content/dam/en/shrm/topics-tools/news/talent-acquisition/Onboarding-New-Employees.pdf

[7] P. Gregory, D.E. Strode, H. Sharp, and L. Barroca, "An onboarding model for integrating newcomers into agile project teams," *Information and Software Technology*, Vol. 143, 2022, p. 106792.

[8] J. Buchan, S.G. MacDonell, and J. Yang, "Effective team onboarding in agile software development: techniques and goals," in *ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 2019, pp. 1–11.

[9] A. Ju, H. Sajnani, S. Kelly, and K. Herzig, "A case study of onboarding in software teams: Tasks and strategies," in *IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 613–623.

[10] R. Britto, D. Smite, L.O. Damm, and J. Börstler, "Evaluating and strategizing the onboarding of software developers in large-scale globally distributed projects," *Journal of Systems and Software*, Vol. 169, 2020, p. 110699.

[11] P. Herath, M.O. Ahmad, and T. Gustavsson, "Onboarding newcomers in large-scale agile projects," in *SAC '26: Proceedings of the 41st ACM/SIGAPP Symposium on Applied Computing*, 2026, pp. 1002–1009.

[12] M.O. Ahmad, "Effective handling of large scale agile secure solution development teams," *Journal of Decision Systems*, Vol. 34, No. 1, 2025, p. 2458875.

[13] P. Patanakul, "Managing large-scale IS/IT projects in the public sector: Problems and causes leading to poor performance," *The Journal of High Technology Management Research*, Vol. 25, No. 1, 2014, pp. 21–35.

[14] M. Bloch, S. Blumberg, and J. Laartz, "Delivering large-scale IT projects on time, on budget, and on value," *Harvard Business Review*, Vol. 5, No. 1, 2012, pp. 2–7.

[15] M.O. Ahmad and O. Al-Baik, "Beyond technical debt unravelling organisational debt concept," in *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*, 2024, pp. 802–809.

[16] D.A. Tamburri, P. Kruchten, P. Lago, and H.v. Vliet, "Social debt in software engineering: Insights from industry," *Journal of Internet Services and Applications*, Vol. 6, No. 1, 2015, p. 10.

