

# e-Informatica

software engineering journal

2015

volume 9

issue 1



e-Informatica



**e-Informatica**  
software engineering journal

2015 volume 9 issue 1



e-Informatica



Wrocław University of Technology

Editors

Zbigniew Huzar (*Zbigniew.Huzar@pwr.edu.pl*)

Lech Madeyski (*Lech.Madeyski@pwr.edu.pl*, <http://madeyski.e-informatyka.pl/>)

Institute of Informatics

Wrocław University of Technology, 50-370 Wrocław, Poland

e-Informatica Software Engineering Journal

[www.e-informatyka.pl/wiki/e-Informatica/](http://www.e-informatyka.pl/wiki/e-Informatica/), DOI: 10.5277/e-informatica

Editorial Office Manager: Wojciech Thomas

Proofreader: Anna Tyszkiewicz

Typeset by Wojciech Myszka with the L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> Documentation Preparation System

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, transmitted in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publishers.

Printed in the camera ready form

© Copyright by Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2015

OFICYNĄ WYDAWNICZĄ POLITECHNIKI WROCŁAWSKIEJ

Wybrzeże Wyspiańskiego 27, 50-370 Wrocław

<http://www.oficyna.pwr.edu.pl>;

e-mail: [oficwyd@pwr.edu.pl](mailto:oficwyd@pwr.edu.pl); [zamawianie.ksiazek@pwr.edu.pl](mailto:zamawianie.ksiazek@pwr.edu.pl)

ISSN 1897-7979

Printed by beta-druk, [www.betadruk.pl](http://www.betadruk.pl)

# Editorial Board

## Co-Editors-in-Chief

Zbigniew Huzar (Wrocław University of Technology, Poland)

Lech Madeyski (Wrocław University of Technology, Poland)

## Editorial Board Members

Pekka Abrahamsson (VTT Technical Research Centre, Finland)

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)

Norman Fenton (Queen Mary University of London, UK)

Joaquim Filipe (Polytechnic Institute of Setúbal/INSTICC, Portugal)

Thomas Flohr (University of Hannover, Germany)

Félix García (University of Castilla-La Mancha, Spain)

Carlo Ghezzi (Politecnico di Milano, Italy)

Janusz Górski (Gdańsk University of Technology, Poland)

Andreas Jedlitschka (Fraunhofer IESE, Germany)

Barbara Kitchenham (Keele University, UK)

Stanisław Kozielski (Silesian University of Technology, Poland)

Ludwik Kuźniarz (Blekinge Institute of Technology, Sweden)

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 Technology, Poland)

Zygmunt Mazur (Wrocław University of Technology, Poland)

Bertrand Meyer (ETH Zurich, Switzerland)

Matthias Müller (IDOS Software AG, Germany)

Jürgen Münch (Fraunhofer IESE, Germany)

Jerzy Nawrocki (Poznań Technical University, Poland)

Janis Osis (Riga Technical University, Latvia)

Łukasz Radliński (University of Szczecin, Poland)

Guenter Ruhe (University of Calgary, Canada)

Krzysztof Sacha (Warsaw University of Technology, Poland)

Rini van Solingen (Drenthe University, The Netherlands)

Miroslaw Staron (IT University of Göteborg, Sweden)

Tomasz Szmuc (AGH University of Science and Technology Kraków, Poland)

Iwan Tabakow (Wrocław University of Technology, Poland)

Burak Turhan (University of Oulu, Finland)

Rainer Unland (University of Duisburg-Essen, Germany)

Sira Vegas (Polytechnic University of Madrid, Spain)

Corrado Aaron Visaggio (University of Sannio, Italy)

Bartosz Walter (Poznań Technical University, Poland)

Bogdan Wiszniewski (Gdańsk University of Technology, Poland)

Jaroslav Zendulka (Brno University of Technology, The Czech Republic)

Krzysztof Zieliński (AGH University of Science and Technology Kraków, Poland)



# Contents

Editorial . . . . .	7
Data Flow Approach to Testing Java Programs Supported with DFC <i>Ilona Bluemke, Artur Rembiszewski</i> . . . . .	9
Cross-Project Defect Prediction with Respect to Code Ownership Model: An Empirical Study <i>Marian Jureczko, Lech Madeyski</i> . . . . .	21
Resolving Conflict and Dependency in Refactoring to a Desired Design <i>Iman Hemati Moghadam, Mel Ó Cinnéide</i> . . . . .	37
An Approach to Assessing the Quality of Business Process Models Expressed in BPMN <i>Malgorzata Sadowska</i> . . . . .	57
Using the Cognitive Walkthrough Method in Software Process Improvement <i>Péter Balázs Polgár</i> . . . . .	79
Construction of Variable Strength Covering Array for Combinatorial Testing Using a Greedy Approach to Genetic Algorithm <i>Priti Bansal, Sangeeta Sabharwal, Nitish Mittal, Sarthak Arora</i> . . . . .	87
Model Driven Web Engineering: A Systematic Mapping Study <i>Karzan Wakil, Dayang N.A. Jawawi</i> . . . . .	107





# Editorial

Following the mission of e-Informatica Software Engineering Journal, we present this ninth edition with seven scientific papers. The papers refer to different topics, but belong to the scope of the journal, including methods, practices, technologies and tools in the software development cycle, with particular emphasis on empirical evaluation.

The first paper by Bluemke and Rembiszewski (*Data Flow Approach to Testing Java Programs Supported with DFC* [1]) presents the data flow coverage testing of Java programs. In general, the data flow is considered an effective testing technique for fault localization, but it is not used in industry because supporting tools are not scalable for large programs. The proposed original testing method and the elaborated tool, as an Eclipse plug-in, brings hope to overcome this obstacle. The tool was applied for comparison of testing Java programs using data flow and mutation techniques. The results show that the effectiveness of mutation testing is higher than the effectiveness of data flow testing, but mutation techniques appear to be more expensive than the data flow if time and effort are considered.

The subject of the second paper by Jureczko and Madeyski (*Cross-Project Defect Prediction with Respect to Code Ownership Model: An Empirical Study* [2]) is a statistical analysis of several dozen versions of industrial, open-source and academic projects. The main result of the analysis shows that the open-source, industrial and academic projects may be treated as separate categories of projects with regard to defects prediction, and, in consequence, the prediction models trained on the projects depend on project category. This result does not seem to be surprising, but a more interesting issue is identification of the reasons for these differences. The work makes the next step towards cross-project reusability of defect prediction models and facilitates their adoption, which has been very limited so far.

How to detect conflicts and dependencies in refactoring during software design is the leading problem discussed in the third paper by Moghadam, and Ó Cinnéide (*Resolving Conflict and Dependency in Refactoring to a Desired Design* [3]). A novel automated approach to refactoring scheduling in the presence of inter-refactoring conflicts and depen-

dencies is proposed. Evaluation based on several sample programs and one non-trivial open source application demonstrates the ability of the approach to schedule the input refactorings to achieve the desired design for a medium-sized, real-world application.

A practical proposal of a meta-model for a quality assessment models of actual models expressed as process diagrams in BPMN 2.0 is proposed in the fourth paper by Sadowska (*An Approach to Assessing the Quality of Business Process Models Expressed in BPMN* [4]). The proposal is based on an elaborated quality model containing selected characteristics, respective metrics and, finally, proposed quality criterions. A programming tool implementing the meta-model was designed, and through a survey-based experiment was evaluated. The results showed the usefulness of the tool and the proposed approach.

The fifth paper by Polgár (*Using the Cognitive Walkthrough Method in Software Process Improvement* [5]) may be regarded as an introductory discussion on how to use the cognitive walkthrough method to improve the software development process. An outline of how to apply this method and what are the significant changes necessary for its implementation are briefly presented.

Instead of expensive combinatorial testing,  $t$ -way testing is usually adopted to trigger faults due to interactions between components in a component based software system. A genetic algorithm based on the greedy principle used to generate optimal variable strength covering array is the main focus of the sixth paper by Bansal et al. (*Construction of Variable Strength Covering Array for Combinatorial Testing Using a Greedy Approach to Genetic Algorithm* [6]). The proposed approach was evaluated on several benchmark configurations. The experiments showed that the elaborated algorithm, integrating greedy and meta-heuristic techniques, outperforms, except simulated annealing, other existing state-of-the-art algorithms in terms of variable strength covering array sizes.

In the last paper Wakil and Jawawi (*Model Driven Web Engineering: A Systematic Mapping Study* [7]) present a survey of more than 300 primary studies from last five years on Model Driven Web

Engineering (MDWE) mainly for identification of needs for future research. The paper brings a classification and statistics of the main research topics on MDWE (Web applications, services, modeling, requirements and design, testing and quality, development methodologies, management, and economics), publication forms (conferences, workshops,

journals), and their character (validation, opinion, proposals, experience, evaluation).

We look forward to receiving high quality contributions from researchers and practitioners in software engineering for the next issue of the journal.

Editors  
Zbigniew Huzar  
Lech Madeyski

## References

- [1] I. Bluemke and A. Rembiszewski, "Data flow approach to testing Java programs supported with DFC," *e-Informatica Software Engineering Journal*, Vol. 9, 2015, pp. 9–19. [Online]. [http://www.e-informatyka.pl/attach/e-Informatica\\_-\\_Volume\\_9/eInformatica2015Art1.pdf](http://www.e-informatyka.pl/attach/e-Informatica_-_Volume_9/eInformatica2015Art1.pdf)
- [2] M. Jureczko and L. Madeyski, "Cross-project defect prediction with respect to code ownership model: An empirical study," *e-Informatica Software Engineering Journal*, Vol. 9, 2015, pp. 21–35. [Online]. [http://www.e-informatyka.pl/attach/e-Informatica\\_-\\_Volume\\_9/eInformatica2015Art2.pdf](http://www.e-informatyka.pl/attach/e-Informatica_-_Volume_9/eInformatica2015Art2.pdf)
- [3] I.H. Moghadam and M.O. Cinnéide, "Resolving conflict and dependency in refactoring to a desired design," *e-Informatica Software Engineering Journal*, Vol. 9, 2015, pp. 37–56. [Online]. [http://www.e-informatyka.pl/attach/e-Informatica\\_-\\_Volume\\_9/eInformatica2015Art3.pdf](http://www.e-informatyka.pl/attach/e-Informatica_-_Volume_9/eInformatica2015Art3.pdf)
- [4] M. Sadowska, "An approach to assessing the quality of business process models expressed in BPMN," *e-Informatica Software Engineering Journal*, Vol. 9, 2015, pp. 57–77. [Online]. [http://www.e-informatyka.pl/attach/e-Informatica\\_-\\_Volume\\_9/eInformatica2015Art4.pdf](http://www.e-informatyka.pl/attach/e-Informatica_-_Volume_9/eInformatica2015Art4.pdf)
- [5] P.B. Polgár, "Using the cognitive walkthrough method in software process improvement," *e-Informatica Software Engineering Journal*, Vol. 9, 2015, pp. 79–86. [Online]. [http://www.e-informatyka.pl/attach/e-Informatica\\_-\\_Volume\\_9/eInformatica2015Art5.pdf](http://www.e-informatyka.pl/attach/e-Informatica_-_Volume_9/eInformatica2015Art5.pdf)
- [6] P. Bansal, S. Sabharwal, N. Mittal, and S. Arora, "Construction of variable strength covering array for combinatorial testing using a greedy approach to genetic algorithm," *e-Informatica Software Engineering Journal*, Vol. 9, 2015, pp. 87–105. [Online]. [http://www.e-informatyka.pl/attach/e-Informatica\\_-\\_Volume\\_9/eInformatica2015Art7.pdf](http://www.e-informatyka.pl/attach/e-Informatica_-_Volume_9/eInformatica2015Art7.pdf)
- [7] K. Wakil and D.N.A. Jawawi, "Model driven web engineering: A systematic mapping study," *e-Informatica Software Engineering Journal*, Vol. 9, 2015, pp. 107–142. [Online]. [http://www.e-informatyka.pl/attach/e-Informatica\\_-\\_Volume\\_9/eInformatica2015Art6.pdf](http://www.e-informatyka.pl/attach/e-Informatica_-_Volume_9/eInformatica2015Art6.pdf)

# Data Flow Approach to Testing Java Programs Supported with DFC

Iłona Bluemke\*, Artur Rembiszewski\*

*\*Institute of Computer Science, Warsaw University of Technology*

I.Bluemke@ii.pw.edu.pl, a.rembiszewski@gmail.com

## Abstract

Code based (“white box”) approach to testing can be divided into two main types: control flow coverage and data flow coverage. The data flow testing was introduced to structural programming languages and later adopted for object languages. Among many tools supporting code based testing of object programs, only JaBUTi and DFC (Data Flow Coverage) support data flow testing of Java programs. DFC is a tool implemented at the Institute of Computer Science, Warsaw University of Technology, as an Eclipse plug-in. The objective of this paper is to present data flow coverage testing of Java programs supported by DFC. DFC finds all definition-use pairs in tested unit and also provides the definition-use graph for methods. After test execution, the information which def-use pairs were covered is shown. An example of data flow testing of Java program is also presented.

**Keywords:** data flow testing, coverage testing

## 1. Introduction

One of the key issues in developing software systems is effective testing. Popular approaches to testing include “black box” and “white box”. Black-box and white-box testing are complementary to each other in the sense that they are likely to uncover different classes of faults. Black-box testing focuses on the software functional requirements. It aims at faults related to incorrect or missing functions, interface errors, behavior or performance errors and initialization and termination errors. White-box testing focuses on the internal structure of the program, to guarantee that all independent paths within a code have been executed at least once, exercise all logical decisions on their true and false sides, execute all loops at their boundaries and within their operational bounds and exercise internal data structures. White box approach can be divided into two main types: data flow coverage methods

and control flow coverage. Control flow coverage methods were studied, e.g. by Woodward and Hennell (2006) [1], Malevris and Yates (2006) [2].

The idea of data flow testing has been proposed in the seventies by Herman (1976) [3]. In this testing relationships between data are used to select test cases.

Although experiments conducted in 1999 by Mei-Hwa Chen, Kao H.M. [4] show, that data flow testing applied to object programs can be very effective, this approach is not widely used for object programs. Among many tools supporting code based testing of object programs, only JaBUTi [5] supports dataflow testing of Java programs. At the Institute of Computer Science, Warsaw University of Technology, a tool, called DFC (Data Flow Coverage), for data flow testing of Java program was implemented. DFC is implemented as an Eclipse plug-in so can be used with other testing tools available in the eclipse environment.

The objective of this paper is to present data flow coverage testing of Java programs supported by DFC. The introduction to dataflow approach and related work is given in section 2. DFC, presented in section 3, finds all *definition-use* pairs in tested unit (section 2) and provides also the *definition-use* graph for methods. After the execution of the test, the tester is provided with information whose *def-uses* pairs were covered, so she/he can add new tests for not covered pairs. The tester decides which methods change the state of an object. Such approach is novel and not available in other testing tools. In section 4 an example of a Java program is used to explain the data flow coverage testing. Advantages and disadvantages of data flow testing of Java programs are also discussed. Section 5 contains some remarks.

## 2. Data Flow Testing

In structural testing, also called “white box” testing, tests are derived from the source code. Structural testing methods can be divided into two categories: code coverage and data flow coverage. Each of these techniques includes several criteria defining specific requirements which should be satisfied by the test set. The requirements determined by a testing criterion may be used either for test set evaluation or test set generation. In code coverage methods test data may be chosen, e.g. to execute all statements in the code (statement coverage), or to traverse all branches (branch coverage). Both control-flow and data-flow based testing criteria were originally defined to test procedural programs, they were also extended to object oriented programs. The adequacy of testing activity is related to the determination of test criterion effectiveness. Test effectiveness is related to the task of creating the smallest test set for which the output indicates the largest set of failures. The idea of data flow testing has been introduced by Herman (1976) [3]. Later, it was also studied by Laski and Korel (1983) [6]. Rapps and Weyuker (1985) [7] showed how to select test data using data flow information. Ostrand and Weyuker (1991) [8] were

analyzed the test adequacy in data flow testing. Further research in data flow testing was also made by, e.g. Harrold and Rothermel (1994) [9], Harold and Soffa (1989) [10], Vincenzi, Maldonado, Wong and Delamaro (2005) [11], Laski and Stanley (2009) [12]. Quite recently new methods have been proposed by Chaim and de Araujo (2013, 2014) [13, 14] and Vivanti (2014) [15].

In data flow testing the relations between data are the basis to design test cases. Different sub-paths from *definition* of a variable (assignment) into its *use* are tested.

A definition-use pair (*def-u*) is an ordered pair  $(d, u)$ , where  $d$  is a statement containing definition of a variable  $v$ , and  $u$  a statement containing the use of  $v$  or some memory location bound to  $v$  that can be reached from  $d$  over some program path.

Test criteria are used to select particular definition-use pairs. A test satisfies a *def-u* pair, if executing the program with this test causes the traversal of a sub-path from the definition to the use of this variable  $v$  without any  $v$  redefinition. A *def-u* pair is feasible if there exists some program input being to exercise the pair.

Data-flow testing criteria proposed by Rapps and Weyuker [7] use the *def-use* graph (DUG), which is an extension of the control flow graph (CFG) with information about the set of variables defined – *def()* and used – *use()* in each node/edge of the CFG. An example of DUG for a code in Listing 1 is presented in Figure 1. A program can be uniquely decomposed into a set of disjoint blocks whose characteristic property is the fact that if the first statement of the block is executed, the other statements are executed in a given order. The first statement of the block is the only statement which may be executed directly after the execution of a statement in another block. The last statement is the only one which may have a successor in the execution outside the block. Every conditional transfer must be the last statement of a block. The program graph  $G$  representing a program consists of one node  $i$  corresponding to each block  $b_i$  of the program and an edge from node  $j$  to node  $k$ , denoted  $(j, k)$ , if and only if either the last statement of  $b_i$  is not an uncon-

ditional transfer and it physically precedes the first statement of  $b_k$ , or the last statement of  $b_i$  is a transfer whose target is the first statement of  $b_k$ .

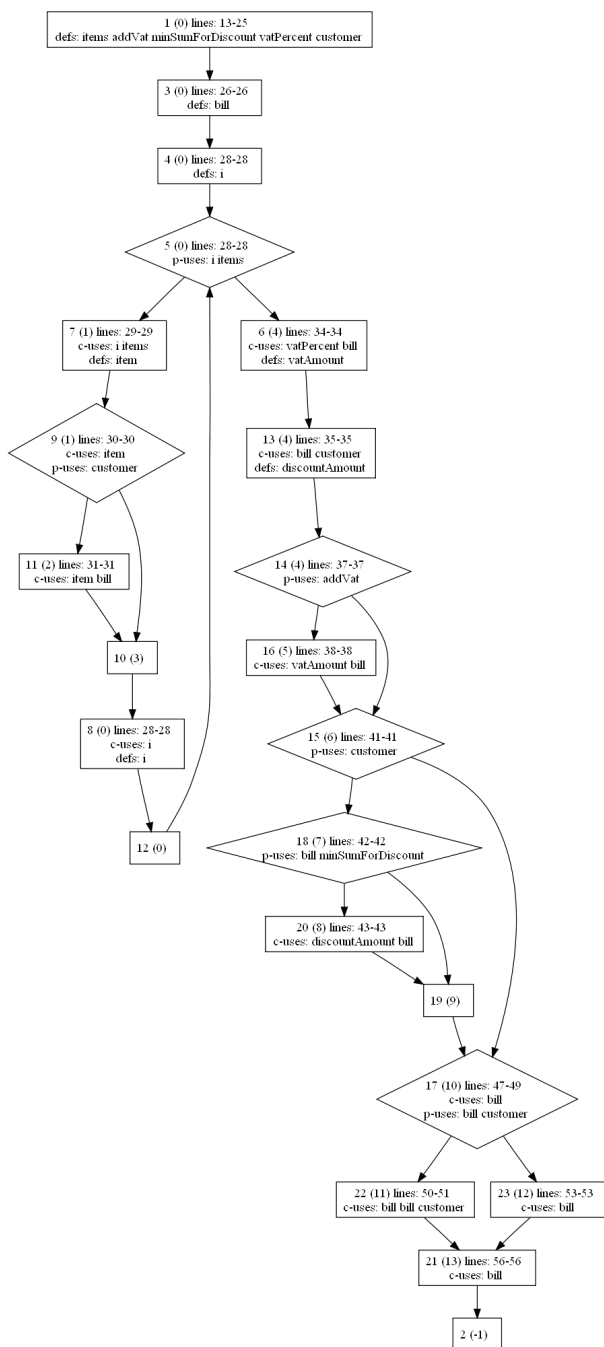


Figure 1. DUG for code given in Listing 1

The node corresponding to the block whose first statement is the start statement of the program is the start node and has no predecessors. A node corresponding to a block whose final statement is a halt statement is known as an exit node and

has no successors. In addition, a node has two successors if and only if, the final statement of its corresponding block is a conditional transfer. All transfer statements should be effective, it means that two successors are different nodes. For every pair of nodes  $i$  and  $j$ , there is at most one edge from node  $i$  to node  $j$ . A path can be represented as a finite sequence of nodes.

In data flow testing the path selection criteria are based on an investigation of the ways in which values are associated with variables and how these associations can affect the execution of the program. This analysis focuses on the occurrences of variables within the program. Each variable occurrence is classified as being a definitional (*def*), computation-use (*c-use*), or predicate-use (*p-use*) occurrence.

The *def/use* graph (DUG) is constructed from a program graph by associating each node  $i$  with the set of variables for which this node  $i$  contains a definition  $def(i)$  and the  $c-use(i)$  (the set of variables for which node  $i$  contains a *c-use*). The edge  $(i, j)$  of DUG is associated with  $p-use(i, j)$  (the set of variables for which edge  $(i, j)$  contains a *p-use*).

Many *def-u* criteria have been proposed and compared. The first criteria, introduced by Rapps and Weyuker, contain e.g.: *all-nodes*, *all-edges*, *all-defs*, *all-du-paths*, *all-p-uses*, *all-c-uses/some-p-use*.

The criterion, called *all-defs* states that for each DUG node  $i$  and all variables  $v$ ,  $v \in def(i)$  (defined in this node) at least one path  $(i, j)$  is covered. In node  $j$  this variable is used  $v \in use(j)$  and on this path the variable  $v$  is not redefined.

The decision which criterion to use as a basis for test data selection depends on several factors, including the size of the program, time and cost requirements and consequence of failure. The “stronger” the selected criterion, the more closely the program is scrutinized in an attempt to locate program faults but a “weaker” criterion can be fulfilled using fewer test cases. The criteria *all-nodes* (statement coverage) and *all-edges* (branch coverage) are often used in program testing despite the fact that it is well known that they are weak criteria. Certainly they represent the necessary conditions, for if some portion of

the program has never been executed, one would not in general feel confident about its behavior. Similar intuition motivated Rapps and Weyuker in the definition of *all-defs* criterion. Even if every statement and branch had been executed, if the result of some computation had never been used, one would have little evidence that the intended computation had been performed.

The *all-uses* criterion requires that test data force some path to be traversed between every definition and each of its uses. A stronger requirement is that test data cause every path between a definition and its uses to be traversed. If the program contains loops, there may be infinitely many such paths. Rapps and Weyuker’s “strongest” criterion, *all-du-paths*, requires that test data cause the traversal of every *du-path* between a definition and each of its uses, thus avoiding this problem. Rapps and Weyuker also proved inclusion between the test criteria shown in Figure 2.

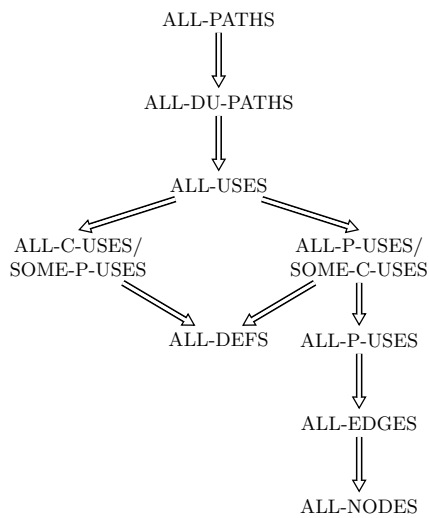


Figure 2. Inclusion of data flow test criteria [7]

The dataflow technique, described above, was dedicated to structural programming languages and does not consider data flow interactions which arise when the methods are invoked in an arbitrary order. In 1989 Harold and Soffa elaborated [10] the inter-procedural data flow testing. They proposed an algorithm, called PLR, to find *def-u* pairs if the variable definition is introduced in one procedure, and the variable usage is in called or calling procedures. The al-

gorithm works on inter-procedural control flow graph built from control flow graphs of dependent procedures. A call site is replaced by a call and a return node. Control flow graphs are connected by added edges from the call node to entry nodes and from exit nodes to return nodes to represent procedure calls in the program. A special entry node represents the entry to the “main” procedure of the program. The PLR algorithm first computes the definition and alias information for each procedure. Then, using the data flow framework, propagates the local information to obtain inter-procedural reaching definitions from which inter-procedural *def-use* pairs can be calculated. This method can be adapted to global variables, class attributes and referenced method arguments in testing object programs. The *def-use* pairs can be used to test possible interactions between methods. Data flow approach to test classes gives opportunities to find errors in classes that may not be uncovered by functional testing.

For object programs, in 1994 Harrold and Rothermel proposed [9] three levels of data flow testing:

- *Intra-method* level is based on the basic Rapps and Weyuker algorithm, it is performed on each method individually; class attributes and methods interactions cannot be taken into account. This level of testing is equivalent to unit testing in procedural language programs.
- *Inter-method* tests are applied to the public method together with other methods in its class that it calls directly or indirectly. *def-u* pairs for class attributes can be found in this approach. This level of testing is equivalent to integration testing of procedures in procedural language programs.
- *Intra-class* – interactions of public methods are tested, when they are called in various sequences. The set of possible sequences of public methods calls is infinite so only the subset of it is tested. Since users of a class may invoke sequences of methods in indeterminate order the *intra-class* testing can increase the confidence at witch sequences of calls interact properly.

For each of the above described testing levels appropriate *def-u* pairs were defined in 1994 by Harrold and Rothermel [9], i.e. *intra-method*, *inter-method* and *intra-class*.

### 3. DFC – A Tool for Data Flow Testing

The process of testing software is extremely expensive in terms of labour, time and cost so many tools supporting this process have been developed but the authors found only one – JaBUTi [5], dedicated to the data flow testing of Java programs (when the research was started in 2008). Data flow testing of object programs can reveal many errors. An experiment described by Mei-Hwa Chen and Kao [4], shows, that in the data flow testing of C++ programs, the number of detected errors was four times greater, than in other code coverage methods, i.e. instructions and conditions coverage. The results of this experiment motivated us to build a tool for data flow testing of Java programs.

Data flow testing cannot be applied in isolation so the authors decided to implement a tool supporting this approach, DFC – Data Flow Coverage (Fig. 3), as an Eclipse plug-in. In Eclipse Java programming environment and testing tools, e.g. JUnit are available. DFC finds all *def-u* pairs in testing Java code and after the test provides tester information which *def-u* pairs were covered. Based on this information, the tester can decide which coverage criteria should be used and add appropriate test cases (shown in section 4). In preparing the test cases the tester can also use *def-use* graph (DUG) for a method provided by DFC.

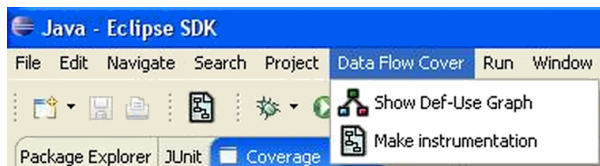


Figure 3. DFC menu

In object languages the data flow testing ideas proposed for structural languages must be modified. One of the main problems which must be solved is identification which method is able to

modify the object state and which one is only using it. In DFC *def-u* pairs are the *intra-method*. Definitions of class attributes are located in the first node of the DUG graph of the tested method. The first node of DUG also contains definitions of arguments of the tested method.

**Definitions of variable *x*** are e.g.:

1. `int x; Object x; x = 5; x = y; x = new Object();`  
`x = get_object(param);`
2. *x* is an object and a state modifying method, it is called in its context:  
`x.method1();`
3. *x* is an object and one of its attributes is modified:  
`x.a = 5;`

An instruction **uses a variable *x***, e.g.:

1. its value is assigned:  
`w = 2*x; x++;`
2. *x* is an object and a reference is used in an instruction:  
`w = x; method1(x); if (x == null)`
3. *x* is an object and a method, using the state of this object is called in its context:  
`x.method1();`
4. *x* is an object and one of its attributes is used in the instruction:  
`w = 2*x.a;`

In DFC the tester can decide which method defines and which uses the object state.

In Figure 4 the main parts of DFC and its collaboration with the Eclipse environment are presented. The modules of DFC are denoted by bold lines.

The input for DFC is the Java source code (*SRC* in Fig. 4). A DFC user has to identify which file will be tested and indicate it to DFC. Module *Knowledge base* analyses the source code and generates the list of classes and methods. On this list the tester may mark methods as modifying or using object state. The module *Instrumentation* instruments a source code i.e. adds extra instructions needed for finding data flow coverage and builds *def-use* graph (DUG). DUG (example for source code from Listing 1 is shown in Fig. 1) contains information concerning the control flow, variable definitions and usage in its nodes. DUG is the input for module *Visualization*, drawing the graph, and *Re-*

Table 1. Test cases for method `doShopping`

Method	Test Cases						
	1	2	3	4	5	6	7
<code>addVat</code>	false	true	false	false	true	false	false
<code>minSumForDiscount</code>	200	200	20	200	20	200	200
<code>vatPercent</code>	20	20	20	20	20	20	20
<code>customer</code>	item1,	item1,	item1,	item1,	item1,	item1,	
<code>needItems</code>	item5	item5	item5	item5	item5	item5	
<code>customer</code>	10	10	10	10	10	10	10
<code>getDiscountPercent ()</code>							
<code>customer</code>	100	100	100	100	100	50	100
<code>getMoneyAmount ()</code>							
<code>customer.isSpecial()</code>	false	false	true	true	true	false	false

quirements – finding all *def-u* pairs. The constructed DUG graph is presented after pressing Show DUG button in the DFC menu (Fig. 3). The instrumented code should be compiled and run in the Eclipse environment and the responsibility for these activities is taken by the programmer.

coverage to DFC. The *Analyzing* module locates covered and not covered *def-u* pairs. More details on DFC implementation and its usage were given by Rembiszewski [16] and by Bluemke, Rembiszewski [17].

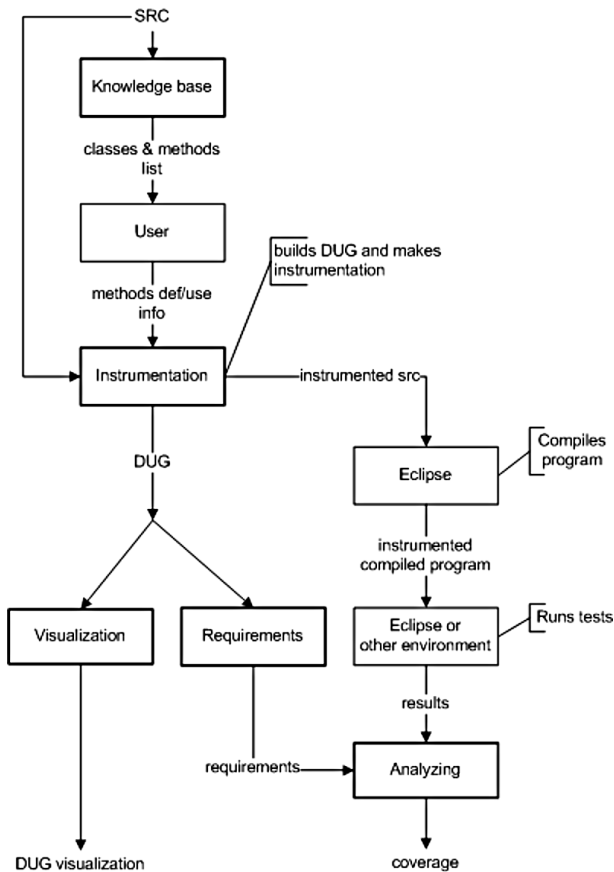


Figure 4. The idea of testing with DFC

During the test execution, an extra code added by *Instrumentation* module sends data concerning

## 4. Example

In this section the data flow testing of Java code is presented in a small example. In Listing 1 the source code of the method `doShopping` from class `Shop` is shown. This method was tested with the DFC tool. The DUG for this method is shown in Figure 1. The *all-defs* coverage criterion was used. In Table 1 the test cases are listed. Column “Name” contains the name of a variable or method returning the private attribute. The following columns contain the values used in test cases. In the objects of class `Item` the method `getPrice` returns 10 for variable `item1` and 50 for `item5`. Code testing, shown in Listing 1, was performed in two phases. In the first phase the *def-use* chains were covered. In DFC all methods were initially identified as not modifying the state of object and using it (similarly to JaBUTi [5]).

**Listing 1.** Method `doShopping` from class `Shop`

```

25) public Bill doShopping(Customer customer) {
26)     Bill bill = new Bill();
27)
28)     for (int i=0; i<items.size(); i++) {
29)         Item item = items.get(i);
30)         if (customer.need(item))
  
```



```

31)    bill.add(item.getPrice());
32)  }
33)
34)    double vatAmount = (vatPercent/100) *
           bill.getTotalSum();
35)    double discountAmount = bill.getTotalSum() *
           (customer.getDiscountPercent()/100);
36)
37)    if (addVat) {
38)        bill.add(vatAmount);
39)    }
40)
41)    if (customer.isSpecial()) {
42)        if (bill.getTotalSum() > minSumForDiscount) {
43)            bill.subtract(discountAmount);
44)        }
45)    }
46)
47)    bill.close();
48)
49)    if (bill.getTotalSum() <= customer.getMoneyAmount()) {
50)        bill.pay();
51)        customer.getFromAccount(bill.getTotalSum());
52)    } else {
53)        bill.cancel();
54)    }
55)
56)    return bill;
57) }

```

When the full coverage of *all-defs* pairs was achieved, DFC was reconfigured; the methods modifying an object state and using it were marked as shown in Figure 5 (such functionality is not available in JaBUTi [5]). The DUG graph after this modification is presented in Figure 6.

After the re-execution of tests, new test cases for not covered *def-use* pairs were added. The test results are given in Table 2.

The “Definition” column shows the name and line number containing the definition. If a line contains two definitions (e.g. line 28) of the same variable, the column of definition is given after a coma. Definitions 1–11 were found in the first phase, while definitions 12–18 in the second one. It can be observed, that definitions found in the first phase are the subset of definitions found in the second The “Test cases” *n*, column contains letter *Y*, if this

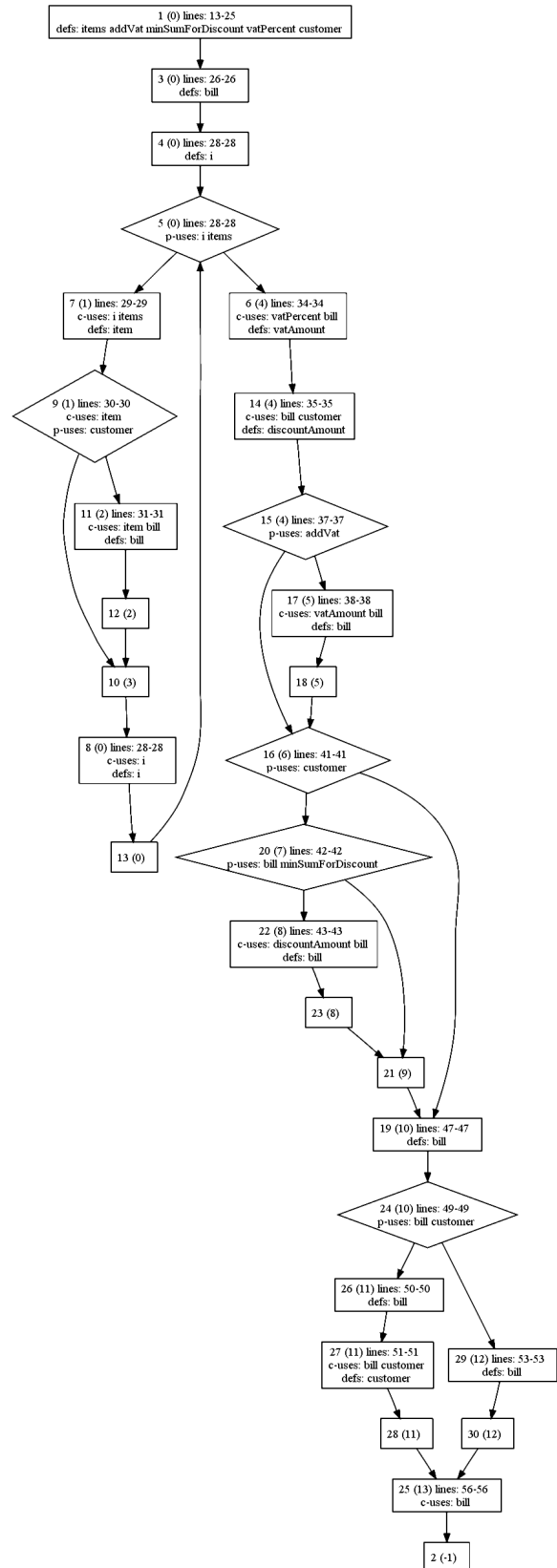


Figure 5. DUG for code in Listing 1 after the indication of methods changing object state

Table 2. The results of tests for method `doShopping`

Definition		Line	Test Cases					
Name	1		2	3	4	5	6	
1	<code>items</code>	13	Y(29)	Y(29)	Y(29)	Y(29)	Y(29)	Y(29)
2	<code>addVat</code>	14	N	Y(37)	Y(37)	Y(37)	Y(37)	Y(37)
3	<code>minSumForDiscount</code>	15	N	N	N	Y(42)	Y(42)	Y(42)
4	<code>vatPercent</code>	16	Y(34)	Y(34)	Y(34)	Y(34)	Y(34)	Y(34)
5	<code>customer</code>	25	Y(30)	Y(30)	Y(30)	Y(30)	Y(30)	Y(30)
6	<code>bill</code>	26	Y(31)	Y(31)	Y(31)	Y(31)	Y(31)	Y(31)
7	<code>i</code>	28,11	Y(28)	Y(28)	Y(28)	Y(28)	Y(28)	Y(28)
8	<code>i</code>	28,32	Y(28)	Y(28)	Y(28)	Y(28)	Y(28)	Y(28)
9	<code>item</code>	29	Y(30)	Y(30)	Y(30)	Y(30)	Y(30)	Y(30)
10	<code>vatAmount</code>	34	N	Y(38)	Y(38)	Y(38)	Y(38)	Y(38)
11	<code>discountAmount</code>	35/39	N	N	Y(43)	Y(43)	Y(43)	Y(43)
12	<code>bill</code>	31	Y(34)	Y(34)	Y(34)	Y(34)	Y(34)	Y(34)
13	<code>bill</code>	38/37	N	N	N	N	Y(43)	Y(43)
14	<code>bill</code>	43	–	–	–	–	–	–
15	<code>bill</code>	47	N	N	N	N	N	Y(49)
16	<code>bill</code>	50	Y(51)	Y(51)	Y(51)	Y(51)	Y(51)	Y(51)
17	<code>bill</code>	53	N	N	N	N	N	Y(56)
18	<code>customer</code>	51	–	–	–	–	–	–
Coverage – phase 1			64%	82%	91%	100%	100%	100%
Coverage – phase 2			56%	69%	75%	81%	87%	100%

definition is in covered *def-use* pairs in phase. The line number of the covered usage is given in brackets. Letter *N* in column of Table 2 means, that the covered pair does not contain this definition and char “–” is written, if reachable usage for this definition does not exist.

In the last two rows (Tab. 2) the data flow coverage for the two phases are calculated. This coverage was calculated as a percentage of covered pairs to all pairs which could be covered. The first phase needed the first four test cases to cover all *def-use* pairs. In the second phase the methods modifying the state of object and using it were marked manually in DFC on the screen shown in Figure 6. The same four test cases produced the coverage of only 81% so some new test cases have to be constructed. **Test case number 5** revealed an error in the `doShopping` method. In this test case `vat` is added to the bill and the client receives also a discount. In the code (Listing 1) the discount is calculated before the `vat` is added instead of being calculated afterwards. The modified code is given in Listing 2. We tried not to change line numbering as much as possible. In Table 2, in rows 11 and 13,

the new line numbers follow the slash character. After the modification, all test, were re-executed and another test case, **number 6**, was added to obtain the 100% coverage. In this example we showed, that identification which method is modifying the object’s state forced the tester to add a test case revealing an error.

In the code given in Listing 1, the definition of the `bill` in line 43 and the `customer` in line 51 no reachable usage exist. According to data flow coverage rules applied to structural languages such a situation can be seen as an anomaly in the code. In object programs such a situation is not an indicator of code anomalies. The `bill` is also defined in line 47. For a simple variable two successive assignments are incorrect, the second one, erases the first one. For an object variable successive assignments, e.g. changing object’s state, may be reasonable. The `customer`, defined in line 51 is not used in the method. This is not an indicator of code anomalies because `customer` is a referenced argument of the `doShopping` method and can be used outside this method.

The success of testing (phase 2) strongly depends on the correct identification of the method

Data Flow Cover - methods information			
Class or Variable	Method	Can modify object state	Uses object state
Customer	isSpecial(0)	No	Yes
Bill	getTotalSum(0)	No	Yes
ArrayList<Item>	size(0)	No	Yes
Bill	add(1)	Yes	Yes
Customer	getMoneyAmount(0)	No	Yes
Customer	need(1)	No	Yes
Customer	getDiscountPercent(0)	No	Yes
Bill	subtract(1)	Yes	Yes
Item	getPrice(0)	No	Yes
Bill	close(0)	Yes	No
Bill	cancel(0)	Yes	No
Customer	getFromAccount(1)	Yes	Yes
Bill	pay(0)	Yes	No
ArrayList<Item>	get(1)	No	Yes

Figure 6. Configuration screen in DFC

definition and use of the object's state. Test cases 1–6 from Table 1 do not test program execution if `bill` is empty. This is tested in test case 7. This test case would be executed if the method `adds` in class `Bill` was marked as modifying an object but not using it. To cover the definition of `bill` in line 26 (Listing 1 and Tab. 2) the instruction in line 34 should be executed but the redefinition in line 31 should be skipped. In this example the simple coverage criteria *all-defs* was used. Other coverage criteria, e.g. *all-uses* can reveal the error in the `doShopping` method without manually setting which method modifies and which one only uses the object state.

#### Listing 2. Modified method `doShopping`

```

25) public Bill doShopping(Customer customer) {
26)     Bill bill = new Bill();
27)
28)     for (int i=0; i<items.size(); i++) {
29)         Item item = items.get(i);
30)         if (customer.need(item))
31)             bill.add(item.getPrice());
32)     }
33)
34)     double vatAmount = (vatPercent/100) *
35)         bill.getTotalSum();
36)
37)     if (addVat) {
38)         bill.add(vatAmount);
39)     }
40)     double discountAmount = bill.getTotalSum() *
41)         (customer.getDiscountPercent()/100);

```

```

41)     if (customer.isSpecial()) {
42)         if (bill.getTotalSum() > minSumForDiscount) {
43)             bill.subtract(discountAmount);
44)         }
45)     }
46)
47)     bill.close();
48)
49)     if (bill.getTotalSum() <=
50)         customer.getMoneyAmount()) {
51)         bill.pay();
52)         customer.getFromAccount(bill.getTotalSum());
53)     } else {
54)         bill.cancel();
55)     }
56)     return bill;
57) }

```

#### 4.1. DFC and JaBUTi

In JaBUTi [5] every call of a method in the context of an object variable is treated as using an object state. In DFC a method call is treated as using the object state if the state of an object variable is not changed. An example given in Listing 3.

#### Listing 3. Simple example

```

1) a = new Object();
2) if(...)
3) a.setState(...);
4) a.m();

```

JaBUTi will not notice the coverage of *def-use* pair in lines (3, 4). In DFC it is possible, if the `setState` method is correctly indicated as modifying the object state. This simple example shows, that DFC is able to treat more instructions as defining, it is able to show the coverage of a greater number of *def-use* pairs and more errors can possibly be detected.

## 5. Conclusions

Many authors, e.g. Beizer [18] suggest that effective testing can be achieved if different testing approaches, e.g. functional and structural are used. In the development of software systems thorough testing can be the crucial issue. In this paper the authors presented DFC, an Eclipse plug-in, designed and implemented at the Institute of Computer Science Warsaw University of Technology, supporting data flow testing of the Java methods. By supporting data flow testing of Java classes we provide opportunities to find an error which may not be uncovered by black box testing. In the Eclipse environment there are other tools available for testing Java programs using different techniques, e.g. JUnit [19], EclEmma [20] or TPTP [21]. EclEmma provides information about instruction coverage. In DFC a tester can design tests to achieve, e.g. *def-uses* or *all-uses* coverage criteria which also guarantee instruction coverage (as proved by Rapps and Weyuker in [7]).

It has been shown that the data flow is an effective testing technique (e.g. in Hutchins et.al. [22]), very useful in fault localization (e.g. in Santelices et.al. [23]) but it is not used in industry. This phenomenon can be explained by the fact that tools supporting data flow testing are not scalable for large systems due to the costs associated with tracking *def-u* associations at the run time. Recently (2013), Chaim and de Araujo [13] have proposed a novel algorithm, called Bitwise Algorithm (BA) to tackle this problem. The new algorithm utilizes efficient bitwise operations and data structures to track the intra procedural *def-u*. They also showed that BA is at least as good as the most efficient data flow instrumentation tech-

niques, and that it can be up to 100% more efficient. In 2014 de Araujo and Chaim [14] presented the BA implementation in programs compiled into byte codes. Maybe their results will encourage vendors to consider including data flow testing in commercial testing tools.

In 2011 Bluemke and Kulesza [24] compared the data flow and the mutation testing of several Java programs. Experiments were conducted in the Eclipse environment. DFC plugin was used to support the data flow testing while MuClipse [25] and Jumble [26] plugins were used for the mutation testing. The results of testing six Java programs using data flow and mutation techniques showed that the effectiveness of mutation testing is higher than the effectiveness of data flow testing. The mutation technique appeared also to be more expensive than the data flow one, if time and effort are considered.

Finally, the authors have outlined the direction for the future research. An interesting and important study would be applying DFC in industry projects to evaluate the cost and benefits of data flow based criteria in testing Java programs. Unfortunately for several years the authors have not been able to raise interest in this subject in the software industry (the tool is available for free and only three university researchers downloaded it, none from industry). One of the reasons may be the effort needed in this approach. In DFC a tester manually identifies defining and using methods (Fig. 4). However, this process is time consuming, it cannot be conducted automatically. To identify if a method is defining or using the object state, the analysis of the source code must be performed. In complex, industry programs many libraries are used so the access to the source code is limited. Decompilation of the library code preceding the analysis process, or the comparison of the value returned by `hashCode()` before and after the method call (this approach needs additional code instrumentation and re-execution of test cases) might be a solution. Comparing the effort needed with possible obtained results, the authors think it is not worth to implement these approaches. In JaBUTi [5], other tool supporting data flow testing of Java program, every call of a method is treated as using object state. In section 4 the authors demonstrated an example showing that

for some programs the identification of methods defining object's state enables finding more errors.

## References

- [1] M.R. Woodward and M.A. Hennell, "On the relationship between two control-flow coverage criteria: all JJ-paths and MCDC," *Information and Software Technology*, Vol. 48, No. 7, 2006, pp. 433–440.
- [2] N. Malevris and D.F. Yates, "The collateral coverage of data flow criteria when branch testing," *Information and Software Technology*, Vol. 48, No. 8, 2006, pp. 676–686.
- [3] P. Herman, "A data flow analysis approach to program testing," *Australian Computer Journal*, Vol. 8, No. 3, 1976, pp. 92–96.
- [4] M.H. Chen and H.M. Kao, "Testing object-oriented programs – an integrated approach," in *10th International Symposium on Software Reliability Engineering, Proceedings*. IEEE, 1999, pp. 73–82.
- [5] JaBUTi homepage. (Accessed 12.2007). [Online]. <http://jabuti.incubadora.fapesp.br/>
- [6] J.W. Laski and B. Korel, "A data flow oriented program testing strategy," *Software Engineering, IEEE Transactions on*, No. 3, 1983, pp. 347–354.
- [7] S. Rapps and E.J. Weyuker, "Selecting software test data using data flow information," *IEEE Transactions on Software Engineering*, No. 4, 1985, pp. 367–375.
- [8] T.J. Ostrand and E.J. Weyuker, "Data flow-based test adequacy analysis for languages with pointers," in *Proceedings of the symposium on Testing, analysis, and verification*. ACM, 1991, pp. 74–86.
- [9] M.J. Harrold and G. Rothermel, "Performing data flow testing on classes," in *ACM SIGSOFT Software Engineering Notes*, Vol. 19, No. 5. ACM, 1994, pp. 154–163.
- [10] M.J. Harrold and M.L. Soffa, "Interprocedural data flow testing," in *ACM SIGSOFT Software Engineering Notes*, Vol. 14, No. 8. ACM, 1989, pp. 158–167.
- [11] A.M.R. Vincenzi, J.C. Maldonado, W.E. Wong, and M.E. Delamaro, "Coverage testing of Java programs and components," *Science of Computer Programming*, Vol. 56, No. 1, 2005, pp. 211–230.
- [12] J. Laski and W. Stanley, *Software verification and analysis: An integrated, hands-on approach*. Springer Science & Business Media, 2009.
- [13] M.L. Chaim and R.P.A. De Araujo, "An efficient bitwise algorithm for intra-procedural data-flow testing coverage," *Information Processing Letters*, Vol. 113, No. 8, 2013, pp. 293–300.
- [14] R.P.A. de Araujo and M.L. Chaim, "Data-flow testing in the large," in *IEEE Seventh International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2014, pp. 81–90.
- [15] M. Vivanti, "Dynamic data-flow testing," in *Companion Proceedings of the 36th International Conference on Software Engineering*. ACM, 2014, pp. 682–685.
- [16] A. Rembiszewski, "Data flow coverage of object programs," Master's thesis, Institute of Computer Science, Warsaw University of Technology, 2009, (in Polish).
- [17] I. Bluemke and A. Rembiszewski, "Dataflow testing of Java programs with DFC," in *Advances in Software Engineering Techniques*, ser. Lecture Notes in Computer Science, T. Szmuc, M. Szpyrka, and J. Zendulka, Eds. Springer Berlin Heidelberg, 2012, Vol. 7054, pp. 215–228. [Online]. [http://dx.doi.org/10.1007/978-3-642-28038-2\\_17](http://dx.doi.org/10.1007/978-3-642-28038-2_17)
- [18] B. Beizer, *Software system testing and quality assurance*. Van Nostrand Reinhold Co., 1984.
- [19] JUnit homepage. (Accessed 12.2008). [Online]. <http://www.junit.org/>
- [20] Eclemma 1.2.0. (Accessed 04.2008). [Online]. <http://www.eclemma.org/>
- [21] TPTP: Eclipse test & performance tools platform project. (Accessed 2008). [Online]. <http://www.eclipse.org/tptp/>
- [22] M. Hutchins, H. Foster, T. Goradia, and T. Ostrand, "Experiments of the effectiveness of dataflow – and controlflow – based test adequacy criteria," in *Proceedings of the 16th international conference on Software engineering*. IEEE Computer Society Press, 1994, pp. 191–200.
- [23] R. Santelices, J.A. Jones, Y. Yu, and M.J. Harrold, "Lightweight fault-localization using multiple coverage types," in *IEEE 31st International Conference on Software Engineering*. IEEE, 2009, pp. 56–66.
- [24] I. Bluemke and K. Kulesza, "A comparison of dataflow and mutation testing of Java methods," in *Dependable Computer Systems*. Springer, 2011, pp. 17–30.
- [25] Muclipse homepage. (Accessed 01.2011). [Online]. <http://muclipse.sourceforge.net/index.php>
- [26] Jumble homepage. (Accessed 12.2008). [Online]. <http://jumble.sourceforge.net/index.ht>



# Cross-Project Defect Prediction with Respect to Code Ownership Model: An Empirical Study

Marian Jureczko\*, Lech Madeyski\*\*

*\*Department of Computer Engineering, Wrocław Univeristy of Technology*

*\*\*Faculty of Computer Science and Management, Wrocław University of Technology*

marian.jureczko@pwr.edu.pl, lech.madeyski@pwr.edu.pl

## Abstract

The paper presents an analysis of 83 versions of industrial, open-source and academic projects. We have empirically evaluated whether those project types constitute separate classes of projects with regard to defect prediction. Statistical tests proved that there exist significant differences between the models trained on the aforementioned project classes. This work makes the next step towards cross-project reusability of defect prediction models and facilitates their adoption, which has been very limited so far.