[17] T. Gustavsson, M.O. Ahmad, and H. Saeeda, "Job satisfaction at risk: Measuring the role of process debt in agile software development," *Journal of Systems and Software*, Vol. 222, 2025, p. 112350.
[18] M.O. Ahmad and T. Gustavsson, "The Pandora's box of social, process, and people debts in software engineering," *Journal of Software: Evolution and Process*, Vol. 36, No. 2, 2024, p. e2516.
[19] T.N. Bauer and B. Erdogan, "Organizational socialization: The effective onboarding of new employees," in *APA Handbook of Industrial and Organizational Psychology, Vol. 3: Maintaining, Expanding, and Contracting the Organization*, S. Zedeck, Ed. American Psychological Association, 2011.
[20] J.E. Van Maanen and E.H. Schein, "Toward a theory of organizational socialization," MIT Alfred P. Sloan School of Management, Working paper 960-77, 1977.
[21] G.R. Jones, "Socialization tactics, self-efficacy, and newcomers' adjustments to organizations," *Academy of Management Journal*, Vol. 29, No. 2, 1986, pp. 262–279.
[22] N.B. Moe, V. Stray, and M.R. Goplen, "Studying onboarding in distributed software teams: A case study and guidelines," in *Proceedings of the 24th International Conference on Evaluation and Assessment in Software Engineering*, 2020, pp. 150–159.
[23] R. Yates, N. Power, and J. Buckley, "Characterizing the transfer of program comprehension in onboarding: An information-push perspective," *Empirical Software Engineering*, Vol. 25, No. 1, 2020, pp. 940–995.
[24] F. Fagerholm, A.S. Guinea, J. Münch, and J. Borenstein, "The role of mentoring and project characteristics for onboarding in open source software projects," in *Proceedings of the 8th ACM/IEEE international symposium on empirical software engineering and measurement*, 2014, pp. 1–10.
[25] I. Steinmacher, A.P. Chaves, T.U. Conte, and M.A. Gerosa, "Preliminary empirical identification of barriers faced by newcomers to Open Source Software projects," in *Brazilian Symposium on Software Engineering*. IEEE, 2014, pp. 51–60.
[26] K.J. Stol, V. Stray, N.B. Moe, and T. Sporsem, "Onboarding software professionals in a hybrid world," *IEEE Transactions on Software Engineering*, 2026.
[27] D. Šmite, F. Zieris, and L.O. Damm, "A wave of resignations in the aftermath of remote onboarding," *Journal of Systems and Software*, 2026, p. 112872.
[28] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical software engineering*, Vol. 14, No. 2, 2009, pp. 131–164.
[29] P. Bansal and K. Corley, "The coming of age for qualitative research: Embracing the diversity of qualitative methods," 2011, pp. 233–237.
[30] A. Dubois and L.E. Gadde, "'Systematic combining' – A decade later," *Journal of Business Research*, Vol. 67, No. 6, 2014, pp. 1277–1284.
[31] K.M. Eisenhardt and M.E. Graebner, "Theory building from cases: Opportunities and challenges," *Academy of management journal*, Vol. 50, No. 1, 2007, pp. 25–32.
[32] I. Etikan, S.A. Musa, R.S. Alkassim et al., "Comparison of convenience sampling and purposive sampling," *American journal of theoretical and applied statistics*, Vol. 5, No. 1, 2016, pp. 1–4.
[33] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative research in psychology*, Vol. 3, No. 2, 2006, pp. 77–101.
[34] V. Braun and V. Clarke, "Reflecting on reflexive thematic analysis," *Qualitative research in sport, exercise and health*, Vol. 11, No. 4, 2019, pp. 589–597.
[35] H.J. Klein, B. Polin, and K. Leigh Sutton, "Specific onboarding practices for the socialization of new employees," *International Journal of Selection and Assessment*, Vol. 23, No. 3, 2015, pp. 263–283.
[36] A. Bacchelli and C. Bird, "Expectations, outcomes, and challenges of modern code review," in *35th international conference on software engineering (ICSE)*. IEEE, 2013, pp. 712–721.
[37] T. Enberg, S. Alander, and M. Paasivaara, "Onboarding for an agile software development company," in *International Conference on Agile Software Development*. Springer, 2024, pp. 144–162.
[38] M. Fowler, J. Highsmith et al., "The agile manifesto," *Software development*, Vol. 9, No. 8, 2001, pp. 28–35.