**Keywords:** software metrics, software defect prediction, cross-project prediction

## 1. Introduction

Assuring software quality is known to require time-consuming and expensive development processes. The costs generated by those processes may be minimized when the defects are predicted early on, which is possible by means of defect prediction models [1]. Those models, based on software metrics, have been developed by a number of researchers (see Section 2). The software metrics which describe artifacts of the software development process (e.g. software classes, files) are generally used as the models' input. The model output usually estimates the probability of failure, the occurrence of a defect or the expected number of defects. The predictions are made for a given artifact. The idea of building models on the basis of experienced facts, called inductive inference, is discussed in the context of software engineering by Samuelis [2].

Defect prediction models are extraordinarily useful in software testing process. The available resources are usually limited and, therefore, it may be difficult to conduct the comprehensive

tests on, and the reviews of, all artifacts. Defect prediction models are extraordinarily useful in the software testing process. The available resources are usually limited and, therefore, it may be difficult to conduct the comprehensive tests on, and the reviews of, all artifacts [1]. The predictions may be used to assign varying priorities to different artifacts (e.g. classes) under test [3,4]. According to the 80:20 empirical rule, a small amount of code (often quantified as 20% of the code) is responsible for the majority of software defects (often quantified as 80% of the known defects in the system) [5,6]. Therefore, it may be possible to test only a small amount of artifacts and find a large amount of defects. In short, a well-designed defect prediction model may save a lot of testing efforts without decreasing software quality.

The defect prediction models may give substantial benefits, but in order to build a model, a software measurement program must be launched. According to Kaner and Bond [7], only few companies establish such programs, even fewer succeed with them and many of the com-

panies use them only to conform to the criteria laid down in the Capability Maturity Model [8]. The costs may be one of the reasons behind the limited adoption of the defect prediction models, e.g. an average overhead for metrics collection was estimated to be 4–8% of the overall value [8,9]. Using cross-project defect prediction could reduce the expenditure, since one model may be used in several software projects, which means that it is not necessary to launch a completely new software measurement program for each project. The cross-project defect prediction is also helpful with solving the problems connected with the lack of historical data indispensable to train a model [10]. Unfortunately, the body of knowledge of cross-project defect prediction does not support us with the results that are advanced enough to be used (details in the next section). The intention of this work is to reveal the facts regarding high level prediction boundaries, particularly the possibility of using prediction models across different project code ownership models. The conducted experiments aiming at verifying whether cluster of projects can be derived from the source code ownership model, where the cluster is a group of software projects that share a common prediction model. Such finding eases the application of defect prediction by removing the necessity of training the model. It is enough to identify the cluster a given project belongs to and use the common model.

This study investigates whether there is a relevant difference between industrial, open-source and academic software projects with regard to defect prediction. In order to explore that potential disparity, the data from 24 versions of 5 industrial projects, 42 versions of 13 open-source projects and 17 versions of 17 academic projects were collected. Several defect prediction models were built and the efficiency of the predictions was compared. Statistical methods were used in order to decide whether the obtained differences were significant or not. It is worth mentioning that a somehow related question of suitability of software quality model for projects within different application domains was raised by Villalba et al. [11], whereas the results of other works

which investigate the cross-project defect prediction (i.e. [12,13]) suggest that the code ownership model might be a relevant factor with regard to the prediction performance.

The rest of this paper is organized as follows: subsequent sections describe related work, the design of empirical evaluation used in this study (including the details of data collection and analysis methodology), the descriptive statistics of the collected data, the results of empirical evaluation, threats to the validity of the empirical study, and conclusions.

## 2. Related Work

Cross-project reusability of defect prediction models would be extremely useful. Such generalized prediction models would serve as a starting point in software development environments that cannot provide historical data and, as a result, they would facilitate the adoption of defect prediction models.

A preliminary work in this area was conducted by Subramanyam and Krishnan [14]. The authors investigated a software project where C++, as well as Java, were employed and found substantial differences between classes written in different programming languages with regard to defect prediction, e.g. the interaction effect (the defect count grows with the CBO value for C++ class but decreases for the Java classes; the relation was calculated with respect to the DIT metric). Hence, the results indicate issues with regard to cross-language predictions. The authors investigated only one project, nevertheless, similar difficulties might arise in cross-project predictions.

Nagappan et al. [15] analyzed whether defect predictors obtained from one project history are applicable to other projects. It turned out that there was no single set of metrics that would fit into all five investigated projects. The defect prediction models, however, could be accurate when obtained from similar projects (the similarity was not precisely defined, though). The study was extended by Zimmerman et al. [12], who performed 622 cross-project predictions for 12 real world applications. A project was considered



as a strong predictor for another project when all precision, recall, and accuracy were greater than 0.75. Only 21 cross-project validations satisfied this criterion, which sets the success rate at 3.4%. Subsequently, the guidelines to assess the cross-project prediction chance of success were given. The guidelines were summarized in a decision tree. The authors constructed separate trees for assessing prediction precision, recall, and accuracy, but only the tree for precision was given in the paper.

A study of cross-company defect prediction was conducted by Turhan et al. [10]. The authors concluded that there is no single set of static code features (metrics) that may serve as defect predictor for all the software projects. The effectiveness of the defect prediction models was measured using probability of detection (pd) and probability of false alarm (pf). Cross-company defect prediction dramatically increased pd as well as pf. The authors were also able to decrease the pf by applying the nearest neighbor filtering. The similarity measure was the Euclidean distance between the static code features. However, there was still a drawback for cross-company defect prediction models: the project features which might influence the effectiveness of cross-company predictions were not identified.

In an earlier paper [13], we presented an empirical study showing that data mining techniques may be used to identify project clusters with regard to cross-project defect prediction. The k-means and Kohonen's neural networks were applied to correlation vectors in order to identify the clusters. The correlation vectors were calculated for each version of each project respectively and represented Pearson's correlation coefficients between software metrics and numbers of defects. Subsequently, a defect prediction model was created for each identified cluster. In order to validate the existence of a cluster, the efficiency of the cluster model was compared with the efficiency of a general model. The general model was trained using data from all the projects. Six different clusters were identified and the existence of two of them was statistically proven. The clusters characteristics were consistent with Zimmerman's

findings [12] about the factors that are critical in cross-project prediction. In our paper, we make a step towards simplifying the setup of defect prediction in the software development process. The laborious activities regarding calculation of correlation vectors and mining the clusters are not needed as it is obvious from the very beginning to which cluster a project belongs. A subset of the same data set had already been analysed in [16,17]. In [16] 5 industrial and 11 open-source projects were investigated. The study was focused on the role of the size factor in defect prediction. The other paper was focused on the cross-project defect prediction. In [17], published in Polish, being a preliminary study to this one, we focused on the differences and similarities between industrial, open-source and academic projects, whereas in this paper, we additionally performed a comprehensive statistical analysis. As a result, new projects may take advantage of the prediction models developed for the aforementioned classes of the existing projects.

It is also worth mentioning that a different approach, based on the idea of inclusion additional software projects during the training process, can provide a cross-project perspective on software quality modelling and prediction [18].

A comprehensive study of cross-project defect prediction was conducted by He et al. [19]. The authors investigated 10 open source projects to check whether training data from other projects can provide better prediction results than training data from the same project – in the best cases it was possible. Furthermore, in 18 out of 34 cases the authors were able to obtain a Recall greater than 70% and a Precision greater than 50% for the cross-project defect prediction.

### 3. Empirical Evaluation Design

#### 3.1. Data Collection

The data from 83 versions of 35 projects was collected and analyzed. It covers 24 versions of 5 industrial projects, 42 versions of 13 open-source projects and 17 versions of 17 academic projects.

The number of versions is greater than the number of projects because there were projects in which data from several versions were collected. For example, in the case of the Apache Ant project (<http://ant.apache.org>), versions 1.3, 1.4, 1.5, 1.6 and 1.7 were analyzed. For each of the analysed versions there was an external release, which was visible to the customer or user.

Each of the investigated industrial projects is a custom-built enterprise solution. All of the industrial projects have already been successfully developed by different teams of 10 to 40 developers and installed in the customer environments. All of them belong to the insurance domain but implement different feature sets on top of Java-based frameworks. Each of the industrial projects were developed by the same vendor.

The following open-source projects were investigated: Apache Ant, Apache Camel, Apache Forrest, Apache Log4j, Apache Lucene, Apache POI, Apache Synapse, Apache Tomcat, Apache Velocity, Apache Xalan, Apache Xerces, Ckjm and Pbeans.

The academic projects were developed by the fourth and the fifth-year graduate MSc computer science students. The students were divided into the groups of 3, 4 or 5 persons. Each group developed exactly one project. The development process was highly iterative (feature driven development). Each project lasted one year. During the development for each feature UML documentation was prepared. Furthermore, high level of test code coverage was obtained by using the latest testing tools, e.g. JUnit for unit tests, FitNesse for functional tests. The last month of development was used for additional quality assurance and bug fixing; the quality assurance was conducted by an external group of subjects (i.e. not the subjects involved in the development). The projects were from different domains, were built using different frameworks, had different architectures and covered different sets of functionalities, nonetheless all of them were written in Java.

The objects of the measurement were software development products (Java classes). The following software metrics were used in the study:

- Chidamber & Kemerer metrics suite [20]: Weighted Method per Class (WMC), Depth of Inheritance Tree (DIT), Number Of Children (NOC), Coupling Between Object classes (CBO), Response For a Class (RFC) and Lack of Cohesion in Methods (LCOM);
- a metric suggested by Henderson-Sellers [21]: Lack of Cohesion in Methods (LCOM3);
- Martin’s metrics [22]: Afferent Couplings (Ca) and Efferent Couplings (Ce).
- QMOOD metrics suite [23]: Number of Public Methods (NPM), Data Access Metric (DAM), Measure Of Aggregation (MOA), Measure of Functional Abstraction (MFA) and Cohesion Among Methods (CAM);
- quality oriented extension of Chidamber & Kemerer metrics suite [24]: Inheritance Coupling (IC), Coupling Between Methods (CBM) and Average Method Complexity (AMC),
- two metrics which are based on the McCabe’s cyclomatic complexity measure [25]: Maximal Cyclomatic Complexity (Max\_CC) and Average Cyclomatic Complexity (Avg\_CC),
- Lines Of Code (LOC),
- Defects – the dependent variable; in the case of industrial and open source projects the sources of the defect data were testers and end users, in the case of academic projects the source of the defect data were students (not involved in a development of a particular projects) who validated the developed software against the specification during the last month (devoted to testing) of the 1-year project.

Definitions of the metrics listed above can be found in [16]. In order to collect the metrics, we used a tool called Ckjm ([http://gromit.iar.pwr.wroc.pl/p\\_inf/ckjm](http://gromit.iar.pwr.wroc.pl/p_inf/ckjm)). The version of Ckjm employed here was reported earlier by Jureczko and Spinellis [16]). The defect count was collected with a tool called BugInfo (<http://kenai.com/projects/buginfo>). The collected metrics are available online in a Metric Repository (all metrics: <http://purl.org/MarianJureczko/MetricsRepo>, metrics used in this research: [http://purl.org/MarianJureczko/MetricsRepo/IET\\_CrossProjectPrediction](http://purl.org/MarianJureczko/MetricsRepo/IET_CrossProjectPrediction)). Data sets related

to the analyzed software defect prediction models are available from the R package [26] to streamline reproducible research [27, 28].

The employed metrics might be considered as the classic ones. Each of them has been in use for at least several years. Hence, the metrics are well known, already recognized by the industry and have a good tool support. There is a number of other metrics, some of them very promising in the field of defect prediction, e.g. the cognitive, the dynamic and the historical metrics [15, 29, 30]. A promising set of metrics are process metrics analysed by Madeyski and Jureczko [31]. Some of them, like Number of Distinct Committers (NDC) or Number of Modified Lines (NML), can significantly improve defect prediction models based on classic software product metrics [31]. Nevertheless, we decided not to use them, since those metrics are not so popular yet (especially in the industry) and the collecting process could be challenging. One may expect that the limited tool support would result in decreasing the number of investigated projects, which is a crucial factor in cross-project defect prediction. We fully understand and accept the value of using metrics that describe a variety of features. Furthermore, we are involved in the development of a tool which includes support for the historical metrics [32]. We are going to use the tool to collect metrics for future experiments.

### 3.2. Data Analysis Methodology

The empirical evaluation described further was performed to verify whether there is a difference between industrial, open-source and academic projects with regard to defect prediction. Statistical hypotheses were formulated. The defect prediction models were built and applied to the investigated projects. The efficiency of prediction was used to evaluate the models and to verify the hypotheses.

To render that in a formal way, it is necessary to assume that  $E(M, v)$  is the evaluation function. The function assesses the efficiency of prediction of the model  $M$  on the version  $v$  of the investigated software project. Let  $c_1, c_2, \dots, c_n$  be the classes from the version  $v$  in descending

order of predicted defects according to the model  $M$ , and let  $d_1, d_2, \dots, d_n$  be the number of defects in each class.  $D_i$  is the  $\sum(d_1, \dots, d_i)$ , i.e., the total defects in the first  $i$  classes. Let  $k$  be the smallest index so that  $D_k > 0.8 * D_n$ , then  $E(M, v) = k/n * 100\%$ . Such evaluation function has been used as it clearly corresponds with the software projects reality we faced. It is closely related to the quality goal: detect at least 80% of defects. The evaluation function shows how many classes must be tested (in practice it corresponds well to how much effort must be committed) to reach the goal when testing according to prediction model output. The properties of the group of evaluation functions that the one selected by us belongs to have been analysed by Weyuker et al. [33].

The empirical evaluation is defined in a generic way which embraces three investigated classes of software projects. Let  $A, B$  and  $C$  be those classes (i.e. industrial, open-source and academic, respectively). Let us interpret the classes of software projects as the sets of versions of software projects. Let  $A$  be the object of the current empirical evaluation.  $B$  and  $C$  will be investigated in subsequent experiments using the analogous procedure. Let  $a$  be a member of the set  $A$ ,  $a \in A$ . Let  $M_x$  be the defect prediction model which was trained using data from versions that belong to set  $X$ . Specifically, there are  $M_A, M_B, M_C$  and  $M_{B \cup C} (B \cap C = \neg A)$ . Subsequently, the following values were calculated for each  $a \in A$ :

- $E(M_A, a)$  – let us call the set of obtained values  $E_A$ ,
- $E(M_B, a)$  – let us call the set of obtained values  $E_B$ ,
- $E(M_C, a)$  – let us call the set of obtained values  $E_C$ ,
- $E(M_{B \cup C}, a)$  – let us call the set of obtained values  $E_{B \cup C}$ .

The following statistical hypothesis may be formulated with the use of  $E_A, E_B, E_C$  and  $E_{B \cup C}$  sets:

- $H_{0,A,B}$  –  $E_A$  and  $E_B$  come from the same distribution.
- $H_{0,A,C}$  –  $E_A$  and  $E_C$  come from the same distribution.

Table 1. Descriptive statistics of metrics in different projects classes ( $\bar{X}$  – mean;  $s$  – standard deviation;  $r$  – Pearson correlation coefficient; \* – correlation significant at 0,05 level)

	Industrial			Open-source			Academic		
	$\bar{X}$	$s$	$r$	$\bar{X}$	$s$	$r$	$\bar{X}$	$s$	$r$
WMC	5.2	8.6	0.13*	10.5	13.7	0.29*	9.7	10.8	0.38*
DIT	3	1.7	-0.07*	2.1	1.3	-0.01	2.1	1.7	0.29*
NOC	0.6	8.8	0	0.5	3.4	0.03*	0.2	1.5	-0.04
CBO	15.1	20.8	0.26*	10.3	16.9	0.20*	8	8.1	0.25*
RFC	24.7	27.5	0.28*	27.1	33.3	0.34*	26	30.6	0.53*
LCOM	37.6	660.2	0.03*	95.2	532.9	0.19*	62	276.5	0.37*
Ca	2.8	17.7	0.16*	5.1	15.3	0.11*	3.8	6.7	0.19*
Ce	12.3	10.3	0.24*	5.4	7.4	0.26*	4.7	5.9	0.38*
NPM	3.4	7.9	0.08*	8.4	11.5	0.22*	7.4	8.7	0.08*
LCOM3	1.4	0.6	-0.04*	1.1	0.7	-0.07*	1.1	0.6	-0.11*
LOC	170.4	366.7	0.26*	281.3	614.7	0.29*	248.1	463.4	0.53*
DAM	0.2	0.3	0.02*	0.5	0.5	0.06*	0.6	0.5	0.07*
MOA	0.1	1	0.03*	0.8	1.8	0.27*	0.8	1.6	0.27*
MFA	0.6	0.4	-0.06*	0.4	0.4	-0.02*	0.3	0.4	0.16*
CAM	0.6	0.2	-0.13*	0.5	0.3	-0.19*	0.5	0.2	-0.17*
IC	1.1	1.1	-0.04*	0.5	0.8	0.06*	0.3	0.5	-0.03
CBM	1.7	2.4	-0.03*	1.5	3.1	0.10*	0.5	1.5	0.02
AMC	30.4	39.8	0.14*	28.1	80.7	0.07*	21.2	24.7	0.24*
Max_cc	3.3	5.7	0.17*	3.8	7.5	0.17*	3.2	5.5	0.31*
Avg_cc	1.3	1.5	0.13*	1.3	1.1	0.12*	1.1	0.8	0.17
Defects	0.232	0.887		0.738	1.906		0.382	1.013	

- $H_{0,A,BUC} - E_A$  and  $E_{BUC}$  come from the same distribution.

The alternative hypothesis:

- $H_{1,A,B} - E_A$  and  $E_B$  come from different distributions.
- $H_{1,A,C} - E_A$  and  $E_C$  come from different distributions.
- $H_{1,A,BUC} - E_A$  and  $E_{BUC}$  come from different distributions.

When the alternative hypothesis is accepted and  $\text{mean}(EA) < \text{mean}(EX)$ , there is a significant difference in prediction accuracy: the model trained on the data from set  $A$  gives a significantly better prediction than the model trained on the data from set  $X$ . The predictions are made for all the project versions which belong to the set  $A$ . Hence, the data from set  $X$  should not be used to build defect prediction models for the project versions which belong to set  $A$ .

The hypotheses were evaluated by the parametric  $t$ -test for dependent samples. The general assumptions of parametric tests were investigated beforehand. The homogeneity of variance was tested using Levene's test and the normality of

distribution was tested using the Shapiro-Wilk test [34]. All hypotheses were tested on the default significance level:  $\alpha = 0.05$ .

## 4. Experiments and Results

### 4.1. Descriptive Statistics

The three aforementioned classes of software projects, namely: industrial, open-source and academic, were described in Table 1. The description provides information about the mean value and standard deviation of each of the analyzed software metrics. Each metric is calculated per Java class, and the statistics are based on all Java classes in all projects that belong to a given class of projects. Moreover, the correlations with the number of defects were calculated and presented. Most of the size related metrics (WMC, LCOM, LOC, Max\_cc and Avg\_cc) had higher values in open-source projects, while coupling metrics (CBO and Ce) had the greatest values in the industrial projects. In the case of each of the in-

Table 2. The number of classes per project

	Industrial		Open-source		Academic	
	$\bar{X}$	$s$	$\bar{X}$	$s$	$\bar{X}$	$s$
No of classes	2806.3	831.7	302.6	219.4	56.4	58.4

investigated project classes, the RFC metric has a higher correlation coefficient with the number of defects. However, there are major differences in its value (it is 0.28 in the case of the industrial projects, and 0.53 in the case of the academic projects) and in the sets of other metrics that are highly correlated with the number of defects.

Collected data suggest that in all kinds of the projects (industrial, open-source and academic) the mean LOC per class follows the rule of thumb presented by Kan in his book [35] (Table 12.2 in Chapter 12) that LOC per C++ class should be less than 480. A similar value is expected for Java. Bigger classes would suggest poor object-oriented design. Low LOC per class does not mean that the code under examination is small. To give an impression with regard to the size of the investigated projects Table 2 presents numbers of classes per project.

The number of defects per Java class is presented in Table 3. It appears that the number of defects per Java class in industrial and academic projects is close with regards to standard deviation and mean, while open source projects are characterized by higher standard deviation as well as mean. A plausible explanation is that lower standard deviations in academic and industrial projects come from a more homogeneous development environment than in open source projects.

## 4.2. Empirical Evaluation Results

The empirical evaluation has been conducted three times. Each time a different class of software projects was used as the object of study.

### 4.2.1. Industrial Projects

The descriptive statistics are summarized in Table 4. The models that were trained on the data from the industrial projects ('Industrial models'

Table 3. The number of defects per Java class

Type	$\bar{X}$	$s$
Open-source	0.7384	1.9
Industrial	0.2323	0.9
Academic	0.3816	1.0

in Table 4) gave the most accurate prediction. The predictions were made only for the industrial projects. Furthermore, the mean value of the evaluation function  $E(M, v)$  was equal to 50.82 in the case of the 'Industrial models.' In the case of the other models, the mean values equaled 53.96, 55.38 and 73.59. A smaller value of the evaluation function implies better predictions.

The predictions obtained from the models trained on the data from the industrial projects were compared with the predictions from the other models. The predictions were made only for the industrial projects. The difference was statistically significant only in the case of comparison with the predictions from models trained on the data from the academic projects (see Table 5). In this case the calculated effect size  $d = 1.56$  is extremely high (according to magnitude labels proposed by Cohen  $d$  effect size equal to 0.2 is considered small, equal to 0.5 is considered medium, while equal to 0.8 is considered high) while the power of a test (i.e., the probability of rejecting  $H_0$  when in fact it is false) is equal to 1. This important finding shows that there exist significant differences between industrial and academic software projects with respect to defect prediction.

### 4.2.2. Open-source Projects

The descriptive statistics of the results of applying defect prediction models to open-source projects are presented in Table 6. The models, which were trained on the data from the open-source projects ('Open-source models' in Table 6), gave the most accurate prediction.

Table 7 shows the  $t$ -test statistics, power and effect size calculations for the open-source projects.

In all of the cases  $p$ -value is lower than 0.05. However, instead of coming to the conclusions

Table 4. Models evaluations for the industrial projects

Model	Non-industrial	Open-source	Academic	Industrial
$\bar{X}$	53.96	55.38	73.59	50.82
s	13.29	12.01	9.68	9.86

Table 5. Dependent samples  $t$ -test for the industrial projects

	$H_{0,ind,open \cup acad}$	$H_{0,ind,open}$	$H_{0,ind,acad}$
$t, df = 23$	-.979	-1.482	-7.637
$p$	0.338	0.152	0.000
effect size $d$	0.200	0.302	1.559
power	0.244	0.418	1

Table 6. Models evaluations for the open-source projects

Model	Industrial	Non-open-source	Academic	Open-source
$\bar{X}$	57.67	57.26	65.17	54.00
s	19.22	18.02	14.31	16.66

now we suggests performing the Bonferroni correction (explained in Section 4.2.4) beforehand. The calculated effect sizes are between medium and small in the first two cases, while medium to large in the last case which is an important finding suggesting serious differences between open source and academic projects with respect to defect prediction. The power of a test is calculated as well very high (close to 1) probability of rejecting  $H_0$  is suggested when in fact it is false.

#### 4.2.3. Academic Projects

The descriptive statistics of the results of applying defect prediction models to academic projects are presented in Table 8. The obtained results are surprising. The ‘Academic models’ gave almost the worst predictions. Slightly worse were only the ‘Industrial models’, whilst the ‘Open-source models’ and ‘Non-academic models’ gave definitely better predictions.

The analysis presented in Table 9 is based on the  $t$ -test for dependent samples. The predictions obtained from the models which were trained on the data from the academic projects, were compared with predictions from the other models. The predictions were made only for the academic

projects. The differences were not statistically significant, the effect sizes were below small (in the first and third case) and between small and medium in the second case.

#### 4.2.4. Bonferroni Correction

Since several different hypotheses were tested on the same data set, the following kinds of errors were likely to occur: errors in inference, including confidence intervals which fail to include their corresponding population parameters, or hypothesis tests that incorrectly reject the null hypothesis. Among several statistical techniques that have been developed to prevent such instances is the Bonferroni correction. The correction is based on the idea that if  $n$  dependent or independent hypotheses are being tested on a data set, then one way of maintaining the familywise error rate is to test each individual hypothesis at a statistical significance level of  $1/n$  times. In our case  $n = 9$  as there are nine different hypotheses. Hence, the significance level should be decreased to  $0.05/9 = 0.0055$ . Consequently, the  $H_{0,ind,acad}$  and  $H_{0,open,acad}$  hypotheses will be rejected but  $H_{0,open,ind}$  and  $H_{0,open,ind \cup acad}$  hypotheses will not be rejected.

Table 7. Dependent samples *t*-test for the open-source projects

	$H_{0,open,ind}$	$H_{0,open,ind\cup acad}$	$H_{0,open,acad}$
$t, df = 41$	-2.363	-2.193	-4.325
$p$	0.023	0.034	0.000
effect size $d$	0.365	0.338	0.667
power	0.752	0.695	0.995

Table 8. Models evaluations for the academic projects

Model	Industrial	Open-source	Non-academic	Academic
$\bar{X}$	56.34	50.60	53.19	55.02
s	20.71	15.56	18.54	20.21

Table 9. Dependent samples *t*-test for the academic projects

	$H_{0,acad,ind}$	$H_{0,acad,open}$	$H_{0,acad,ind\cup open}$
$t, df = 16$	0.312	-1.484	-0.696
$p$	0.759	0.157	0.496
effect size $d$	0.076	0.360	0.169
power	0.009	0.412	0.164

### 4.3. Which Metrics are Relevant?

There is some evidence that the three investigated code ownership models differ with respect to defect prediction. Therefore, it could be helpful for further research to identify the casual relations that drive those differences. Unfortunately, the scope of software metrics (independent variables of the defect prediction models) does not cover many aspects that may be a direct cause of defects. Let us assume that there is an inexperienced developer who gets a requirement to implement and he does something wrong, he introduces a defect into the system. The true cause of the defect is a composition of several factors including the developer's experience, requirement complexity and the level of maintainability of the parts of the system that were changed by the developer. Only a fraction of the aforementioned factors can be covered by the metrics available from software repositories and hence many of them must be ignored by the defect prediction models. Taking into consideration the above arguments we decided not to define the casual relations upfront, but investigate what emerges from the models we obtained. Since it does not follow the commonly

used procedure (start with a theory and then look for confirmation in empirical data), it is important to keep in mind that results of such analysis do not indicate casual relations, but only a coexistence of some phenomena.

The analysis is based on the relevancy of particular metrics in models obtained for different code ownership types. The defect prediction model has the following form:

$$ExpectedNumberOfDefects = a_1 * M_1 + a_2 * M_2 \dots$$

where  $a_i$  represents coefficients obtained from regression,  $M_i$  software metrics (independent variables in the prediction). For each metric we calculated its importance factor (the factors are calculated for each code ownership model respectively) using the following form:

$$IF_{M_i} = \frac{a_i * \bar{M}_i}{\sum_j |a_j * \bar{M}_j|}$$

where  $\bar{M}_i$  is the average value of metric  $M_i$  in the type of code ownership for which the factor is calculated (the averages are reported in Tab. 1). The above definition results in a factor that shows for

given metric what part of the prediction model output is driven by the metric and additionally preserves the sign of the metric’s contribution. The obtained values of importance factors are reported in Tab. 10.

Table 10. Importance of metrics in different code ownership models.

Model	Industrial	Open-source	Academic
WMC	-0.06	0	0
DIT	-0.08	0	0.08
NOC	-0.01	0	0
CBO	0.17	0	-0.36
RFC	0.19	0.35	0.04
LCOM	0	0	0
Ca	0	0.05	0.17
Ce	0	0.17	0.20
NPM	0	0	0.01
LCOM3	0.13	0	-0.05
LOC	0.05	0.14	0
DAM	-0.01	-0.17	-0.04
MOA	0	0.12	0
MFA	0.05	0	-0.02
CAM	-0.16	0	0
IC	0	0	-0.02
CBM	-0.03	0	0.01
AMC	-0.04	0	0
Max_cc	0.01	0	0
Avg_cc	0	0	0

The importance factors show that there are similarities as well as differences between the prediction models trained for different types of code ownership. In all of them an important role is played by the RFC metric and all of them take into consideration coupling related metrics. However, in the case of academic projects these are Ca and Ce, in the case of open-source Ce is much more important than Ca and for industrial projects only CBO matters. There are also metrics with positive contribution in only one type of projects (positive contribution means that the metric value grows with the number of expected defects), i.e. LCOM3 for industrial projects (as well as the mentioned earlier CBO), MOA and LOC for the open-source projects, DIT for the academic ones. More significant differences have been observed with regard to negative contribution. The greatest negative contribution for

academic projects have CBO and LCOM3, for open-source DAM and for industrial CAM.

## 5. Threats to Validity

### 5.1. Construct Validity

Threats to construct validity refer to the extent to which the measures accurately reflect the theoretical concepts they are intended to measure [34]. The mono-method bias reflects the risk of a single means of recording measures. As a result, an important construct validity threat is that we cannot guarantee that all the links between bugs and versioning system files, and, subsequently, classes, are retrieved (e.g. when there is no bug reference identifier in the commit comment), as bugs are identified according to the comments in the source code version control system. In fact, this is a widely known problem and the method that we adopted is not only broadly used but also represents the state of the art with respect to linking bugs to versioning system files and classes [36, 37].

A closely related threat concerns anonymous inner classes. We cannot distinguish whether a bug is related to anonymous inner classes or their containing class, due to the file-based nature of source code version control systems. Hence, it is a common practice not to take into consideration the inner classes [38–40]. Fortunately, the inner classes usually constitute a small portion of all classes (in our study it was 8.84%).

Furthermore, the guidelines of commenting bugfixes may vary among different projects. Therefore, it is possible that the interpretation of the term bug is not unique among the investigated projects. Antoniol et al. [41] showed that a fraction of issues marked as bugs are problems unrelated to corrective maintenance. We did our best to remove such occurrences manually but in future research we plan to apply the suggestion by Antoniol et al. to filter the non-bug issues out.

It is also worth mentioning that it was not possible to track operations like changing the class name or moving the class between pack-



ages. Therefore, after such a change, the class is interpreted as a new one.

## 5.2. Statistical Conclusion Validity

Threats to statistical conclusion validity relate to the issues that affect the validity of inferences. In our study we used robust statistical tools: SPSS and Statistica.

## 5.3. Internal Validity

The threats to internal validity concern the true causes (e.g., external factors) that may affect the outcomes observed in the study. The external factor we are aware of is the human factor pointed out by D'Ambros et al. [38]. D'Ambros et al. decided to limit the human factor as far as possible and chose not to consider bug severity as a weight factor when evaluating the number of defects. We decided to follow this approach as Ostrand et al. reported how those severity ratings are highly subjective and inaccurate [42].

Unfortunately, each of the investigated clusters (i.e. code ownership models) has limited variability. All academic projects were developed at the same university. However, they differ a lot with respect to requirements and architecture. Only two open-source projects (PBeans and ckjm) do not come from Apache and all of them can be classified as a tool or library what is in opposition to industrial projects which are enterprise solutions that employ database systems. Furthermore, all the industrial projects were developed by the same vendor which poses a major threat to external validity. In consequence, it is possible to define an alternative hypothesis that explains the differences between clusters, e.g. the open-source cluster can be redefined into tools & libraries, while the industrial cluster can be redefined into enterprise database oriented solution and then a hypothesis that regards difference between such clusters may be formulated. Unfortunately, those alternative hypotheses cannot be invalidated without additional projects and even with additional projects we cannot avoid further alternative hypotheses with more fancy definitions of cluster boundaries. The root cause

of this issue is the sample selection procedure which does not guarantee random selection. Only a small part of the population of software projects is available for researchers and collecting data for an experiment is a huge challenge taking into account how difficult is to get access to the source code or software metrics of real, industrial projects. We were addressing this issue by taking into consideration the greatest possible number of projects we were able to cover with a common set of software metrics. It does not solve the issue, but reduces the risk of accepting wrong hypothesis due to some data constellations that are a consequence of sample selection.

## 5.4. External Validity

The threats to external validity refer to the generalization of research findings. Fortunately, in our study we considered a wide range of different kinds of projects. They represent different ownership models (industrial, open source and academic), belong to different application domains and have been developed according to different software development processes. However, our selection of projects is by no means representative and that poses a major threat to external validity. For example, we only considered software projects developed in the Java programming language. Fortunately, thanks to this limitation all the code metrics are defined identically for each system, so we have alleviated the parsing bias.

## 6. Conclusions

Our study has compared three classes of software projects (industrial, open-source and academic) with regard to defect prediction. The analysis comprised the data collected from 24 versions of 5 industrial projects, 42 versions of 13 open-source projects, and 17 versions of 17 academic projects. In order to identify differences among the classes of software projects listed above, defect prediction models were created and applied. Each of the software project classes was investigated through verifying three statistical hypotheses. The following two noteworthy findings

were identified: two of the investigated hypotheses were rejected:  $H_{0,ind,acad}$  and  $H_{0,open,acad}$ . In the case of  $H_{0,open,ind \cup acad}$  and  $H_{0,open,ind}$   $p$ -values were below 0.05, but the hypotheses can not be rejected due to the Bonferroni correction. Such results are not conclusive due to threats to external validity discussed in Section 5.4, as well as the possibility that even small changes in the input data may change the decision regarding hypothesis rejection in both directions and thus we encourage further investigation.

As a result, we obtained some evidence that the open-source, industrial and academic projects may be treated as separate classes of projects with regard to defects prediction. In consequence, we do not recommend using models trained on the projects from different code ownership model, e.g. making predictions for an industrial project with a model trained on academic or open-source projects. Of course the investigated classes (i.e. academic, industrial and open-source) may not be optimal and smaller classes could be identified in order to increase the prediction performance. Identification of smaller projects classes constitutes a promising direction for further research.

The prediction models trained for each of the investigated classes of projects were further analysed in order to reveal key differences between them. Let us focus on the differences between open-sources and industrial ones as we have very limited evidence to support the thesis that academic projects constitutes a solid class of projects with respect to defect prediction. However, the analysis revealed an almost self-explaining fact with regard to this class of projects. Namely, the deeper the inheritance tree the more likely it is that a student will introduce a defect in such a class. Typically, API of a class with a number of ancestors is spread among the parent classes and thus a developer that looks at only some of them may not understand the overall concept and introduce changes that are in conflict with the source code of one of the other classes. In consequence, we may expect that inexperienced developer, e.g. a student, will miss some important details.

The differences between open-source and industrial projects can be explained by the so called

crowd-driven software development model commonly used in open-source projects. High value of the model output in those projects is mainly driven by the LOC and MOA metrics. The first of them simply represents the size of a class and it is not surprising that it could be challenging to understand a big class for someone who commits to a project only occasionally. Furthermore, the MOA metric can be considered in terms of the number of classes that must be known and understood to effectively work with a given one, which creates additional challenges for developers that do not work in a project in a daily manner. There also is the negative contribution of the DAM metrics which also fits well to the picture as high values of this metric correspond with low number of public attributes and thus narrows the scope of source code that a developer should be familiar with. Both, industrial and open-source models use the RFC metric, however, in the case of open-source its more relevant which also supports the aforementioned hypothesis regarding crowd-driven development. The industrial projects are usually developed by people who know the project under development very well. That does not mean that everyone knows everything about each class. When it is necessary to be familiar with a number of different classes to make a single change in the project it is still likely to introduce a defect. However, in the case of industrial projects the effect is not so strong. Furthermore, the industrial prediction model uses metrics that regard flaws in the class internal structure, i.e. LCOM3 and CAM), which make challenges in the development regardless of developer knowledge about other classes.

The models that were trained on the academic projects usually gave the worst predictions. Even in the case of making predictions for academic projects, the models trained on the academic projects did not perform well. It was not the primary goal of the study, but the obtained results made it possible to arrive at that newsworthy conclusion. The academic projects are not good as a training set for the defect prediction models. Probably, they are too immature and thus have too chaotic structure. The obtained results point to the need of reconsidering the relevancy of the

studies on defect prediction that rely solely on the data from the academic projects. Academic data sets were used even in the frequently cited [43–45] and recently conducted [46, 47] studies.

Detailed data are available via a web-based metrics repository established by the authors (<http://purl.org/MarianJureczko/MetricsRepo>). The collected data may be used by other researchers for replicating presented here experiments as well as conducting own empirical studies. The obtained defect prediction models related to the conducted empirical study presented in this paper are available online ([http://purl.org/MarianJureczko/IET\\_CrossProjectPrediction](http://purl.org/MarianJureczko/IET_CrossProjectPrediction)). However, we recommend using the models for cross-project defect prediction with great caution, since the obtained prediction performance is moderate and presumable in most cases can be surpassed by a project specific model.

## References

- [1] L. Briand, W. Melo, and J. Wust, “Assessing the applicability of fault-proneness models across object-oriented software projects,” *IEEE Transactions on Software Engineering*, Vol. 28, No. 7, 2002, pp. 706–720.
- [2] L. Samuelis, “On principles of software engineering-role of the inductive inference,” *e-Informatica Software Engineering Journal*, Vol. 6, No. 1, 2012, pp. 71–77.
- [3] L. Fernandez, P.J. Lara, and J.J. Cuadrado, “Efficient software quality assurance approaches oriented to UML models in real life,” *Idea Group Publishing*, 2007, pp. 385–426.
- [4] M.L. Hutcheson and L. Marnie, *Software testing fundamentals*. John Wiley & Sons, 2003.
- [5] B.W. Boehm, “Understanding and controlling software costs,” *Journal of Parametrics*, Vol. 8, No. 1, 1988, pp. 32–68.
- [6] G. Denaro and M. Pezzè, “An empirical evaluation of fault-proneness models,” in *Proceedings of the 24rd International Conference on Software Engineering*. IEEE, 2002, pp. 241–251.
- [7] C. Kaner and W.P. Bond, “Software engineering metrics: What do they measure and how do we know?” in *10th International Software Metrics Symposium*. IEEE, 2004, p. 6.
- [8] N.E. Fenton and M. Neil, “Software metrics: successes, failures and new directions,” *Journal of Systems and Software*, Vol. 47, No. 2, 1999, pp. 149–157.
- [9] T. Hall and N. Fenton, “Implementing effective software metrics programs,” *IEEE Software*, Vol. 14, No. 2, 1997, pp. 55–65.
- [10] B. Turhan, T. Menzies, A.B. Bener, and J. Di Stefano, “On the relative value of cross-company and within-company data for defect prediction,” *Empirical Software Engineering*, Vol. 14, No. 5, 2009, pp. 540–578.
- [11] M.T. Villalba, L. Fernández-Sanz, and J. Martínez, “Empirical support for the generation of domain-oriented quality models,” *IET software*, Vol. 4, No. 1, 2010, pp. 1–14.
- [12] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, “Cross-project defect prediction: a large scale experiment on data vs. domain vs. process,” in *Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*. ACM, 2009, pp. 91–100.
- [13] M. Jureczko and L. Madeyski, “Towards identifying software project clusters with regard to defect prediction,” in *Proceedings of the 6th International Conference on Predictive Models in Software Engineering*. ACM, 2010, p. 9.
- [14] R. Subramanyam and M.S. Krishnan, “Empirical analysis of CK metrics for object-oriented design complexity: Implications for software defects,” *IEEE Transactions on Software Engineering*, Vol. 29, No. 4, 2003, pp. 297–310.
- [15] N. Nagappan, T. Ball, and A. Zeller, “Mining metrics to predict component failures,” in *Proceedings of the 28th international conference on Software engineering*. ACM, 2006, pp. 452–461.
- [16] M. Jureczko and D. Spinellis, “Using object-oriented design metrics to predict software defects,” in *Models and Methods of System Dependability*. Oficyna Wydawnicza Politechniki Wrocławskiej, 2010, pp. 69–81.
- [17] M. Jureczko and L. Madeyski, “Predykcja defektów na podstawie metryk oprogramowania – identyfikacja klas projektów,” in *Proceedings of the Krajowa Konferencja Inżynierii Oprogramowania (KKIO 2010)*. PWNT, 2010, pp. 185–192.
- [18] Y. Liu, T.M. Khoshgoftaar, and N. Seliya, “Evolutionary optimization of software quality modeling with multiple repositories,” *IEEE Transactions on Software Engineering*, Vol. 36, No. 6, 2010, pp. 852–864.
- [19] Z. He, F. Shu, Y. Yang, M. Li, and Q. Wang, “An investigation on the feasibility of cross-project

- defect prediction,” *Automated Software Engineering*, Vol. 19, No. 2, 2012, pp. 167–199.
- [20] S.R. Chidamber and C.F. Kemerer, “A metrics suite for object oriented design,” *IEEE Transactions on Software Engineering*, Vol. 20, No. 6, 1994, pp. 476–493.
- [21] B.H. Sellers, *Object-Oriented Metrics. Measures of Complexity*. Prentice Hall, 1996.
- [22] R. Martin, “OO design quality metrics – an analysis of dependencies,” in *Proc. Workshop Pragmatic and Theoretical Directions in Object-Oriented Software Metrics*, 1994.
- [23] J. Bansiya and C.G. Davis, “A hierarchical model for object-oriented design quality assessment,” *IEEE Transactions on Software Engineering*, Vol. 28, No. 1, 2002, pp. 4–17.
- [24] M.H. Tang, M.H. Kao, and M.H. Chen, “An empirical study on object-oriented metrics,” in *Proceedings of the Sixth International Software Metrics Symposium*. IEEE, 1999, pp. 242–249.
- [25] T.J. McCabe, “A complexity measure,” *IEEE Transactions on Software Engineering*, No. 4, 1976, pp. 308–320.
- [26] L. Madeyski, *reproducer: Reproduce Statistical Analyses and Meta-Analyses*, 2015, R package. [Online]. <http://CRAN.R-project.org/package=reproducer>
- [27] L. Madeyski and B.A. Kitchenham, “Reproducible Research – What, Why and How,” Wroclaw University of Technology, PRE W08/2015/P-020, 2015.
- [28] L. Madeyski, B.A. Kitchenham, and S.L. Pfleeger, “Why Reproducible Research is Beneficial for Security Research,” (*under review*), 2015.
- [29] J.K. Chhabra and V. Gupta, “A survey of dynamic software metrics,” *Journal of Computer Science and Technology*, Vol. 25, No. 5, 2010, pp. 1016–1029.
- [30] S. Misra, M. Koyuncu, M. Crasso, C. Mateos, and A. Zunino, “A suite of cognitive complexity metrics,” in *Computational Science and Its Applications–ICCSA*. Springer, 2012, pp. 234–247.
- [31] L. Madeyski and M. Jureczko, “Which Process Metrics Can Significantly Improve Defect Prediction Models? An Empirical Study,” *Software Quality Journal*, Vol. 23, No. 3, 2015, pp. 393–422. [Online]. <http://dx.doi.org/10.1007/s11219-014-9241-7>
- [32] M. Jureczko and J. Magott, “QualitySpy: a framework for monitoring software development processes,” *Journal of Theoretical and Applied Computer Science*, Vol. 6, No. 1, 2012.
- [33] E.J. Weyuker, T.J. Ostrand, and R.M. Bell, “Comparing the effectiveness of several modeling methods for fault prediction,” *Empirical Software Engineering*, Vol. 15, No. 3, 2010, pp. 277–295.
- [34] L. Madeyski, *Test-driven development: An empirical evaluation of agile practice*. Springer, 2010.
- [35] S.H. Kan, *Metrics and models in software quality engineering*. Addison-Wesley Longman Publishing Co., Inc., 2002.
- [36] M. Fischer, M. Pinzger, and H. Gall, “Populating a release history database from version control and bug tracking systems,” in *Proceedings International Conference on Software Maintenance*. IEEE, 2003, pp. 23–32.
- [37] T. Zimmermann, R. Premraj, and A. Zeller, “Predicting defects for eclipse,” in *International Workshop on Predictor Models in Software Engineering*. IEEE, 2007, pp. 9–9.
- [38] M. D’Ambros, A. Bacchelli, and M. Lanza, “On the impact of design flaws on software defects,” in *10th International Conference on Quality Software (QSIC)*. IEEE, 2010, pp. 23–31.
- [39] M. D’Ambros, M. Lanza, and R. Robbes, “An extensive comparison of bug prediction approaches,” in *7th IEEE Working Conference on Mining Software Repositories (MSR)*. IEEE, 2010, pp. 31–41.
- [40] A. Bacchelli, M. D’Ambros, and M. Lanza, “Are popular classes more defect prone?” in *Fundamental Approaches to Software Engineering*. Springer, 2010, pp. 59–73.
- [41] G. Antoniol, K. Ayari, M. Di Penta, F. Khomh, and Y.G. Guéhéneuc, “Is it a bug or an enhancement?: a text-based approach to classify change requests,” in *Proceedings of the 2008 Conference of the Center for Advanced Studies on Collaborative Research: Meeting of Minds*. ACM, 2008.
- [42] T.J. Ostrand, E.J. Weyuker, and R.M. Bell, “Where the bugs are,” in *ACM SIGSOFT Software Engineering Notes*, Vol. 29, No. 4. ACM, 2004, pp. 86–96.
- [43] V.R. Basili, L.C. Briand, and W.L. Melo, “A validation of object-oriented design metrics as quality indicators,” *IEEE Transactions on Software Engineering*, Vol. 22, No. 10, 1996, pp. 751–761.
- [44] F. Brito e Abreu and W. Melo, “Evaluating the impact of object-oriented design on software quality,” in *Proceedings of the 3rd International Software Metrics Symposium*. IEEE, 1996.
- [45] W.L. Melo, L. Briand, and V.R. Basili, “Measuring the impact of reuse on quality and productivity in object-oriented

- systems,” Univ. of Maryland, Dep. of Computer Science, College Park, MD, USA 20742, Tech. Rep., 1995. [Online]. <http://drum.lib.umd.edu/bitstream/handle/1903/686/CS-TR-3395.pdf?sequence=2>
- [46] K. Aggarwal, Y. Singh, A. Kaur, and R. Malhotra, “Empirical study of object-oriented metrics,” *Journal of Object Technology*, Vol. 5, No. 8, 2006, pp. 149–173.
- [47] P. Martenka and B. Walter, “Hierarchical model for evaluating software design quality,” *e-Informatica Software Engineering Journal*, Vol. 4, No. 1, 2010, pp. 21–30.



# Resolving Conflict and Dependency in Refactoring to a Desired Design

Iman Hemati Moghadam\*, Mel Ó Cinnéide\*\*

\**Department of Computer Science, University College London, United Kingdom*

\*\**School of Computer Science and Informatics, University College Dublin, Ireland*

I.moghadam@ucl.ac.uk, mel.ocinneide@ucd.ie

## Abstract

Refactoring is performed to improve software quality while leaving the behaviour of the system unchanged. In practice there are many opportunities for refactoring, however, due to conflicts and dependencies between refactorings, only certain orders of refactorings are applicable. Selecting and ordering an appropriate set of refactorings is a challenging task for a developer. We propose a novel automated approach to scheduling refactorings according to their conflicts and dependencies as well as their contribution to design quality expressed by a desired design. The desired design is an improved version of the current program design, and is produced by the developer. Our approach is capable of detecting conflicts and dependencies between refactorings, and uses a sequence alignment algorithm to identify the degree of similarity between two program designs expressed as sequence of characters, thereby measuring the contribution of a refactoring to achieving the desired design. We evaluated our approach on several sample programs and one non-trivial open source application. Our results demonstrate the ability of the approach to order the input refactorings so as to achieve the desired design even in the presence of intense inter-refactoring conflict and dependency, and when applied to a medium-sized, real-world application.