[39] A.C. Edmondson, *The Fearless Organization: Creating Psychological Safety in the Workplace for Learning, Innovation, and Growth*. John Wiley and Sons, 2018.
[40] M.O. Ahmad, "Psychological safety, leadership and non-technical debt in large-scale agile software development," in *18th Conference on Computer Science and Intelligence Systems (FedCSIS)*. IEEE, 2023, pp. 327–334.
[41] M.O. Ahmad, "Strengthening large-scale agile teams: The interplay of high-quality relationships, psychological safety, and learning from failures," *Journal of Software: Evolution and Process*, Vol. 37, No. 1, 2025, p. e2759.
[42] V. Stray, N.B. Moe, and G.R. Bergersen, "Are daily stand-up meetings valuable? A survey of developers in software teams," in *International Conference on Agile Software Development*. Springer, 2017, pp. 274–281.
[43] J.E. Hannay, T. Dybå, E. Arisholm, and D.I. Sjøberg, "The effectiveness of pair programming: A meta-analysis," *Information and software technology*, Vol. 51, No. 7, 2009, pp. 1110–1122.
[44] Scaled Agile, Inc., "Safe studio," 2025, accessed: 24 February 2026. [Online]. https://studio.scaledagile.com/en/unauthenticated
[45] Y. Andriyani, R. Hoda, and R. Amor, "Reflection in agile retrospectives," in *International conference on agile software development*. Springer, 2017, pp. 3–19.
[46] K. Conboy, S. Coyle, X. Wang, and M. Pikkarainen, "People over process: Key challenges in agile development," *IEEE Software*, Vol. 28, No. 4, 2011, pp. 48–57.
[47] E. Tom, A. Aurum, and R. Vidgen, "An exploration of technical debt," *Journal of Systems and Software*, Vol. 86, No. 6, 2013, pp. 1498–1516.
[48] P. Gregory, D.E. Strode, R. AlQaisi, H. Sharp, and L. Barroca, "Onboarding: How newcomers integrate into an agile project team," in *International conference on agile software development*. Springer, 2020, pp. 20–36.
[49] A.F. Rutkowski and C. Saunders, *Emotional and Cognitive Overload: The Dark Side of Information Technology*. Routledge, 2018.
[50] Z. Feng, P. Tandan, I. Steinmacher, M.A. Gerosa, and A. Sarma, "Restructure this: Using AI to restructure onboarding documents to reduce cognitive overload," *arXiv preprint arXiv:2605.19174*, 2026.
[51] M.O. Ahmad and T. Gustavsson, "Nexus between psychological safety and non-technical debt in large-scale agile enterprise resource planning systems development," in *Conference on Practical Aspects of and Solutions for Software Engineering*. Springer, 2023, pp. 63–81.
[52] M. Hartley, "Enhancing employee engagement and retention: The critical role of psychological safety in the onboarding process," *Strategic HR Review*, Vol. 24, No. 3, 2025, pp. 100–105.
[53] R. Britto, D.S. Cruzes, D. Smite, and A. Sablis, "Onboarding software developers and teams in three globally distributed legacy projects: A multi-case study," *Journal of Software: Evolution and Process*, Vol. 30, No. 4, 2018, p. e1921.
[54] Z. Li, P. Avgeriou, and P. Liang, "A systematic mapping study on technical debt and its management," *Journal of systems and software*, Vol. 101, 2015, pp. 193–220.
[55] J.W. Yarbrough and L. Ramos Salazar, "Virtual onboarding and socialization, an exploration of employee discussions and experiences," *Corporate Communications: An International Journal*, Vol. 28, No. 5, 2023, pp. 707–723.
[56] L. Hemphill and A. Begel, "Not seen and not heard: Onboarding challenges in newly virtual teams," Microsoft white Paper, 2016. [Online]. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/Not20Seen20and20Not20Heard20MSR-TR-2011-136.pdf

Appendix A. Interview guide

The interview guide used during the semi-structured interviews is presented below.

Appendix A.1. Background and introduction

- Would you like to tell me a about yourself, your background and your current assignment?
- How long have you been in the industry?
- Could you tell me a little about this software, this product or the system that you're working with?

Appendix A.2. Development practices and quality

- If we start talking about development, do you follow any specific coding standards or quality rules when you develop a system?
- What do you think about this way of having high quality?
- Can you describe how code reviews are conducted within your team?

Appendix A.3. Cross-team collaboration and communication

- What does it look like in your team and the history of your team? How has it changed?
- Do you think that is a good team composition that you have, or would you need other skills in your team?
- What about cooperation with other teams?
- Can you describe any challenges or disagreements that arise within your team or between teams?
- What kind of dependencies do you have, and how do you coordinate stuff with other teams?
- What channels do you use for communication, or do you have rules on that?
- What about knowledge sharing to get to learn from other teams?

Appendix A.4. Onboarding

- What does it look like with onboarding when you get new people into the team?
- When you do this onboarding, do you have any formal process behind that, or is it more informal?
- When a new member joins, how do you get them across the domain and technical aspects?
- Did you get help from the client side or other teams during the onboarding process, or was it primarily from your colleagues?
- How much do a newcomer's personality and team fit play into your decision when hiring a new team member?
- Can you describe any challenge you experienced during the onboarding process?
- What improvements will you suggest for the current onboarding process?

Appendix A.5. Agile ceremonies and practices

- What about other ceremonies or meeting structures that you do?
- Do you have sprint planning? standup meetings, demos or any retrospectives?
- Do you think the ceremonies work well, or is there something that should be changed in the way you do these?
- Did you get any kind of formal training related to Agile?