**Keywords:** refactoring, refactoring scheduling, design similarity

## 1. Introduction

Refactoring is performed to improve the quality of the software in some way. It may involve floss refactoring, where minor improvements are applied frequently, typically several times a day, or it may involve remedial refactoring<sup>1</sup> where a more significant design overhaul takes place [1]. In this paper we are concerned with automated refactoring support for the remedial refactoring scenario. A developer performing remedial refactoring typically has a notion of a desired design that they are refactoring the program towards. This desired design may

come about by way of an interactive design process, as in the work of Simons et al. [3, 4] or it may be created by the intellectual effort of the developer [5,6]. Either way, the challenge the developer faces is that of refactoring the program from its current design to its new, desired design.

In earlier work, we presented an approach to refactor a program based both on its desired design and on its source code [5]. In this work, a new UML-based desired design is first created by the developer based on the current software design and their understanding of how it may be required to evolve. The resulting design is then compared with the original one using a differenc-

---

<sup>1</sup> Termed ‘root canal’ refactoring by Murphy-Hill and Black [1] and ‘batch mode’ refactoring by Liu et al. [2]. Liu et al. provide strong evidence of the practical importance of this type of refactoring.

ing algorithm [7], and the detected differences are expressed as refactoring instances. The original source code is then refactored using a heuristic approach based on the detected refactorings to conform more closely to the desired design [5]. Overall, the process of refactoring the program to comply with its desired design involves three distinct steps as follows:

1. The developer must decide what refactorings are required to bring the program from its current design to its desired design.
2. They must decide in what order the refactorings should be applied.
3. The refactorings must then be applied to the program in this sequence.

As mentioned, recent work has sought to automate this refactoring process. For example, UMLDiff [7,8] is a tool that addresses step (1) by detecting what refactorings are required to bring a program design from its current state to a new desired design. Step (3) is supported by a broad range of refactoring tools that apply individual refactorings, such as the Eclipse Refactoring Tools, and also by more sophisticated research prototypes, such as *Code-Imp*, that can apply a series of refactorings guided by a fitness function [9]. The focus of this paper however, is step (2), the ordering of the refactorings into a valid sequence.

Given a set of refactorings, finding a valid sequence in which they may be applied is a non-trivial problem. A refactoring is characterised by a precondition and a postcondition. The precondition determines if the refactoring may be applied, and the postcondition states what the result of applying the refactoring is, assuming its precondition was true when it was applied. A refactoring may be applicable to the initial program, but if it is not applied then, another refactoring in the sequence may render it inapplicable. Conversely, a refactoring may be inapplicable to the initial program, but another refactoring in the sequence may render it applicable later on. These observations have led to the notions of *conflict* and *dependency* in a refactoring sequences [10,11]. Two refactorings are in *conflict* if they cannot both be applied to the program, e.g. a method cannot be moved

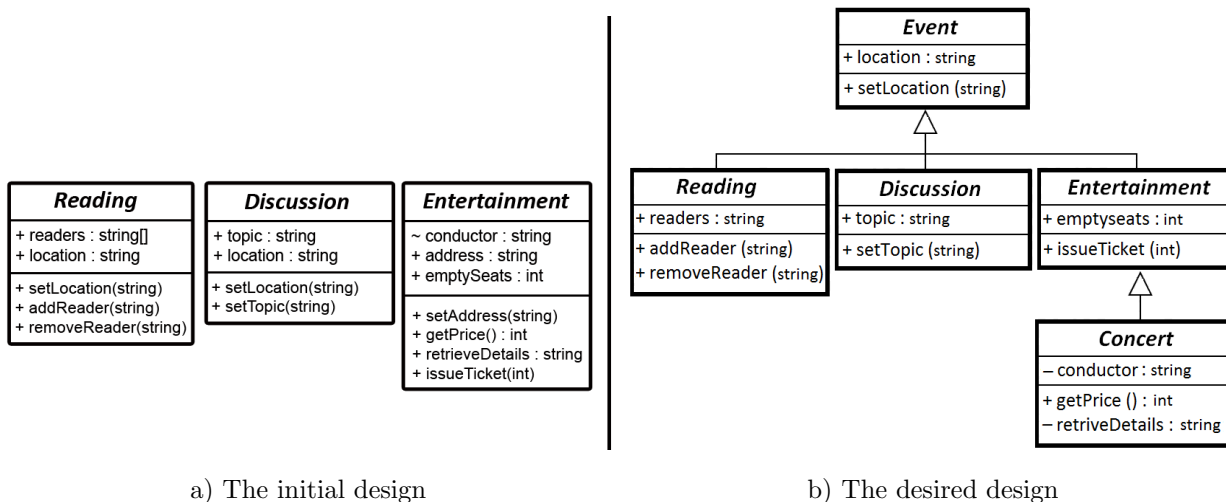
to a deleted class. A *dependency* exists between two refactorings if they can only be applied in a particular order, e.g. a refactoring that creates a new class must be executed before a refactoring that moves a method to that new class. Precise definitions of these terms are provided in Section 3.2.

The question addressed in this paper then is as follows. Given a set of proposed refactorings that are to be applied to a program so as to bring its design to a desired state, how can the refactorings be ordered such that they can be applied to the program while respecting the constraints imposed by the conflicts and dependencies that may exist between the refactorings in the set? To answer this question, we propose an automated refactoring scheduling approach that finds a valid order of the refactorings in the set, according to their conflict and dependency relationships as well as their contribution to achieving the desired design. The main contribution of this paper is two-fold:

- We extend the refactoring scheduling algorithm proposed by Liu et al. [10] by considering not just *conflicts* between refactorings but also *dependencies*. Furthermore, we take into account a type of refactoring conflict not handled in the work of Liu et al., where the application of one refactoring violates the precondition of another. We refer to this algorithm as *REDaCT* (REfactoring Dependency and Conflict).
- We develop the idea of refactoring to a *desired design*, introduced by the authors in earlier work [5], and show how it can be used to guide the refactoring process more effectively. In particular we measure not only the effect of a refactoring in terms of its direct contribution to achieving the desired design, but also its indirect contribution in terms of the refactorings it enables and disables. This extension to the *REDaCT* algorithm is referred to as *REDaCT+*.

The remainder of this paper is structured as follows. Section 2 presents a motivating example to illustrate the necessity of scheduling refactorings. In Section 3 we deal with preliminaries by providing a brief description of the software tool



Figure 1. UML class diagrams of an *Event* application

upon which our approach is based, *Design-Imp*, and defining precisely our notions of conflict and dependency. The proposed scheduling approach to find a valid order between the set of refactorings according to their conflict and dependency relationships, REDaCT, is explained in Section 4, while in Section 5 the REDaCT+ algorithm is presented which extends REDaCT by considering the direct and indirect contribution of each refactoring to achieving the desired design. In Section 6 the REDaCT and REDaCT+ algorithms are evaluated on a number of examples. A survey of related work is presented in Section 7, while in Section 8 we conclude the paper and provide some suggestions for future work.

## 2. Motivating Example

Consider as a motivating example, the simplified UML class diagrams shown in Figure 1. The design in Figure 1a represents the original design while the design in Figure 1b represents the desired design that the developer would like the program to have. We applied the design differencing approach proposed by the authors in earlier work [5] that takes as input two UML class diagrams and then uses a UML design differencing algorithm to find differences between the designs and categorises these as refactoring instances. In other words, the approach returns

the refactorings that are required to bring a program design from its current state to a new desired design. In this example, the desired design is achieved after applying 15 refactorings to the initial design as follows:

**R<sub>1</sub>, R<sub>2</sub>**: Two classes, *Event* and *Concert*, are added to the design using *Extract Hierarchy* and *Extract Subclass* refactorings respectively.

**R<sub>3</sub>, R<sub>4</sub>**: Field *address* and method *setAddress*, both in class *Entertainment*, are renamed to *location* and *setLocation* using *Rename Field* and *Method* refactorings respectively. These refactorings prepare the application of refactorings *R<sub>7</sub>* and *R<sub>10</sub>* described below.

**R<sub>5</sub>, R<sub>6</sub>, R<sub>7</sub>**: The *location* fields in classes *Discussion* and *Reading* and the field *address* in the class *Entertainment* are pulled up to the class *Event* using three separate *Pull Up Field* refactorings.

**R<sub>8</sub>, R<sub>9</sub>, R<sub>10</sub>**: The *setLocation* methods in classes *Discussion* and *Reading* and the method *setAddress* in the class *Entertainment* are pulled up to the class *Event* using three separate *Pull Up Method* refactorings. Refactorings *R<sub>5</sub>* to *R<sub>10</sub>* reduce code duplication and improve readability.

**R<sub>11</sub>, R<sub>12</sub>, R<sub>13</sub>**: Two methods *getPrice* and *retrieveDetails* as well as the field *conductor*, all defined in the class *Entertainment*, are pushed down to the class *Concert* using two separate *Push Down Method* refactorings and one *Push Down Field* refactoring respectively. The motivation for these refactorings is to move features

that are used only in some instances of the original class.

**$R_{14}$ ,  $R_{15}$ :** To simplify the interface and improve understandability, the method *retrieveDetails* and the field *conductor* are made more private using the *Decrease Method Accessibility* and *Decrease Field Accessibility* refactorings respectively.

The aim is to find an order between the aforementioned refactorings that, while requiring the minimum effort, results in the desired design from the initial design. However, because of interdependencies between the refactorings, only some specific refactoring orders are applicable to the initial design. A list of interdependencies between the aforementioned refactorings is as below:

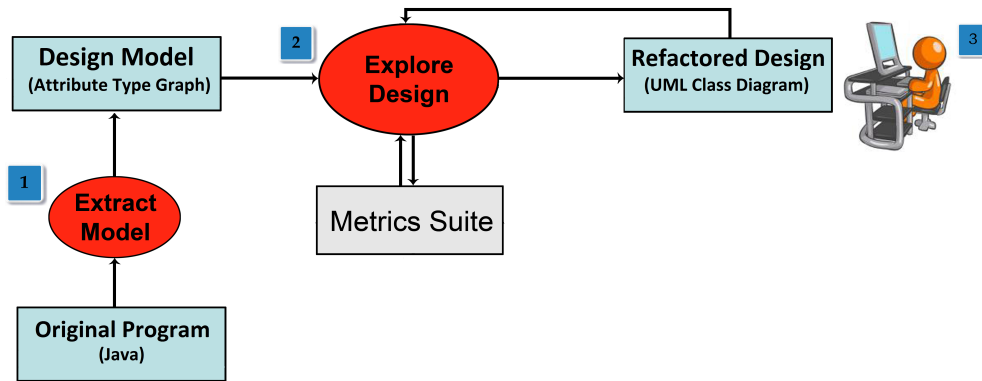
- Refactorings  $R_3$  to  $R_{13}$  are directly dependent on  $R_1$ , or  $R_2$ , i.e. a method or field cannot be moved to a class if the target class has not been created yet.
- The *setLocation* methods in the class *Discussion* use the field *location* of the local class. Therefore, *Pull Up setLocation* ( $R_8$ ) should be applied to the design after *Pull Up location* ( $R_5$ ). Were the method to be pulled up before the field, it would also be necessary to add an instance of the local class as parameter to the method, resulting in a method with two input parameters, which differs from the corresponding method in the desired design.
- For the same reason, two *Pull Up Method* refactorings,  $R_9$  and  $R_{10}$ , should be applied to the design only after their corresponding *Pull Up Field* refactorings,  $R_6$  and  $R_7$ , have been applied.
- The refactoring *Rename address* ( $R_3$ ) can be applied to the design before or after *Pull Up address* ( $R_5$ ). If  $R_3$  is performed before  $R_5$ , then other two *Pull Up Field* refactorings namely  $R_6$  and  $R_7$  can be applied before or after  $R_3$ . Otherwise,  $R_6$  and  $R_7$  must be applied to the design after  $R_3$ . The second case happens because a precondition in *Rename Field* prevents the field name from being changed if there is already a field with the same name in the class and the fields are used by different methods.
- The method *getPrice* should be pushed down to its subclass using refactoring  $R_{11}$  before

moving the method *retrieveDetails* and the field *conductor*. This is necessary as *getPrice* uses both this method and field. Were *retrieveDetails* or *conductor* to be pushed down to the subclass (using  $R_{12}$  or  $R_{13}$ ) before *getPrice*, then they would not be accessible in *getPrice*. Similarly, the method *retrieveDetails* should be moved before the field *conductor*.

- The accessibility of the field *conductor* can be reduced using  $R_{15}$  after the field is pushed down to the subclass by  $R_{13}$ . Were the accessibility of the field to be reduced before the push down refactoring, then the field would not be accessible in the subclass. This would prevent the pushing down of the methods *getPrice* and *retrieveDetails* as well as the field itself to the subclass. A similar situation arises for the *Decrease Accessibility Method* refactoring ( $R_{14}$ ). The accessibility of the method *retrieveDetails* should be reduced only after the method is pushed down to the subclass ( $R_{12}$ ) in order for it to be accessible in *getPrice*.
- The above dependencies also reveal an implicit dependency between the two refactorings *Decrease Accessibility Method* ( $R_{14}$ ) and *Decrease Accessibility Field* ( $R_{15}$ ) with the *Extract Subclass* refactoring ( $R_2$ ). Both *Decrease Accessibility* refactorings can be performed only after the class *Concert* has been created and the corresponding methods and fields have been moved to the newly created class.

As the example above demonstrates, there can be many relationships between refactorings, and even in this simple motivating example it is difficult to identify them all manually. As shown, the application of one refactoring may prevent certain other refactorings or make possible certain other refactorings. What makes the refactoring process more difficult is that the effect of each refactoring is only seen after the refactoring is applied to the design. The preconditions of a refactoring fail at its turn even though they were satisfied at the start of the sequence, and vice versa.

We describe our automated scheduling approach (REDaCT and REDaCT+) that addresses these problems in Sections 4 and 5, but we first present some preliminary information

Figure 2. Typical workflow when using *Design-Imp*

about the software tools we use in this work, and precise definitions of conflict and dependency.

### 3. Preliminaries

This section provides some necessary preliminary information about the software tool employed in our experiments (Section 3.1) and formal definitions for conflict and dependency in a refactoring sequence (Section 3.2).

#### 3.1. Design-Imp

The investigations described in this paper make use of a software tool named *Design-Imp*. *Design-Imp* is an interactive refactoring framework developed by the authors to facilitate experimentation in improving the design of existing programs. It refactors the software system at a higher level of abstraction than its source code. Figure 2 depicts a typical workflow when using *Design-Imp*.

*Design-Imp* takes Java version 7 source code as input, extracts design information from the source code using the extract model process and expresses the extracted information as an attributed type graph [12]. This graph is then refactored using an interactive evolutionary search technique to improve the program according to a fitness function, expressed in terms of standard software quality metrics such as a combination of cohesion and coupling metrics.

The output comprises the refactored graph, expressed as a UML class diagram, as well as detailed refactoring and metrics information. As most of the program detail, especially method bodies, has been abstracted away, faster precondition checking and refactoring execution is possible. The result of the refactoring process is a desired design based on the employed fitness function and confirmed by the developer.

*Design-Imp* uses AGG API<sup>2</sup> as a graph transformation engine [13, 14] to implement graph transformation rules. Each rule (i.e. refactoring) includes a pattern that is specified by two graphs, left and right hand side, and a morphism between them. Transformation rules may specify negative and positive application conditions as transformation preconditions. A negative application condition (NAC) specifies certain structures that are forbidden, while a positive application condition (PAC) expresses certain structures that are necessary to perform a transformation. Currently, *Design-Imp* supports 20 refactorings shown in Table 1. In the rest of this paper we use the term *refactoring* instead of *transformation rule* when referring to a transformation on the graph.

*Design-Imp* defines a meta-model, expressed as a type graph, based on the syntax of the Java language in order to specify how a Java program should be represented as a graph. It is not possible to define all necessary Java constraints as a type graph, e.g. cyclical inheritance is hard to prevent, so we added some general constraints similar to those defined by Mens [15] to our model.

<sup>2</sup> <http://user.cs.tu-berlin.de/~gragra/agg/>

Table 1. A list of refactorings provided by *Design-Imp*

No.	Class-Level Refactorings	Description
1	Rename Class	Changes the name of a class to a new name, and updates its references.
2	Extract Hierarchy	Adds a new subclass to a non-leaf class $C$ in an inheritance hierarchy.
3	Extract Subclass	Adds a new subclass to class $C$ and moves the relevant features to it.
4	Extract Superclass	Adds a new super class to class $C$ and moves the relevant features to it.
5	Collapse Hierarchy	Removes a non-leaf class from an inheritance hierarchy.
6	Inline Class	Moves all features of a class into another class and deletes it.
7	Extract Class	Creates a new class and moves the relevant features from the old class into the new one.
<b>Method-Level Refactorings</b>		
8	Push Down Method	Moves a method from a class to those subclasses that require it.
9	Pull Up Method	Moves a method from some class(es) to the immediate superclass.
10	Rename Method	Changes the name of a method to a new one, and updates its references.
11	Decrease Method Accessibility	Decreases the accessibility of a method, i.e from protected to private.
12	Increase Method Accessibility	Increases the accessibility of a method, i.e from protected to public.
13	Move Method	Creates a new method with a similar body in the class it uses most. Either turns the old method into a simple delegation, or removes it.
<b>Field-Level Refactorings</b>		
14	Push Down Field	Moves a field from a class to those subclasses that require it.
15	Pull Up Field	Moves a field from some class(es) to the immediate superclass.
16	Move Field	Moves a field from a class to another one which uses the field most.
17	Rename Field	Changes the name of a field to a new name, and updates its references.
18	Decrease Field Accessibility	Decreases the accessibility of a field, i.e from protected to private.
19	Increase Field Accessibility	Increases the accessibility of a field, i.e from protected to public.
20	Encapsulate Field	Creates getter and setter methods for the field and uses only those to access the field.

*Design-Imp* is also capable of detecting conflicts and dependencies between refactorings through the use of a static analysis technique provided by AGG API called *critical pair analysis*. Critical pair analysis computes all the potential conflicts and dependencies between refactorings based on the notion of independence of graph transformations [12]. Using this technique, *Design-Imp* can distinguish three kinds of conflict and three kinds of dependency between refactorings as described in Table 2. Definitions for conflict and dependency are presented next in Section 3.2.

### 3.2. Definitions of Conflict and Dependency between Refactorings

In this section we provide precise definitions for the concepts of *conflict* and *dependency*. These

are concerned with the relationships between refactorings and are widely used in this paper.

#### Definition 1: Dependency

For two given refactorings ( $R_1$  and  $R_2$ ),  $R_2$  is **dependent** on  $R_1$  ( $R_2 \rightarrow R_1$ ) if  $R_2$  can be applied after  $R_1$ , but not before that.

In this paper, as shown in Table 2, we distinguish three types of dependency between refactorings: *produce-use*, *delete-forbid*, and *change-use*. A *produce-use* dependency can happen if  $R_1$  produces an element that is used by  $R_2$ . For example, in the motivating example, refactorings  $R_3$  to  $R_{15}$  are dependent on one of  $R_1$  or  $R_2$ . It is a kind of *produce-use* dependency as a method or field can be moved to a class only if the class has already been created.

As another example, consider a method that uses directly a *private* field in its own class. To push this method down to a subclass it is necessary first to increase the accessibility of the

Table 2. Relationships that can be detected between refactorings using AGG [16]

No. Conflict	Description
1 Delete-use	A refactoring <i>deletes</i> a graph object that is <i>used</i> by another refactoring.
2 Produce-forbid	A refactoring <i>produces</i> a graph structure that is <i>forbidden</i> by another refactoring.
3 Change-use	A refactoring <i>changes</i> an attribute value of a graph object in such a way that it can no longer be <i>used</i> by another refactoring.
No. Dependency	Description
1 Produce-use	A refactoring <i>produces</i> a graph object that is <i>used</i> by another refactoring.
2 Delete-forbid	A refactoring <i>deletes</i> a graph objects that is <i>forbidden</i> by another refactoring.
3 Change-use	A refactoring <i>changes</i> an attribute value of a graph object in such a way that it can be <i>used</i> by another refactoring.

field to at least *protected* to make it accessible to the method in the subclass. In this case, the *Push Down Method* refactoring has a *change-use* dependency on the *Increase Field Accessibility* refactoring.

**Definition 2: Asymmetrical Conflict**

For two given refactorings ( $R_1$  and  $R_2$ ),  $R_1$  has an **asymmetrical conflict** with  $R_2$  ( $\mathbf{R}_1 \nrightarrow \mathbf{R}_2$ ) if  $R_2$  cannot be applied after  $R_1$ .

In this paper, as shown in Table 2, we distinguish three kinds of conflicts: *delete-use*, *produce-forbid*, and *change-use*. As an example, a *delete-use* conflict between  $R_1$ , and  $R_2$  can happen if  $R_1$  deletes one or more elements (classes, methods, or fields) that are used by  $R_2$ .

*Asymmetrical conflict* is a one-way conflict. Thus, a conflict between  $R_1$  and  $R_2$  ( $\mathbf{R}_1 \nrightarrow \mathbf{R}_2$ ) does not imply that the application of  $R_2$  will disable  $R_1$ . In addition, while an *asymmetrical conflict* is indeed a kind of *dependency*, we distinguish between them in this paper. In a conflict situation ( $\mathbf{R}_1 \nrightarrow \mathbf{R}_2$ ), both refactorings can be run individually, but  $R_2$  cannot be run after  $R_1$ . However, in a dependency situation ( $\mathbf{R}_1 \rightarrow \mathbf{R}_2$ ),  $R_2$  can only be run if  $R_1$  is run first.

**Definition 3: Symmetrical Conflict**

For two given refactorings ( $R_1$  and  $R_2$ ),  $R_1$  has a **symmetrical conflict** with  $R_2$  ( $\mathbf{R}_1 \leftrightarrow \mathbf{R}_2$ ) if and only if they cannot both be performed on the design in any order, i.e. ( $\mathbf{R}_1 \nrightarrow \mathbf{R}_2 \wedge \mathbf{R}_2 \nrightarrow \mathbf{R}_1 \Rightarrow \mathbf{R}_1 \leftrightarrow \mathbf{R}_2$ ).

As an example of a *symmetrical conflict*, consider a case where a method is moved from the same original class to two different target classes

using two separate *Move Method* refactorings. While both refactorings are applicable, only one of them can be performed on the design. The other refactoring will fail subsequently as the method is no longer in its original class and so cannot be moved from there.

**Definition 4: Uninjurious Refactoring**

A refactoring with no *symmetrical* or *asymmetrical* conflict with any other refactoring is termed an *uninjurious* refactoring, in the terminology of Liu et al. [10]. This type of refactoring is of interest as it can be added to a refactoring sequence at any stage with no deleterious effect in terms of disabling other refactorings.

#### 4. The REDaCT Algorithm: Handling Conflict and Dependency in Software Refactoring Scheduling

In this section we describe one of the key contributions of this paper: the creation of a refactoring scheduling algorithm that can handle the conflicts and dependencies described in Section 3.2.

To find a valid refactoring sequence, we extend the conflict-aware scheduling approach proposed by Liu et al. [10]. They propose a heuristic algorithm to improve refactoring activities by arranging an application sequence for the available conflicting refactorings. Their approach computes *symmetrical* and *asymmetrical conflicts* between refactorings, where, in a conflict situation, the refactoring that has more effect

on software quality, as defined by the QMOOD metric suite [17], has a higher priority than the other one. The solution we present here improves on the approach of Liu et al. in the following regards:

- The approach of Liu et al. only supports *delete-use* and *change-use* conflicts, and does not support *produce-forbid* conflicts, although they do propose this idea as future work. A *produce-forbid* conflict occurs when a refactoring produces an element or structure that is prohibited by the precondition of another refactoring [18]. Our approach handles all the conflict types in Table 2.
- The approach of Liu et al. does not support any kind of dependency between refactorings. In contrast, our approach is capable of detecting all inter-refactoring dependency types as shown in Table 2. By considering dependencies between refactorings, our scheduling algorithm is able to take into account the effect of a refactoring in terms of the other refactorings that it *enables*, whereas Liu et al. only consider cases where a refactoring *disables* other refactorings.

In Section 4.1 we describe how our refactoring scheduling algorithm, REDaCT, handles conflict and dependency relationships between refactorings. In Section 4.2 we discuss the strengths and weaknesses of this approach to refactoring scheduling.

#### 4.1. The REDaCT Scheduling Algorithm

Our proposed scheduling algorithm, REDaCT is presented as pseudocode in Figure 3. As illustrated, the algorithm takes as input the set of refactorings to be scheduled as well as a square matrix, called *RMatrix*, that contains information about how the input refactorings are related to each other. It is assumed that the refactorings are all beneficial, so a perfect solution is where all the refactorings can be applied. REDaCT is a heuristic that attempts to find the longest possible valid sequence of refactorings that can be applied to the initial design.

*RMatrix* is computed by *Design-Imp* and contains information about conflicts and de-

pendencies between refactorings. A character ‘C’ in  $(row_i, column_j)$  of the matrix means an *asymmetrical conflict* exists between refactorings  $R_i$  and  $R_j$ , so applying  $R_i$  will prevent  $R_j$  from running. A *symmetrical conflict* will exist if  $(row_i, column_j)$  also contains a character ‘C’. On the other hand, a character ‘D’ in  $(row_i, column_j)$  means that  $R_j$  is *dependent* on  $R_i$ , so  $R_j$  is only applicable if  $R_i$  has already been applied to the design. Note that a *symmetrical dependency* is an impossibility.

The REDaCT scheduling algorithm is depicted in Figure 3. Five critical steps in the algorithm are highlighted and are elucidated in the paragraphs below:

**Step 1:** In the first step, it is necessary to find refactorings that are not dependent on any refactorings as well as having no conflict with other refactorings. A refactoring with no *symmetrical* or *asymmetrical* conflict with other refactorings is selected in order to prevent it from being disabled by other refactorings that might have an *asymmetrical conflict* with it [10]. However, such an uninjurious refactoring (see Section 3.2) is only selected if it is also not *dependent* on any refactorings except those already added to the refactoring sequence. This step guarantees that, where possible, all refactorings upon which a candidate refactoring is dependent are added to the refactoring sequence early in the process.

After a refactoring is added to the refactoring sequence, its corresponding row and column is removed from *RMatrix* as well. The algorithm may terminate at the end of first step if all refactorings have been added to the refactoring sequence. This only happens if there is no *symmetrical conflict* between any pair of refactorings in the set.

**Step 2:** In the second step, assuming that *RMatrix* is not empty, the score for each applicable refactoring  $R_c$  is computed using the following formula:

$$\begin{aligned} \text{score}(R_c) = & \text{directEffect}(R_c) + \\ & \text{positiveEffect}(R_c) - \\ & \text{negativeEffect}(R_c) \end{aligned} \quad (1)$$

It is assumed that each refactoring in the refactoring set has a positive effect, i.e. that the developer has selected only refactorings that have a positive effect on the design of the program. Therefore, the maximum quality improvement is obtained if all refactorings in the refactoring set are applied to the initial design. Hence, we set the *directEffect* of each refactoring to 1, meaning that the application of each refactoring leads the refactoring process one step closer to the maximum achievable quality improvement. (Later, in Section 5, we will use a more sophisticated approach for computing the direct effect of a refactoring.)

The application of a refactoring  $R_c$  enables refactorings that are dependent on  $R_c$  to be run, assuming that they are not dependent on other available refactorings (see Section 3.2). In this paper, we count all these effects as the *positiveEffect* of the candidate refactoring  $R_c$ . So the positive effect of a candidate refactoring is the total number of refactorings that are enabled by it.

When a candidate refactoring  $R_c$  is applied to the design, it also disables other refactorings,  $R_o$ , with which it has an *asymmetrical conflict* [10]. In addition, if  $R_o$  is disabled, its dependent refactorings are disabled as well. In this paper, we count all these effects as the *negativeEffect* of the candidate refactoring,  $R_c$ . So the negative effect of a candidate refactoring is the total number of refactorings that are disabled by it.

**Step 3:** In the third step, the refactoring with the highest score is selected and added to the refactoring sequence. Since the score is based on the number of refactorings that will be disabled or enabled by the refactoring, the selection of a high-scoring refactoring promotes refactorings that increase the number of refactorings that can be selected in subsequent iterations.

**Step 4:** After the best refactoring is added to the refactoring sequence, it is necessary to update the scoring of refactorings that have been positively affected by the application of this refactoring. This includes refactorings that have an *asymmetrical conflict* with the selected refactoring [10], as well as refactorings that are *dependent* on the selected refactoring. The score

for these positively affected refactorings is updated using Eq. 2 below. As shown, the merit of the selected refactoring is added to its affected ones in order to increase their chance of being selected in subsequent iterations.

$$\text{score}(R_{\text{affected}}) = \text{score}(R_{\text{affected}}) + \text{score}(R_{\text{selected}}) \quad (2)$$

As shown in Figure 3, after the best refactoring is added to the refactoring sequence, all newly disabled refactorings are also removed from *RMatrix* to prevent them from being needlessly selected in subsequent iterations.

**Step 5:** In this step, refactorings that are neither dependent on, nor in conflict with, any remaining refactorings are added to the refactoring sequence. They can be safely applied at this stage, and doing so immediately prevents such a refactoring from being subsequently disabled by a refactoring with a better score that has an *asymmetrical conflict* with it.

At the end of the algorithm, *refactoringSeq* will contain the longest sequence of refactorings found that can be applied to the initial design.

## 4.2. Summary

To summarise this section, we have presented the REDaCT algorithm, which is our novel approach to refactoring scheduling that extends the state of the art [10] by handling a more extensive range of conflicts and dependencies. This algorithm is evaluated later in Section 6. REDaCT ignores the effect the refactorings have in terms of how close they bring the program to its desired design. In the next section we address this issue.

## 5. The REDaCT+ Algorithm: Improving Refactoring Scheduling by Estimating the Contribution of Refactorings to Achieving the Desired Design

The approach proposed by Liu et al. uses the QMOOD metric suite [17] to measure the effect

---

**Input:** *refactoringSet*: set of refactorings to be scheduled.

**Input:** *RMatrix*: square matrix contains relationships between refactorings.

**Output:** *refactoringSeq*: contains a valid order of refactorings.

---

```

procedure SCHEDULING ALGORITHM(refactoringSet, RMatrix)
  refactoringSeq = null
  while (hasUninjuriousRefactoring()) do                                ▷ Step 1
    refactoringSeq.add(pickUninjuriousRefactoring())
    updateRMatrix()
  end while
  if (!RMatrix.isEmpty()) then                                        ▷ Step 2
    measureScore()                                                    ▷ Step 2
    repeat                                                            ▷ Step 3
      refactoringSeq.add(pickBestRefactoring())                        ▷ Step 3
      updateScores()                                                  ▷ Step 4
      updateRMatrix()
      while (hasUninjuriousRefactoring()) do                        ▷ Step 5
        refactoringSeq.add(pickUninjuriousRefactoring())
        updateRMatrix()
      end while
    until (RMatrix.isEmpty())
  end if
  return refactoringSeq
end procedure

```

---

The functions used are defined as follows:

**hasUninjuriousRefactoring()**: Returns *true* if *RMatrix* contains at least one independent and uninjurious refactoring. Otherwise, returns *false*.

**pickUninjuriousRefactoring()**: Returns the first independent and uninjurious refactoring.

**pickBestRefactoring()**: Returns the refactoring with the highest score.

**updateRMatrix()**: The selected refactoring is removed from *RMatrix*(). The refactorings with which the selected refactoring has a conflict are removed from *RMatrix*() as well.

**measureScore()**: Computes the score for all remaining refactorings using equation 1.

**updateScores()**: Updates the score for the affected refactorings using equation 2.

---

Figure 3. The REDaCT algorithm. It orders the input refactorings to create the longest possible applicable refactoring sequence, in the presence of conflict and dependency between the refactorings

of refactorings on software quality. However, applying refactorings to the design and measuring their effect requires considerable effort. In addition, as no dependency between refactorings is detected by Liu et al., the impact of each refactoring is measured individually, and that cannot capture the real effect of a sequence of refactorings. In Section 5.1 below we describe a known, string-based approach to comparing software designs and put this to novel use in Section 5.2 to introduce a novel, lightweight approach to measuring the effect of a refactoring without actually applying the refactoring to the design. Finally, in Section 5.3, this approach to measuring refactoring effect is included in the scheduling algorithm to improve the accuracy of

the scheduling approach; we term this extension *REDaCT+*.

### 5.1. Measuring Similarity between Software Designs

In this paper, an improvement in quality means an improvement in the similarity between the initial and desired designs. Therefore, during the refactoring process, a refactoring that improves the similarity between the initial and desired designs has priority over other refactorings.

To measure the degree of similarity between two designs, REDaCT+ uses a sequence alignment algorithm called *Fast Optimal Global Sequence Alignment Algorithm (FOGSAA)* de-



veloped by Chakraborty and Bandyopadhyay [19]. This algorithm is capable of finding the best alignment between two input strings with a lower computational complexity than other global alignment approaches. Full details of this algorithm are presented in the paper cited above.

To use the FOGSAA alignment algorithm in REDaCT+, the first step is to represent program's features such as classes, methods, fields etc. as a sequence of characters. In this paper, we use the approach proposed by Kessentini et al. [20] as a method to represent program elements as a string. Each element in the input Java program is represented using a specific character as follows: *Class (C)*, *generalization relationship (G)*, *realization relationship*<sup>3</sup> (*I*), *attribute (A)*, *method (M)*, *method parameter (P)*, and a coupling between two classes (*R*). As an example, the representation of class *B* shown in Figure 4a is *CGMMPR*. This sequence shows that the class inherits from another class, has a coupling relationship with one other class in the program and contains two methods, where the second method has a parameter.

However, our representation differs from that of Kessentini et al. [20] in two significant ways. Firstly, in their work each character includes more detailed information such as name, type, accessibility etc. depending on the program element it represents. However, in our approach the element name is the only information that is included with each character. Secondly, in Kessentini et al. [20] every method invocation or field reference is represented by one *R* character. Therefore, if a class invokes a method in another class *n* times, *n* *R* characters are added to the resulting string. However, the number of accesses to fields and methods in a class is usually far greater than the number of fields and methods in the class, so this approach overemphasises the importance of *R* relationships over the other types when measuring similarity. To improve the efficiency of the alignment algorithm, we use a single *R* character to denote a coupling from the original class to another class without counting

the number of connections between the two classes.

## 5.2. Expressing Refactoring Effect on the String Representation of a Program

Expressing the program as a sequence of characters and using an alignment algorithm to measure similarity between strings helps in measuring the effect of refactorings without actually applying them to the design. However, in order to do this it is necessary to determine first how the resulting string should be changed when a refactoring is applied to it.

Figure 4 illustrates an example of how the *Move Method* refactoring changes the software design and the related string representation. In this example, method *b2(c)* is moved from its original class, named *B*, to a related target class named *C*. Figures 4a, and 4b show the UML design and the related string before and after refactoring respectively. Because the classes are related through the method parameter, after refactoring the input parameter is removed from the method signature. As illustrated, the sequence that shows class *A* (the first part in each sequence indicated by *CA*) is not changed as the refactoring has no effect on that. However, both sequences related to class *B*, and *C* (the second and third parts in each sequence) are changed because of the refactoring.

For each of the refactoring types in Table 1, its effect on the string representation of a program design is defined in a similar way as described for the *Move Method* refactoring. This enables us to estimate quickly the approximate effect of a refactoring without having to operate on source code parse trees. Note that the effect of a refactoring is only measurable when all refactorings upon which it depends have been applied to the design, e.g. the effect of a *Move Method* refactoring is only measurable if the target class has already been created. We use a topological sort to create a linear ordering of the refactorings based on their dependency to ensure that all refactorings upon which a refactoring depends precede it in the ordering. Note that the refac-

<sup>3</sup> Realization in Java is the relationship between a class and an interface that it implements.

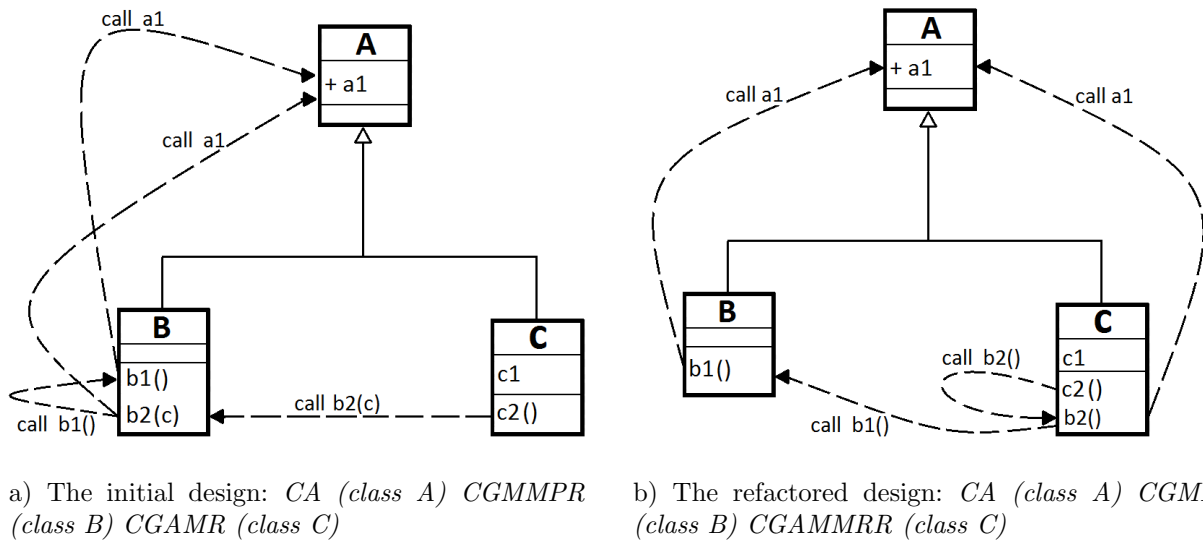


Figure 4. How refactoring effect is expressed on the string representation of a design

toring sequence produced by the topological sort algorithm does not show the optimal ordering between refactorings as it does not take conflicts between refactorings into account.

The quality improvement achieved by a refactoring is thus measured by the FOGSAA string alignment algorithm. It expresses the difference in similarity between the current and desired designs before and after the applied refactoring. Using this approach, we can determine, for any refactoring under consideration, to what extent it contributes to achieving the desired design. In the next section we include this measure in the refactoring scheduling approach.

### 5.3. Including Refactoring Effect in the Scheduling Algorithm

The REDaCT scheduling algorithm described earlier in Section 4 tries to order the refactorings so that the maximum number of refactorings can be applied to the design. However, finding the longest sequence of refactorings is not always the best option to order refactorings, especially if there are *symmetrical conflicts* between refactorings, and different refactorings make different contributions to achieving the desired design.

To improve the scheduling algorithm, we extend it to include the contribution of the refactoring to achieving the desired design. We term

this contribution the *refactoringEffect*, and it is measured as described in Section 5.2. Thus, the scoring function defined by Eq. 1 in Section 4 is changed as below. As shown, the default value for *directEffect* used in Eq. 1 is changed from 1 to the contribution of the refactoring on the similarity between designs:

$$\text{score}(R_c) = \text{refactoringEffect}(R_c) + \text{positiveEffect}(R_c) - \text{negativeEffect}(R_c) \quad (3)$$

All components of this summation are equally weighted. Thus the decision on whether to accept a refactoring depends equally on the contribution the refactoring makes to the desired design, the effect of refactorings it enables and the effect of refactorings it disables. In Section 6 this new approach, REDaCT+, is evaluated and it is compared with the vanilla REDaCT algorithm.

## 6. Evaluation

We have presented an algorithm for refactoring scheduling in the presence of conflict and dependency (REDaCT) and augmented this algorithm to exploit a desired design, if one is available (REDaCT+). In this section we evaluate these

algorithms by applying them to a number of examples and assessing the results.

This section is structured as follows. In Section 6.1 we test the correctness of the REDaCT algorithm by applying it to the Event system described in the motivating example of Section 2. In Section 6.2 we demonstrate the necessity for the REDaCT+ algorithm, and evaluate this algorithm. In Section 6.3 we evaluate the ability of the REDaCT+ algorithm to schedule a ‘noisy’ refactoring sequence to achieve a desired design, while in Section 6.4 we evaluate the REDaCT+ algorithm on a medium-sized open source application. In Section 6.5 we summarise the results of our experiments.

### 6.1. Testing the Correctness of the REDaCT Algorithm

To test that the scheduling algorithm operates correctly, we applied it to the Event system that was used as a motivating example in Section 2. The aim is to determine if our refactoring scheduling algorithm can order the refactorings in such a way as to generate the desired design shown in Figure 1b from its initial one depicted in Figure 1a. The refactorings in question are  $R_1$  to  $R_{15}$  as defined in Section 2. As detailed in that section, a considerable amount of conflict and dependency exists between these refactorings and it is not immediately clear if they can all be applied to the initial design or not, so this forms a robust test for the REDaCT algorithm.

Applying REDaCT to the refactoring set shown in Section 2 yielded an ordering that enabled all 15 refactorings to be applied to the design as follows:  $R_1, R_3, R_5, R_8, R_4, R_2, R_6, R_7, R_{10}, R_9, R_{11}, R_{12}, R_{13}, R_{14}, R_{15}$ . The resulting design was identical to the refactored design, meaning that the refactorings were indeed performed in the correct order. As no *symmetrical conflict* was detected between the input refactorings, all 15 refactorings could be applied to the design.

The example used here is small, but the conflicts and dependencies between the refactorings are more complicated than would usually be encountered in a real-world system. In a larger sys-

tem, typically only a few refactorings are applied to a class and its immediate relatives, so the level of conflict and dependency tends to be lower and sparser than in the example we use here. Nevertheless, when such conflicts and dependencies occur, they have to be addressed.

The result we obtain above demonstrates the ability of the REDaCT algorithm to handle the conflicts and dependencies between refactorings and hence to find an effective application order for the given refactorings.

### 6.2. Contrasting the REDaCT and REDaCT+ Algorithms

The example we use here is a simplistic Automatic Teller Machine (ATM) simulation application [21]. It was developed by an inexperienced Java programmer, and so we expect that its design is not optimum and is easy to improve. Using *Design-Imp*, this ATM application was refactored using a fitness function defined as a combination of the two software metrics SCC (Similarity-based Class Cohesion) [22] and DCC (Direct Class Coupling) [17]. Table 3 depicts the refactoring sequence  $R_1 \dots R_{10}$  that led to the design for the ATM system depicted in Figure 5.

When REDaCT was applied to the original ATM design and the set of refactorings, it produced the refactoring sequence  $R_8, R_5, R_1, R_2, R_4, R_3, R_6, R_9, R_7, R_{10}$ . This refactoring sequence is correct in that it yields the desired design when applied to the original ATM program design. However it is apparent that during the refactoring process the methods *printReceipt()* and *displayBalance()* are needlessly moved around various classes before being placed in their final target class.

To test if REDaCT is capable of finding a better sequence, we added two new *Move Method* refactorings,  $R_{11}$  and  $R_{12}$ , to the refactoring set. These new refactorings directly move the methods *printReceipt()* and *displayBalance()* from their original class to their correct target class. These refactorings are highlighted in grey in Table 3. The addition of the new refactorings created two *symmetrical conflicts* as follows:  $R_1 \leftrightarrow R_{12}$ , and  $R_3 \leftrightarrow R_{11}$ , and,

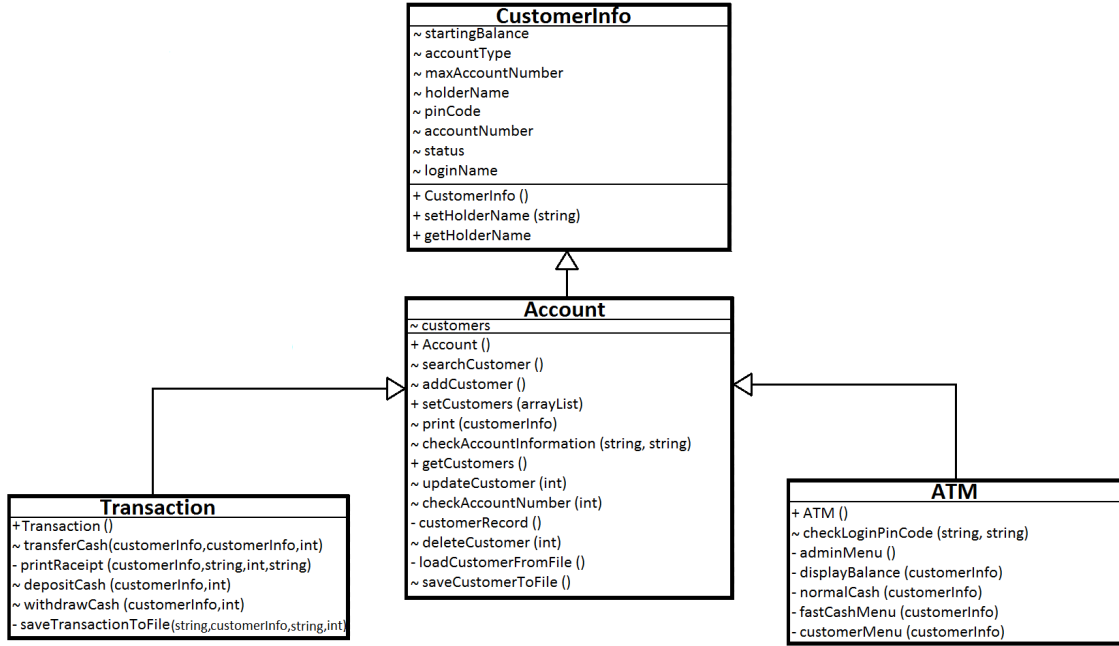


Figure 5. UML class diagram of an ATM application after refactoring

Table 3. Sequence of refactorings applied to the ATM application.  $R_1$  to  $R_{10}$  represent the sequence produced by *Design-Imp* in creating the design of Figure 5.  $R_{11}$  and  $R_{12}$  were both added by hand to test the REDaCT+ algorithm

No.	Refactoring	Feature	Source class	Target class
$R_1$	Move Method	displayBalance()	Transaction	CustomerInfo
$R_2$	Move Method	displayBalance()	CustomerInfo	ATM
$R_3$	PullUp Method	printReceipt()	ATM	Account
$R_4$	Encapsulate Field	customers	Account	
$R_5$	PushDown Method	print()	CustomerInfo	Account
$R_6$	PullUp Method	printReceipt()	Account	CustomerInfo
$R_7$	Decrease Method Accessibility	displayBalance()	ATM	
$R_8$	Encapsulate Field	holderName	CustomerInfo	
$R_9$	Move Method	printReceipt()	CustomerInfo	Transaction
$R_{10}$	Decrease Method Accessibility	printReceipt()	Transaction	
$R_{11}$	Move Method	printReceipt()	ATM	Transaction
$R_{12}$	Move Method	displayBalance()	Transaction	ATM

because of the dependencies among the refactorings, three new *symmetrical conflicts* as follows:  $R_2 \leftrightarrow R_{12}$ ,  $R_6 \leftrightarrow R_{11}$  and  $R_9 \leftrightarrow R_{11}$ .

In this new situation, an optimal scheduling algorithm should select both new *Move Method* refactorings instead of other refactorings ( $R_1$ ,  $R_2$ ,  $R_3$ ,  $R_6$  and  $R_9$ ), as the newly added *Move Method* refactorings move the methods directly to their target class, which is the clearest and most comprehensible solution. However, the REDaCT algorithm still selects the same

sequence as it did before, without including  $R_{11}$  and  $R_{12}$  in the sequence. This happens because the REDaCT algorithm favours refactorings that in turn increase the number of refactorings that can be selected in subsequent iterations of the algorithm.

We tested if the REDaCT+ algorithm could find the optimum refactoring sequence for the example described above, where the vanilla REDaCT algorithm failed, and found that the improved REDaCT+ algorithm did indeed find

the equally good, but shorter, order among the 12 input refactorings shown in Table 3. REDaCT+ moved the two methods *printReceipt*, and *displayBalance* directly to their target class using the two added *Move Method* refactorings  $R_{11}$  and  $R_{12}$ , and rejected the now-superfluous refactorings  $R_1$ ,  $R_2$ ,  $R_3$ ,  $R_6$  and  $R_9$ . REDaCT+ succeeded as it does not simply apply the maximum number of refactorings as REDaCT does, but it uses its knowledge of the design desired to select refactorings that moved the program design towards this desired design, as detailed in Section 5.

### 6.3. Evaluating REDaCT+ on a ‘Noisy’ Refactoring Sequence

To test how REDaCT+ would deal with a larger application, we applied it to the *Design-Imp* software itself. *Design-Imp* contains 65 classes that include 227 attributes and 600 methods, and so is larger than the earlier examples. In this experiment we also wanted to test how well REDaCT+ could deal with a ‘noisy’ refactoring set, one that contains redundant refactorings and so is a superset of the set of refactorings required to bring the initial program to its desired design.

The ‘noisy’ refactoring set was created as follows. First we use *Design-Imp* to automatically refactor the *Design-Imp* software twice in order to create two separate desired designs. As described in Section 3.1, the search-based algorithm used by *Design-Imp* is a stochastic one so each refactoring process yields a different set of refactorings but with the possibility of some commonality between the two refactoring sets. Figure 6 shows a breakdown of the combined set of refactorings produced by both refactoring processes. The ‘noisy’ refactoring set then is the union of these two sets. Duplicates were not removed, so this set (technically a multiset) contained 60 refactorings in total. In fact only one refactoring appeared in both refactoring sets, so the combined set contained 59 unique refactorings. A total of 3 conflicts and 298 dependencies

were found to exist in this combined refactoring set.

We then selected one of these resulting designs as the final desired design and used REDaCT+ to try to refactor the initial design towards the selected final desired design. This is a robust challenge, where both the conflict and dependency aspects of the base REDaCT algorithm have to combine with the ‘desired design’ aspects of the REDaCT+ algorithm in order to filter out the unnecessary refactorings and attempt to produce a refactoring sequence that yields the given desired design.