Appendix A.6. Organizational context

- What are the different roles supporting development activities?
- How are these different roles integrated into the team?
- What roles exist outside the development team, and how they contribute to coordination across teams?
- If you look at the rest of the organization, is there anything that's hindering you or stopping you from doing a good job?

Appendix A.7. Closing

- If you could change one thing about the way your organization works, what would it be?
- Is there anything else you would like to add?

Authors and affiliations

Pavithra Herath
e-mail: pavithra.herath@kau.se
ORCID: https://orcid.org/0009-0007-3510-1433
Department of Computer Science, Karlstad University, Sweden

Muhammad Ovais Ahmad
e-mail: ovais.ahmad@kau.se
ORCID: https://orcid.org/0000-0002-7885-0369
Department of Computer Science, Karlstad University, Sweden

Tomas Gustavsson
e-mail: tomas.gustavsson@kau.se
ORCID: https://orcid.org/0000-0002-1512-6592
Karlstad Business School, Karlstad University,Sweden

e-Informatica Software Engineering Journal (EISEJ) is an international, fully open-access (CC-BY 4.0 without any fees for both authors and readers), blind peer-reviewed computer science journal using a fast, continuous publishing model (papers are edited, assigned to volume, receive DOI & page numbers, and are published immediately after acceptance without waiting months in a queue to be assigned for a specific volume/issue) without a paper length limit.

The journal was co-financed under the programme "Development of scientific journals" from the funds of the Ministry of Education and Science, contract no RCN/SP/0219/2021/1. The Journal is now financially supported by the Department of Artificial Intelligence of Wrocław University of Science and Technology, Poland. The journal is published by Wrocław University of Science and Technology under the auspices of the Software Engineering Section of the Committee on Informatics of the Polish Academy of Sciences.

Aims and Scope

The purpose of **e-Informatica Software Engineering Journal** is to publish original and significant results in all areas of software engineering research.

Our focus is on empirical software engineering, as well as data science (in particular, artificial intelligence and machine learning methods) in software engineering.

e-Informatica Software Engineering Journal is published online. The online version is from the beginning published as a gratis, no authorship fees, open-access journal, which means that it is available at no charge to the public.

Topics of interest

- AI/ML/LLMs for software engineering
- Software engineering for AI
- Software requirements engineering and modeling
- Software architectures and design
- Software components and reuse
- Software testing, analysis and verification
- Agile software development methodologies and practices
- Model driven development
- Software quality
- Software measurement and metrics
- Reverse engineering and software maintenance
- Empirical and experimental studies in software engineering (incl. replications)
- Evidence-based software engineering
- Systematic reviews and mapping studies (see SEGRESS guidelines)
- Statistical analyses and meta-analyses of experiments
- Robust statistical methods
- Reproducible research in software engineering
- Object-oriented software development
- Aspect-oriented software development
- Software tools, containers, frameworks and development environments
- Formal methods in software engineering.
- Internet software systems development
- Dependability of software systems
- Human-computer interaction
- AI and knowledge based software engineering
- Data science in software engineering
- Prediction models in software engineering
- Mining software repositories
- Search-based software engineering
- Multiobjective evolutionary algorithms
- Tools for software researchers or practitioners
- Project management
- Software products and process improvement and measurement programs
- Process maturity models

Funding acknowledgements: Authors are requested to identify who provided financial support for the conduct of the research and/or preparation of the article and to briefly describe the role of the sponsor(s), if any, in study design; in the collection, analysis, and interpretation of data; in the writing of the paper. If the funding source(s) had no such involvement, then this should be stated as well.

Detailed paper requirements can be found on the Paper Requirements and Recommendations web page.

The submissions will be accepted for publication on the basis of positive reviews done by the international Editorial Board and reviewers.

English is the only accepted language of publication.

To submit an article, please use the Manuscript Central journal submission web page.

Subsequent issues of the journal will appear continuously according to the reviewed and accepted submissions.

The journal is included in the ISI Web of Science, Scopus, DBLP, Google Scholar, DOAJ, BazTech etc.

http://www.e-informatyka.pl/



e-Informatica

ISSN 1897-7979