REDaCT+ was able to find a refactoring sequence that was 93% correct and brought the program design to one close to the given desired design. In two cases the algorithm categorised two correct *Move Field* refactorings as detrimental even though both refactorings were critical to achieving the desired design. This problem occurred as *Design-Imp* incorrectly detected a spurious dependency between these two refactorings and another truly detrimental refactoring.

To sum up, this example shows the ability of the REDaCT+ algorithm to filter out refactorings that do not help in achieving the desired design. It also reveals how invalid inter-refactoring dependencies detected by *Design-Imp* can affect the accuracy of the REDaCT+ algorithm, leading to two correct refactorings being ignored. From this we see that the ability of REDaCT+ to correctly order a refactoring set is dependent on the accuracy of the detected conflicts and dependencies between the refactorings.

### 6.4. Evaluating REDaCT+ on an Open Source Example

To investigate how REDaCT+ works with a non-trivial open source application, we used *JGraphX*<sup>4</sup> as an application to investigate. *JGraphX* is a medium-sized Java library that is used to display interactive diagrams and graphs and comprises 188 classes, 1356 attributes and 2908 methods.

In this experiment, we used *Design-Imp* to automatically refactor the design of the origi-

<sup>4</sup> <http://www.jgraph.com/jgraphdownload.html>

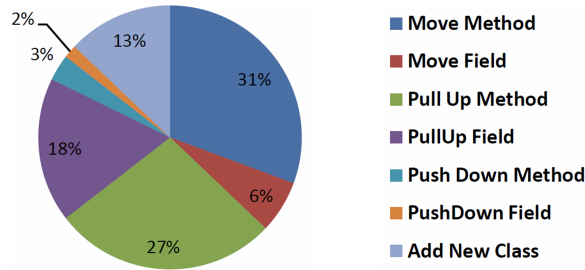


Figure 6. Breakdown of the 60 refactorings included in the refactoring set

nal *JGraphX* application to create a desired design to use in our experiments. We then used *Design Imp* to generate a refactoring set from the differences between the two designs, which yielded a refactoring set containing 50 refactorings.

To test the REDaCT+ algorithm, we applied it to the initial and desired designs of the *JGraphX* application as well as the generated refactoring set. The challenge here is that this is an open source application that the experimenters have no *a priori* knowledge of, and the refactoring set is non-trivial in size. However, REDaCT+ produced a refactoring sequence that transformed the initial design to the desired design with 100% success and with no spurious refactorings appearing in the sequence.

Note that in a large software system there are many possibilities for refactoring, and the size of the system tends to lead to less dependency and conflict between applied refactorings. However, as mentioned, the aim of this experiment was to show the proposed approach can be used for larger applications as well.

## 6.5. Evaluation Conclusion

The goal of this evaluation section was to provide an overall assessment of how the REDaCT and REDaCT+ algorithms perform when faced with a variety of challenges.

We demonstrated the basic correctness of the REDaCT algorithm in Section 6.1 by showing that it is able to correctly sequence a heavily conflicted and interdependent set of refactorings. While the refactoring set was small in size, the intensity of the conflict and dependency was far

greater than that found in our tests with an open source example in Section 6.4.

In Section 6.2 we demonstrated that, when several options exist, REDaCT fails to find the shortest refactoring sequence but that REDaCT+ is able to succeed by using its extra knowledge of the desired design that the refactoring sequence is trying to achieve. This established the case for the REDaCT+ algorithm and this is the algorithm that we evaluate further.

In Section 6.3 we demonstrated that the REDaCT+ algorithm can schedule a ‘noisy’ refactoring sequence that has many optically-relevant but useless refactorings. A perfect solution was not achieved, but the 93% success rate is very satisfactory. In effect this means that a design very close to the desired design is achieved, and it is left to the developer to perform the final refactoring steps by hand.

In our final evaluation in Section 6.4 we demonstrated that the REDaCT+ algorithm can find a correct refactoring sequence when trying to schedule a non-trivial refactoring set (50 refactorings) on a medium-sized open source application. This demonstrates that REDaCT+ can perform well in a more realistic context.

We found that the size of program under investigation has only a minor effect on the speed of the REDaCT+ scheduling algorithm. The only time-consuming part is the algorithm used to identify conflicts and dependencies between refactorings. The number of comparisons to find conflict and dependency between refactorings is equal to  $n * (n - 1) / 2$ , where  $n$  is the number of refactorings included in the refactoring set.

## 7. Related Work

The work related to this paper can be divided into three research areas: ranking refactoring opportunities, search-based refactoring and scheduling refactoring. These topics are discussed below.

Ranking refactoring opportunities involves sorting refactoring opportunities according to one or more criteria such as their impact on the overall design quality. Tsantalis and Chatzigeorgiou [23] propose a semi-automatic approach to identifying refactoring opportunities related to code smells based on system history and find that a refactoring opportunity involving a highly changeable code fragment is most likely to be improved through refactorings in the future, and therefore such refactorings should have a higher priority than others. In other work they propose an approach for detecting Move Method refactoring opportunities based on code smells [24] and also propose an approach to finding refactoring opportunities that introduce polymorphism as an alternative to state checking [25]. In these approaches, detected refactoring opportunities are ranked based on their impact on the overall design quality. However, the effect of refactorings is measured individually, without considering impact of refactorings on one another, and so it cannot guarantee to find the best sequence in which to apply the proposed refactorings.

In contrast to the aforementioned semi-automatic approaches, other research works aim to automate the whole process by using *search based refactoring* to find and apply a sequence of refactorings to a program [26]. In this approach, the process of refactoring is guided by a fitness function and a refactoring is accepted only if it improves the merit of the design based on metrics included in the fitness function. This approach has been used for several purposes: software quality improvement [27, 28], to fix code smells [20, 29] and to apply design patterns [30]. Although search-based refactoring techniques help to automatically find a close-to-optimal sequence of refactorings based on the employed

fitness function, they do not usually generate the most effective refactoring order.

In the remainder of this section we examine closely related research that also aims to detect *dependencies* and *conflicts* between refactorings in order to sort refactorings into an optimum order for application. Mens et al. [11] use parallel and sequential dependency analysis to detect relationships among a set of input refactorings. Like our approach, the program and refactoring activities are considered as a graph and graph-based rules respectively. However, they only focus on specifying relationship between refactorings without introducing a practical algorithm on how the refactorings should be arranged. Further, in their work, sequential dependencies are only detected after a candidate refactoring is applied to the design and then find out which refactorings become applicable or inapplicable. Thus, the approach cannot detect existing sequential dependencies between refactorings at once without applying refactorings to the source code.

Zibran and Roy [31] introduce a scheduling approach based on three factors: maximised quality gain measured based on standard software quality metrics, minimised refactoring effort estimated by a proposed refactoring effort model and satisfaction of higher priorities defined manually by the developer. They formulate the scheduling of code clone refactorings as a *constraint satisfaction optimisation problem*, and use *constraint programming* to implement the proposed model.

Estimation of clone refactoring effort using an effort model has also been investigated by Bouktif et al. [32]. They use a simple genetic algorithm to schedule clone refactoring activities in order to achieve the greatest quality improvement with minimum resource consumption while satisfying priority constraints. However, they ignore conflicts between clone refactorings and assume that duplicated codes can be refactored independently, which is not a correct assumption.

Among the research works focused on scheduling code smell refactoring, the work of Liu et al. [10] is most closely related to ours.

They propose a heuristic algorithm to schedule code smell refactorings. We extend their refactoring scheduling algorithm by considering not just *conflicts* between refactorings but also *dependencies*. Furthermore, we take into account a type of refactoring conflict not handled in their work, where the application of one refactoring violates the precondition of another. We also introduce the idea of a desired design and show how it can be included in the scheduling algorithm to guide the refactoring process more effectively. We measure not only the effect of a refactoring in terms of its direct contribution to achieving the desired design, but also its indirect contribution in terms of the refactorings it enables and disables. Later work by Liu et al. [2] looks at the related problem of scheduling code smell detection and resolution when a software system is radically refactored in ‘batch’ mode. They show that the order in which code smells are addressed is significant due to the overlap between smells, and show that refactoring effort can be reduced significantly (by up to 20%) by appropriate scheduling.

Lee et al. [33] also take into account that a refactoring can also *enable* other refactorings. They use a genetic algorithm to identify an appropriate refactoring schedule for code clones. To support both cases, the original refactoring set is updated according to changes applied in the program after the refactoring is performed. The effect of each refactoring sequence expressed as a chromosome is measured using the QMOOD quality model [17]. However, the fitness evaluation is expensive due to the fact that each chromosome must be individually applied to the system and then its effect on quality measured. It is different from our approach, where refactorings are applied to a sequence of characters (representing the source code) and each refactoring is simulated only once and its effect on the quality is measured at that time.

## 8. Conclusions and Future Work

In this paper we presented the REDaCT algorithm, our approach to refactoring schedul-

ing in the presence of inter-refactoring conflicts and dependencies that extends the state of the art [10] by handling a more extensive range of conflicts and dependencies. We also developed an extension to this algorithm, REDaCT+, that also takes into account the contribution of each refactoring towards achieving a given *desired design* for the software. To validate our proposed scheduling approach, we carried out evaluations on four examples: two small constructed examples, a software tool developed by the authors and a medium-sized open source software system. The results obtained demonstrated that REDaCT can order a refactoring sequence in the presence of conflicts and dependencies, that REDaCT+ can outperform REDaCT and that REDaCT+ works well even when many irrelevant refactorings are included in the refactoring set.

In future work we plan to explore further the process of creating the desired design, using an interactive process similar to that proposed by Simons et al. [3,4]. This paves the way for the REDaCT+ refactoring algorithm to form part of a larger, interactive framework that helps the developer to create a desired design, and then refactors the code to comply with this design, a notion described in our earlier work [5]. In this context, more extensive evaluation with larger software systems and with software developers will be necessary.

## Acknowledgments

This work was funded by the Irish Programme for Research in Third-Level Institutions (PRTL) and in part by Science Foundation Ireland grant 10/CE/I1855 to Lero – the Irish Software Research Centre ([www.lero.ie](http://www.lero.ie)).

## References

- [1] E. Murphy-Hill and A.P. Black, “Refactoring tools: Fitness for purpose,” *IEEE Software*, Vol. 25, No. 5, 2008, pp. 38–44.
- [2] H. Liu, Z. Ma, W. Shao, and Z. Niu, “Schedule of bad smell detection and resolution: A new



- way to save effort,” *IEEE Transactions on Software Engineering*, Vol. 38, No. 1, 2012, pp. 220–235.
- [3] C.L. Simons, I.C. Parmee, and R. Gwynlliw, “Interactive, evolutionary search in upstream object-oriented class design,” *IEEE Transactions on Software Engineering*, Vol. 36, 2010, pp. 798–816.
- [4] C.L. Simons and I.C. Parmee, “Elegant object-oriented software design via interactive, evolutionary computation,” *IEEE Transactions on Systems, Man, and Cybernetics: Part C: Applications and Reviews*, Vol. 42, No. 6, November 2012, pp. 1797–1805.
- [5] I. Hemati Moghadam and M. Ó Cinnéide, “Automated refactoring using design differencing,” in *Proceedings of the 16th European Conference on Software Maintenance and Reengineering*, ser. CSMR’12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 43–52.
- [6] I. Hemati Moghadam, “Multi-level automated refactoring using design exploration,” in *Proceeding of the 3rd International Symposium on Search Based Software Engineering*, ser. SS-BSE’11. Springer, September 2011, pp. 70–75.
- [7] Z. Xing and E. Stroulia, “Differencing logical UML models,” *Journal of Automated Software Engineering.*, Vol. 14, June 2007, pp. 215–259.
- [8] Z. Xing and E. Stroulia, “Refactoring detection based on UMLDiff change-facts queries,” in *Proceedings of the 13th Working Conference on Reverse Engineering*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 263–274.
- [9] I. Hemati Moghadam and M. Ó Cinnéide, “Code-Imp: a tool for automated search-based refactoring,” in *Proceeding of the 4th workshop on Refactoring tools*, ser. WRT ’11. New York, NY, USA: ACM, 2011, pp. 41–44.
- [10] H. Liu, G. Li, Z.Y. Ma, and W.Z. Shao, “Conflict-aware schedule of software refactorings.” *IET Software*, Vol. 2, No. 5, 2008, pp. 446–460.
- [11] T. Mens, G. Taentzer, and O. Runge, “Analysing refactoring dependencies using graph transformation,” *Software and Systems Modeling*, Vol. 6, No. 3, 2007, pp. 269–285.
- [12] R. Heckel, J.M. Küster, and G. Taentzer, “Confluence of typed attributed graph transformation systems,” in *Graph Transformation*, ser. Lecture Notes in Computer Science. Springer, 2002, Vol. 2505, pp. 161–176.
- [13] G. Taentzer, “AGG: A tool environment for algebraic graph transformation,” in *Applications of Graph Transformations with Industrial Relevance*, ser. Lecture Notes in Computer Science, Vol. 1779. Springer, 2000, pp. 481–488.
- [14] G. Taentzer, “AGG: A graph transformation environment for modeling and validation of software,” in *Applications of Graph Transformations with Industrial Relevance*, ser. Lecture Notes in Computer Science. Springer, 2004, Vol. 3062, pp. 446–453.
- [15] T. Mens, “On the use of graph transformations for model refactoring,” in *Generative and transformational techniques in software engineering*. Springer, 2006, pp. 219–257.
- [16] O. Runge, “The AGG 1.5.0 development environment – the user manual,” 2014. [Online]. <http://user.cs.tu-berlin.de/~gragra/agg/AGG-ShortManual/AGG-ShortManual.html>
- [17] J. Bansiya and C. Davis, “A hierarchical model for object-oriented design quality assessment,” *IEEE Transactions on Software Engineering*, Vol. 28, 2002, pp. 4–17.
- [18] L. Lambers, H. Ehrig, and F. Orejas, “Conflict detection for graph transformation with negative application conditions,” in *Graph Transformations*, ser. Lecture Notes in Computer Science. Springer, 2006, Vol. 4178, pp. 61–76.
- [19] A. Chakraborty and S. Bandyopadhyay, “FOGSAA: Fast optimal global sequence alignment algorithm,” *Scientific reports*, Vol. 3, 2013.
- [20] M. Kessentini, R. Mahaouachi, and K. Ghedira, “What you like in design use to correct bad-smells,” *Software Quality Journal*, Vol. 21, No. 4, 2013, pp. 551–571.
- [21] O. Mursleen, “Java code for atm operations,” June 2010. [Online]. [http://freemsourcecode.net/javaprojects/13191/Java-code-for-ATM-Operations#.VL\\_PC6YqYaA](http://freemsourcecode.net/javaprojects/13191/Java-code-for-ATM-Operations#.VL_PC6YqYaA)
- [22] J.A. Dallah and L.C. Briand, “An object-oriented high-level design-based class cohesion metric,” *Journal of Information & Software Technology*, Vol. 52, No. 12, 2010, pp. 1346–1361.
- [23] N. Tsantalis and A. Chatzigeorgiou, “Ranking refactoring suggestions based on historical volatility,” in *Proceedings of the 15th European Conference on Software Maintenance and Reengineering*, ser. CSMR’11. IEEE Computer Society, March 2011, pp. 25–34.
- [24] N. Tsantalis and A. Chatzigeorgiou, “Identification of move method refactoring opportunities,” *IEEE Transactions on Software Engineering*, Vol. 35, 2009, pp. 347–367.
- [25] N. Tsantalis and A. Chatzigeorgiou, “Identification of refactoring opportunities introducing

- polymorphism,” *Journal of Systems and Software*, Vol. 83, March 2010, pp. 391–404.
- [26] O. Raiha, “A survey on search-based software design,” *Computer Science Review*, Vol. 4, No. 4, 2010, pp. 203–249.
- [27] M. Harman and L. Tratt, “Pareto optimal search based refactoring at the design level,” in *Proceedings of the 9th annual Conference on Genetic and Evolutionary Computation*, ser. GECCO ’07. New York, NY, USA: ACM, 2007, pp. 1106–1113.
- [28] M. O’Keeffe and M. Ó Cinnéide, “Search-based refactoring for software maintenance,” *Journal of Systems and Software*, Vol. 81, No. 4, 2008, pp. 502–516.
- [29] A. Ouni, M. Kessentini, H. Sahraoui, and M. Boukadoum, “Maintainability defects detection and correction: a multi-objective approach,” *Automated Software Engineering*, Vol. 20, No. 1, 2013, pp. 47–79.
- [30] A.C. Jensen and B.H. Cheng, “On the use of genetic programming for automated refactoring and the introduction of design patterns,” in *Proceedings of the 12th annual Conference on Genetic and Evolutionary Computation*, 2010, pp. 1341–1348.
- [31] M.F. Zibrán and C.K. Roy, “Conflict-aware optimal scheduling of prioritised code clone refactoring,” *IET Software*, Vol. 7, No. 3, 2013, pp. 167–186.
- [32] S. Bouktif, G. Antoniol, E. Merlo, and M. Neteler, “A novel approach to optimize clone refactoring activity,” in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO ’06. Seattle, Washington, USA: ACM, 8–12 July 2006, pp. 1885–1892.
- [33] S. Lee, G. Bae, H.S. Chae, D.H. Bae, and Y.R. Kwon, “Automated scheduling for clone-based refactoring using a competent GA,” *Software: Practice and Experience*, Vol. 41, No. 5, 2011, pp. 521–550.

# An Approach to Assessing the Quality of Business Process Models Expressed in BPMN

Małgorzata Sadowska\*

*\*Faculty of Computer Science and Management, Department of Software Engineering,  
Wrocław University of Technology*

`m.sadowska@pwr.edu.pl`

## Abstract

**Introduction:** The quality of business process models is important in the area of model-based software development. To the best knowledge of the author there is no working practical model for quality assessment of BPMN 2.0 Process Diagrams which measures the actual models and automatically interprets the measured values.

**Objectives:** To propose a metamodel for assessing the quality of BPMN 2.0 process models and a working solution – a model for quality assessment of process models in BPMN (called MAQ) and a tool which implements MAQ.

**Methods:** The metamodel was built upon the information presented in ISO/IEC 25010 (2011) standard. The methodology of MAQ was driven by its essential elements. Quality characteristics were selected through a systematic literature review. Quality metrics were identified through a literature review restricted by questions that every relevant literature work had to affirmatively answer. Quality metrics were implemented in the tool and quality criteria were proposed based on the interpretation of the results of measuring a repository of BPMN models. Finally, quality functions were proposed and the complete MAQ was implemented in the tool.

**Conclusions:** MAQ was preliminary evaluated for usefulness through a survey-based experiment. The results showed that the model works in most cases and in general is needed.

**Keywords:** BPMN quality model

## 1. Introduction

Working with models has become a common practice in model-based software development. Models play an important role in the entire development process. In order to model business processes, various notations and languages are used, such as BPMN, UML Activity Diagram, UML EDOC Business Processes, IDEF, ebXML BPSS, Activity-Decision Flow (ADF) Diagram, RosettaNet, LOVeM and Event Process Chains (EPCs). In this paper BPMN was chosen because it is a standard notation used to model business processes [1] and it was created in such a way that it is readily understandable by all

business users, while still being able to represent complex process semantics [2].

Nowadays the need for achieving high quality BPMN models seems to be undeniable [1, 3, 4]. To start with, quality has an impact on the ease of early detection and therefore correction of BPMN models. Early discovery of defects in software artifacts is cheaper than repairing consequences of modelling errors in later design phases [3, 5]. Also, poor quality of process models can result in poor information systems [6]. Next, models of good quality are claimed to have a positive influence on reducing software maintenance costs [7]. Finally, all the mentioned aspects may lead to economic benefits through satisfaction

of user requirements for BPMN models and the resulting software.

The desire to ensure high quality in the actual BPMN models is the background underlying the idea of the model for assessing the quality of BPMN models (called MAQ). MAQ is designed for quality assessment of BPMN 2.0 Process Diagrams, but MAQ was not intended to calculate other types of BPMN models (Collaboration Diagrams or Choreography Diagrams) which can be considered a limitation of the model. MAQ considers every graphical construct for process models defined in the standard, thus it is able to calculate actual models. By “an actual model” the author understands a Process Diagram which is not limited to a truncated subset of BPMN graphical elements, represents complex process semantics and is able to graphically represent the actual business process. Supporting the quality of the so understood actual models in the opinion of the author is essential and is the aim of this paper.

A solution that seeks to help modellers in verifying the quality of actual models in an effective automated way cannot be abstract and should be easy to be directly used on actual BPMN models. Therefore, the focus of this paper is on developing a model, and more importantly, a method for assessing the quality of business process models in BPMN.

Measuring business process models is a relatively new discipline [8] even though the first version of BPMN 1.0 was released already in 2004 and the final adopted specification of BPMN 1.0 was finalized in 2006. Currently, BPMN has already been evaluated both empirically and analytically [9]. The literature describes many metrics that can be potentially used for assessing the quality of business process models. Please notice that MAQ is designed for quality assessment of the actual models which use a full range of the BPMN Process Diagram graphical constructs. The need for supporting all constructs triggered the need to apply a list of assumptions or changes in the selected metrics. This is caused by the fact that many of the original versions of metrics were designed to calculate only models with a truncated subset of elements, not actual models, which represent complex process semantics.

The rest of the paper is organized as follows: Section 2 presents related works, Sections 3-5 define a quality metamodel, the developed MAQ and the implemented tool, Section 6 summarizes preliminary evaluation of MAQ, Sections 7 and 8 present threats to validity and conclusions.

## 2. Related Work

The model for assessing the quality of business process models in BPMN was only found to be directly related to the findings of two other papers: [10] and [11].

The contribution of [11] known as the 3QM-Framework, provides quality marks, metrics and measurement procedures which mainly focus on evaluating the quality of handwritten BPMN models. The overall quality in 3QM-Framework is based on aggregation of metrics and measurement procedures and its result may vary depending on the project context. Therefore, user groups have to derive weighting of measurement. This paper differs by proposing a model for an instant and automatic assessment of quality, which is aimed to be helpful also for non-expert users. Quality marks from 3QM-Framework are referred to in Section 4.1 among other findings from systematic literature review.

Makni et al. [10] implemented a tool which can provide the results of measuring some of BPMN metrics chosen from the literature by its authors. The tool is aimed to help designers to choose a subset of metrics corresponding to design perspectives. Interpretation of the results of measurements is left to users.

A systematic literature review in the area of model quality was conducted by Mohagheghi, Dehlen and Neple [5]. The focus of this paper is set only on business process models created with the use of BPMN. The classification of model quality goals developed in [5] is referred among other literature references to in Section 4.1.

Sánchez-González et al. [4] presented a systematic review of measurements for business processes. The metrics from the review were taken into consideration while developing the

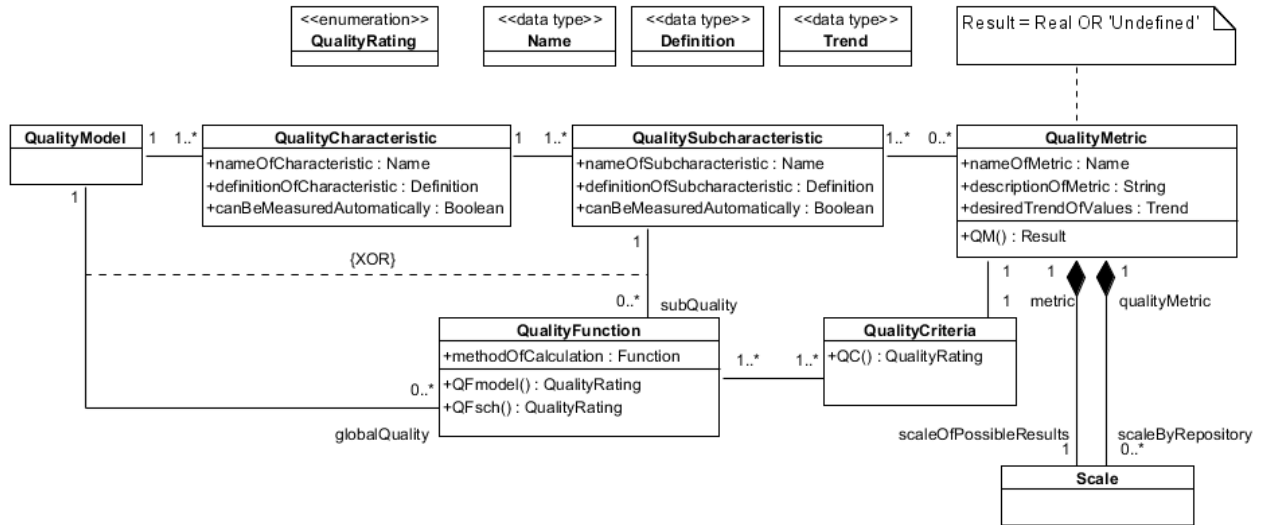


Figure 1. Metamodel for assessing the quality of business process models in BPMN

MAQ. Additional help in choosing relevant metrics came from [12]. Nonetheless, the final list of quality metrics used in the MAQ was extracted from the literature based on the proposed selection criteria listed in Section 4.2. The metrics in some cases were additionally adjusted by the author to calculate actual models in BPMN.

MAQ and the metamodel for assessing the quality of business process models in BPMN was initially developed in the author’s master thesis [13]. This article presents a reanalyzed approach to the information contained in the thesis and the improved version of MAQ, the tool that implements MAQ and the metamodel. There were many major changes and improvements in the metamodel, and some changes in MAQ and the tool. The most important change in MAQ was removing the indicators for the syntactic quality of BPMN models. All the definitions and descriptions presented in the article were rethought and reanalyzed from its initial proposition.

### 3. Metamodel for Assessing the Quality of Business Process Models in BPMN

This section introduces the proposed metamodel for assessing the quality of business process mod-

els in BPMN. The structure of the metamodel is presented in Figure 1. An example instantiation of the metamodel is MAQ (described in Section 4).

The metamodel is built upon the information presented in ISO/IEC 25010 [14] in conjunction with ISO/IEC 14598 [15] standard. Following [14], a quality model is a “defined set of characteristics, and of relationships between them, which provides a framework for specifying quality requirements and evaluation quality.” The hierarchical decomposition is the main idea of the model which is aimed to decompose quality down to a level which can be measured and thus the quality can be evaluated.

By quality characteristics, the quality of actual BPMN model can be described and evaluated. Quality characteristics are further decomposed into related quality subcharacteristics. The role of the subcharacteristics is to specify the general characteristics more concretely. Quality characteristics and subcharacteristics in the metamodel are suggested to be named and defined in a natural language.

In order to talk about measurement, both terms “metric” and “measure” are often used interchangeably by researchers [16]. In this paper the term “quality metric” is adopted. Quality metrics are aimed for measuring quality subchar-

acteristics. One metric may be assigned to more than one subcharacteristic and, as it is suggested in the standard, more than one quality metric may be used to measure a quality subcharacteristic. A scale mathematically defines theoretically possible results that a potential BPMN model can obtain for a specific quality metric, as a result from the calculation of the mathematical equation. The scale is also used to specify a scale of results obtained by a repository of actual models in BPMN – this scale is a subset of the scale of theoretically possible results. Each quality metric also owns a desired trend of values that are favorable for a metric. The trend can be described in a natural language, e.g. the lower obtained value of the quality metric by an actual BPMN model, the better quality of the model.

Quality rating defines rating levels for the measured values. In the metamodel, quality criteria are used to determine the rating levels associated with the results. The results are the obtained values on the scale of quality metric for a specific BPMN model. Finally, quality functions are used to assess the quality of quality subcharacteristics or the overall quality of the actual BPMN model. Quality functions are based on the quality criteria and either quality subcharacteristic or a quality model, which is represented by a XOR constraint in the metamodel.

## 4. MAQ

The metamodel presented in Section 3 defines the structure of MAQ. The metamodel may be used to produce other models for assessing the quality of business processes models in BPMN, MAQ is only one of the possible instantiations of the metamodel. The following subsections are aimed to present how the essential elements of MAQ were obtained.

### 4.1. Selection of Quality Characteristics and Quality Subcharacteristics

The set of quality characteristics and quality subcharacteristics subsequently determined and collected together constitute a hierarchical struc-

ture of MAQ. In order to identify characteristics, a systematic literature review (SLR) was conducted. Following [17], “a systematic literature review (...) is a means of identifying, evaluating and interpreting all available research relevant to a particular research question, or topic area, or phenomenon of interest.” The goal of this review was to provide definitions of characteristics based on the analysis of previous literature in the field. The synthesis of the literature was conducted using a well-defined methodology and search strategy with specified two research questions being addressed: RQ1: “*What quality characteristics of models exist?*” and RQ2: “*Which of the identified quality characteristics are suitable to the developing model?*”.

In order to make the process replicable, the search strategy consisted of several steps as outlined in [17]. At first, keywords were identified in order to minimize the effect of differences in terminologies. The following are the keywords that were formulated from the terms used in the domain and research questions; or their synonyms, alternate words and meaningful combinations: “BPMN,” “business process models,” “model-driven engineering,” “conceptual modelling,” “quality,” “model quality,” “quality characteristics,” “quality goals,” “quality of business process models.” The keywords were used to build search queries in order to obtain relevant articles. Six queries were based on the Boolean AND to join keywords: 1) “model quality” AND “business process models” 2) “quality characteristics” AND “business process models” AND BPMN 3) “quality goals” AND “business process models” AND BPMN 4) “quality of business process models” 5) “conceptual modeling” AND “model quality” 6) “model-driven engineering” AND quality.

The search was conducted within the following electronic databases: ACM Digital Library, SpringerLink, ScienceDirect, Emerald, Academic Search Complete, Elsevier/ICM and ProQuest. Strategy for searching was conducted in two phases. In the first phase, the chosen publication channels were searched for. After eliminating duplicates and reading titles and abstracts in all of the found papers, the literature

was chosen for further reading based on the studies selection criteria. Two inclusion criteria were applied: “paper describes quality characteristics of models” and “paper must contain the search keywords.” In spite of that, two exclusion criteria were used: “paper describes quality characteristics of software products” and “paper does not relate to Software Engineering/Development.” Finally, the full body of the filtered literature was read and the literature relevant for this systematic literature review was identified. After the first phase of searching, the second phase was initiated in order to obtain a more representative set of studies. In this phase, the reference list of all the selected literature was scanned in order to discover more papers. Lastly, if the literature was claimed to be relevant, it was found in electronic databases.

The first phase of searching resulted in 10 papers and the second search phase additionally included 7 papers. The final list of primary studies included in systematic literature review contains the following papers: [3, 5, 11, 18–30]. The description of the studies with identified quality characteristics can be found in [13].

To summarize, the primary studies describe 14 sets of quality characteristics of models. The obtained results concentrate mostly around characteristics of UML models (8 papers). The other resulting papers refer to characteristics for: conceptual models (3 papers), collaborative modelling including models (1 paper) and information models (2 papers). Only 3 papers directly discussing quality characteristics of business process models were found, however, the studies are rather recent, from the years 2010–2012.

In order to answer RQ1 and RQ2, definitions of characteristics from findings, explicitly relevant to BPMN models, were gathered together and compared against each other. The selected and systematized characteristics from the literature particularize the area of quality in order to be relevant for business process models in BPMN. Definitions of the quality characteristics and quality subcharacteristics of MAQ:

1. **“Correctness”** – in accordance with an analysis regarding making correct statements about the domain AND following

BPMN notation according to the specification, e.g. not violating rules and conventions (well formedness and syntactic correctness).

- a) **“Syntactic correctness”** – model in BPMN is syntactically correct if all terms are used in accordance with the syntax rules of the BPMN notation.
  - b) **“Semantic correctness”** – model in BPMN is semantically correct if it corresponds to the domain and the reality of the analysed situation.
2. **“Integrity”** – description of all and only relevant elements of the domain, business process and purpose of modelling.
    - a) **“Informational completeness”** – a correct scope of the BPMN model (does the model in BPMN include all and only relevant features of the domain).
    - b) **“Consistency”** – no contradictions in the model and the domain concepts are adequately represented in the model.
    - c) **“Accordance with purpose”** – it occurs when the BPMN model meets the original goals for why it was created.
  3. **“Modifiability”** – ability of the BPMN model to be modified or changed AND supporting reusability and extensibility.
    - a) **“Changeability”** – support for changes or improvements.
    - b) **“Reusability and extensibility”** – support for the model to be used in the creation of new models or extended with new terms.
  4. **“Complexity”** – related to the complexity of the BPMN model with the goal of simplicity and minimalism.
    - a) **“Minimality and simplicity”** – if the BPMN model contains the minimum possible constructs.
  5. **“Understandability”** – satisfaction of the users and their comprehensibility AND an aesthetics of BPMN model.
    - a) **“User comprehensibility”** – understandable by users – both human and tools.
    - b) **“Aesthetics of model”** – when the organization of the BPMN model improves its look in order to ease its understanding-

model improves its look in order to ease its understanding.

#### Important remarks to MAQ:

(1) MAQ considers only models with the ensured correctness. A BPMN model must be syntactically and semantically correct before the model quality can be assessed. This decision is motivated by the fact that it is useless to assess further quality factors if the model does not adhere to the syntax rules of the BPMN notation (syntactic correctness) and does not correspond to the reality of the analysed situation (semantic correctness). Similarly, as it is useless to discuss the quality of the software which does not fulfil customer requirements.

(2) MAQ only provides the characteristics of *canBeMeasuredAutomatically = false* but it does not take it into further consideration.

#### 4.2. Selection of Quality Metrics

After outlining the quality subcharacteristics (Section 4.1), some important questions arose, such as how to measure the subcharacteristics and how to interpret the measured values. In order to evaluate the quality of subcharacteristics, a set of metrics to measure models of business processes was required. The relevant metrics were identified by performing a literature review restricted by a proposed set of selection criteria. The focus of literature search was to obtain quality metric(s) for each of the quality subcharacteristics, the knowledge of how to calculate each metric and what its values mean.

For each quality subcharacteristic, a metric or a set of metrics were chosen from the literature and rationalized. The choice was based on the selection criteria, which were questions that relevant literature describing metric had to affirmatively answer. The selection criteria for the literature were as follows:

1. Is the metric useful (or can the metric be useful after changes or adjustments) for the context of modelling of business processes in BPMN?
2. Is it possible to calculate the metric for business process models in BPMN (directly or after changes or adjustments)?

3. Is the method of calculating the metric well described in the literature (or is it possible to propose a method of calculating the metric logically)?
4. Is there a general trend known from the literature which identifies a good or bad value of the metric ?
5. Do not the metrics limitations exclude it from being applicable to the relevant sub-characteristic(s)?

Only a few of the selected metrics were created for BPMN models, e.g. Control-flow Complexity metric [31] or Cross-connectivity metric [32]. Many more metrics were originally proposed in the related to BPMN areas, e.g. to measure UML models [5, 30], to measure business processes modelled in the YAWL language [33], or they were adjusted to a new purpose from mature metrics used in software engineering, especially used in object-oriented software engineering, e.g. [1, 8, 34].

The metrics selected from the literature are listed below. As previously explained, for some metrics it was necessary to introduce and apply additional assumptions, adjustments or changes in order to be able to use the metrics on the actual models in BPMN. One general assumption about the metrics calculation method from the literature was adopted in this paper. There are 5 different types of gateways in BPMN. These five types of gateways are: Exclusive Gateway, Event-based Gateway, Inclusive Gateway, Parallel Gateway and Complex Gateway. In MAQ both Data-based Exclusive Decision/Merge Gateway and Event-based Exclusive Decision/Merge Gateway are considered as XOR gateways wherever XOR gateway is stated in the definition of the metric. The distinction between data-based and event-based XOR gateways is based on whether the information required to make the decision is available within the process (data-based is used) or comes from an external source (event-based is used). However, both XORs represent a decision to take exactly one path in the flow so from the point of view of metrics in MAQ they are considered as XOR gateways and calculated in the same way.



The details of the selected quality metrics and the applied changes to their original definitions are presented below. Due to the limited space, in order to get familiar with the method of metric calculation, please refer to the indicated literature references.

### 1. Coupling metric (CP) [8]

*Short description:* CP metric calculates the degree of coupling, which is related to the number of interconnections among the tasks of a process model.

*Desired values:* Low CP values are desired. The higher coupling value of the process, the more difficult it is to change the process and the higher probability that there will be errors in the process.

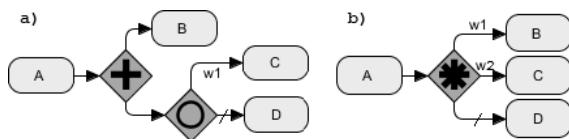
*Assumptions or changes:* The original metric [8] considers AND, OR, XOR gateways and does not provide the calculation method of *connected* function between gateways. In order to be able to calculate actual examples of BPMN models two additional assumptions to the method of calculating the metric were required:

$t_1, t_2$  – activities;  $g_1, \dots, g_n$  – gateways

a)  $connected(t_1, t_2) = 0$ , if  $(t_1 \rightarrow g_1 \rightarrow \dots \rightarrow g_n \rightarrow t_2) \wedge (g_1 \neq \dots \neq g_n) \wedge (t_1 \neq t_2)$ ,

b)  $connected(t_1, t_2) = 0$ , if  $(t_1 \rightarrow \text{Complex Gateway} \rightarrow t_2) \wedge (t_1 \neq t_2)$ .

The following examples illustrate situations where the assumptions are needed:



### 2. Control-flow Complexity metric (CFC) [31, 35]

*Short description:* CFC is an additive metric. In order to calculate the complexity of a process, one should add the control-flow complexity value of all split and join constructs.

*Desired values:* Low CFC values are desired. The greater the overall structural complexity of a process is, the higher value of the CFC will be obtained.

*Assumptions or changes:* CFC metric distinguishes between AND, OR, XOR gateways. In order to allow calculating actual BPMN models with all possible constructions, in MAQ the split for Complex Gateways results in value 0 so that it does not change the result of calculations of CFC metric for the whole BPMN model. Further research may consider changing this method of calculating Complex Gateway.

### 3. Cross-connectivity metric (CC) [32]

*Short description:* Let a process model be given by a set of nodes and a set of directed arcs. Each arc goes from a source node to a destination node. CC metric is designed to measure the strengths of arcs between model nodes. It aims to capture the cognitive effort to understand the relationship between every pair of process model elements.

*Desired values:* High CC values are desired. The more difficult it is to understand the model or the model is more likely to include errors; the lower the CC value is assigned to the model.

*Assumptions or changes:* CC metric is very sensitive to the syntactic correctness of the BPMN model. The following is a list of additional assumptions that were applied to CC metric so that the algorithm could calculate the actual BPMN models:

a) The original CC metric [32] does not address the problem of BPMN models with events and how events influence the results. In BPMN it is hard to model without events. For different types of events (start events, end events and intermediate events) the weight of node is assumed to be equal 0. Further research should provide information if this value should be different or if there should be a spectrum of weights for different event nodes.

b) The original CC metric [32] does not consider any gateway types other than AND, OR, XOR. In order to allow calculating actual BPMN models, metric result for the models with Complex Gateways is currently set as “Undefined.” Further research may consider stating this value.

c) CC metric cannot calculate business process models smaller than two elements (e.g. two tasks), however it seems to be of minor importance because the majority of models are more complex.

d) Following Vanderfeesten et al. [32], CC metric is based on the assumption that tasks in a model have at most one input and output arc while connectors can have multiple input and output arcs. Therefore, the metric result for other BPMN models is currently set as “Undefined.”

#### 4. **Imported Coupling of a Process metric (ICP) [1]**

*Short description:* ICP is a coupling metric that focuses on process if it is highly dependent on external services offered by other processes.

*Desired values:* Low ICP values are desired. The higher ICP value, the more dependent the process is on the services offered by other processes, what might increase delays, costs and error probability.

*Assumptions or changes:* Metrics in MAQ ought to provide a value for the whole BPMN model. The original ICP metric by Khelif et al. [1] calculates the result for each single task or sub-process in BPMN model. In MAQ, ICP metric for the whole business process model is defined as the greatest ICP value obtained by any of its tasks or sub-processes. Additionally, in order to properly calculate ICP values for models in BPMN, associations and data associations should also be included. Therefore the changed metric counts the sent message flows, sequence flows, associations and data associations.

#### 5. **Exported Coupling of a Process metric (ECP) [1]**

*Short description:* ECP is a coupling metric that focuses on process and its influence on the whole model based on how many other processes depend on its services.

*Desired values:* Low ECP values are desired. The higher ECP value, the more other processes depend on the services of the process, which might increase delays, costs and error probability.

*Assumptions or changes:* The assumptions for ECP are nearly the same as the assumptions for ICP, but for the fact that the changed metrics counts the received message flows not the sent message flows.

#### 6. **Fan-in/fan-out metric (FIO) [36]**

*Short description:* FIO metric can be used to analyse the complexity of a business process model based on the modular structure. Modular modelling is supported in BPMN by sub-processes. The metric is similar to the metric proposed by Khelif et al. [1], however, it does not include length.

*Desired values:* Low FIO values are desired. The higher structural complexity of a model or sub-model according to the FIO value, the more difficult it is to use the model and there is more likelihood that it is badly designed.

*Assumptions or changes:* Metrics in MAQ ought to provide a value for the whole BPMN model. The original FIO metric [36] counts only sub-processes (it does not count tasks) and calculates a separate result for each single sub-process in BPMN model. In MAQ, FIO metric for the whole business process model is defined as the greatest FIO value obtained by any of its sub-processes. Due to the fact that from the definition of the metric it is not clear what should be adopted if the model does not have a modular structure, if the model does not have sub-processes, in MAQ FIO value is assumed to be equal zero.

#### 7. **Number of Activities, Joins and Splits (NOAJS) [37]**

*Short description:* Splits in BPMN do not necessarily have corresponding joins. NOAJS complexity metric can measure such not well structured processes based on counting activities, joins and splits together.

*Desired values:* Low NOAJS values are desired.

*Assumptions or changes:* None.

#### 8. **Interface complexity of an activity metric (IC) [37]**

*Short description:* IC metric can be used to evaluate the complexity of processes.

*Desired values:* Low IC values are desired.

*Assumptions or changes:* From the original definition of the metric [37], it is not clear how *Length* should be calculated for BPMN models. In MAQ, it is calculated as follows: *Length*=1 for a task element and *Length*=3 for a sub-process (representing sub-processes as a collection of activities).

Metrics in MAQ ought to provide a value for the whole BPMN model. The original IC metric [37] calculates a result for each single activity. In MAQ, IC metric for the whole model is defined as the sum of all IC values obtained by all activities in the model. These will reduce a limitation of the original metric, which can give the zero result as the value of complexity if an activity has no external interactions, e.g. for the end activities of the process.

*Assumption for the following points 10, 11 and 12 describing Halsted-based Process Complexity metrics:* The following refinement of the metrics is used in MAQ:

$n_1$  – number of unique activities, splits, joints and control-flow elements.

$n_2$  – number of unique data objects, data inputs, data outputs and data stores (duplicates removed).

$N_1$  – number of unique types of activities and control-flow elements used in BPMN model, e.g. task, sub-process, XOR gateway, OR gateway, etc.

$N_2$  – number of unique data types used in the BPMN model – data objects, data inputs, data outputs and data stores.

9. **Halsted-based Process Difficulty metric (HPC\_D)** [37]

*Short description:* HPC\_D is a quantitative measure of complexity and is aimed to calculate the difficulty of the process.

*Desired values:* Low values are desired.

*Assumptions or changes:* HPC\_D has a limitation, because its value cannot be calculated if  $n_2$  equals 0. In MAQ, the result of such calculation is set as “Undefined.”

10. **Halsted-based Process Length metric (HPC\_N)** [37]

*Short description:* HPC\_N is a quantitative measure of complexity and is aimed to calculate the length of the process.

*Desired values:* Low values are desired.

*Assumptions or changes:* HPC\_N metric can be calculated only if ( $n_1 > 0$  and  $n_2 > 0$ ), otherwise it cannot be calculated because the log value is undefined. In MAQ, the result of such calculation is set as “Undefined.”

11. **Halsted-based Process Volume metric (HPC\_V)** [37]

*Short description:* HPC\_V is a quantitative measure of complexity and is aimed to calculate a volume of the process.

*Desired values:* Low values are desired.

*Assumptions or changes:* HPC\_V metric can be calculated only if ( $n_1 + n_2 > 0$ ), otherwise it cannot be calculated because the log value is undefined. In MAQ, the result of such calculation is set as “Undefined.”

12. **Sequentiality metric (S(G))** [12]

*Short description:* S(G) is a structural metric. The sequentiality ratio is the number of arcs between none-connector nodes divided by the number of arcs.

*Desired values:* High S(G) values are desired. The higher S(G) value, the less likely it is to have errors in the overall model.

*Assumptions or changes:* None.

13. **Number of Nodes metric (Sn(G))** [12]

*Short description:* Sn(G) is a structural metric that calculate the number of nodes of process model.

*Desired values:* Low Sn(G) values are desired. The higher Sn(G) value, the more likely it is to have errors in the overall model.

*Assumptions or changes:* None.

14. **Number of Activities metric (NOA)** [37]

*Short description:* NOA metric sums up activities in a business process model. It is a simple and popular metric that can be used to measure complexity.

*Desired values:* Low NOA values are desired.

*Assumptions or changes:* None.

15. **Coefficient of Connectivity metric (CNC(G))** [12]

*Short description:* CNC(G) is a structural metric. The coefficient of connectivity gives the ratio of arcs to nodes in BPMN models.

*Desired values:* Low CNC(G) values are desired. The higher CNC(G) value, the more likely it is to have errors in the overall model.

*Assumptions or changes:* None.

#### 16. Cognitive complexity measure (W) [33]

*Short description:* Cognitive complexity measure is a cognitive weight proposed to measure the effort needed for comprehending the model.

*Desired values:* Low W values are desired. The higher W value, the more difficult it is to understand the model.

*Assumptions or changes:* Cognitive weights of business process model elements in [33] were proposed for YAWL language. Based on the analogy with BPMN language, the adequate cognitive weights for BPMN models are proposed in Table 1. In MAQ, the cognitive weight of the BPMN model is defined as a sum of the cognitive weights of its individual elements.

#### 17. Density metric (D(G)) [12]

*Short description:* D(G) is a structural metric that calculates the ratio of the total number of arcs to the maximum number of arcs.

*Desired values:* Low D(G) values are desired. The higher D(G) value, the more likely it is to have errors in the overall model.

*Assumptions or changes:* None.

The chosen quality metrics were assigned to quality subcharacteristics of MAQ based on information derived from the literature. Some metrics are useful for more than one subcharacteristic (please refer to metamodel in Fig. 1).

**Rationale for assigning metrics to subcharacteristics is as follows:**

*Quality subcharacteristic: “Changeability”* has the following metrics assigned:

- CP: The lower value of coupling, the easier to change the process [8].
- CFC: Models with a reasonable complexity are easier to modify and maintain. The metric may help to develop simpler processes when it is possible [35]. Following [38], CFC metric is suitable to measure changeability.

- D(G) and S(G): In [4], conducted experiments showed that Density and Sequentiality metrics are closely connected with modifiability.

- HPC\_D, HPC\_V and HPC\_N: Metrics can predict maintenance effort [37].

- ECP and ICP: Business process models that have high coupling metric are difficult to be changed or maintained because they have a high level of informational dependency between activities [1].

*Quality subcharacteristic: “Reusability”* and extensibility has the following metric assigned:

- FIO: In accordance with [36], FIO metric detects poor modularization. If modularization is used in a reasonable way, dividing a model in modular sub-models can lead to smaller, reusable models.

*Quality subcharacteristic: “Minimality and Simplicity”* has the following metrics assigned:

- FIO: Following [36], if the examined sub-process in the model has both a large fan-in and a fan-out, this may indicate that the model does not have an appropriate size or was not partitioned into modules in a sensible way. Redesigning in this situation could improve the sub-process.

- NOA and NOAJS: These simple metrics may show models that are badly designed with an excessive number of activities [1].

*Quality subcharacteristic: “User comprehensibility”* has the following metrics assigned:


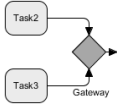
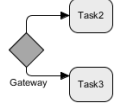
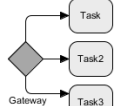
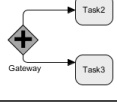
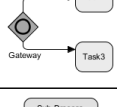



- CFC: It is easier to understand and maintain business process models with low complexity. Business processes should minimize their complexity in order to be helpful to the various stakeholders [35]. Following [38], CFC metric is suitable to measure understandability.

- CC: Models with high cross-connectivity can facilitate understanding of business processes among various stakeholders [32].

- NOA and NOAJS: The metrics provide some information about the understandability of designs [37].

- W: The measure can state whether models are easy or difficult to comprehend [33, 38].

Table 1. Proposition of cognitive weights for BPMN models in Cognitive Complexity Measure

BPMN structure	BPMN symbol	Cognitive weight
Single consecutive step in a work-flow		<b>1</b>
All joins. In [33], the metric was originally defined only for business process models that are well-structured. In BPMN, corresponding joins are not necessary. The weight of join elements is considered as equal to the cognitive weight of sequence elements.		<b>1</b>
XOR-split (exactly one of two branches is chosen)		<b>2</b>
XOR-split (exactly one of more than two branches is chosen)		<b>3</b>
AND-split		<b>4</b>
OR-split or Complex Gateway		<b>7</b>
Sub-process (can be used for decomposing BPMN models)		<b>2</b>
Start or End event		<b>2</b>
Intermediate event (both intermediate events attached to the boundary of activities and intermediate events within the normal flows)		<b>3</b>

- FIO: The metric was developed for analysing the modularization; modular sub-processes can help to make the model easier to comprehend [36].
- Sn(G), CNC(G) and S(G): In [4], the conducted experiments showed that the metrics are closely connected with user's understandability.
- CP: High complexity in a process may result in bad understandability, therefore, process complexity should be kept at low level [8].
- IC: The metric is a measure of complexity of process models and complexity measures the understandability of a design [1].

*Quality subcharacteristic: "Aesthetics of diagrams"* has the following metrics assigned:

- CP: Business process models with high CP metric have complicated connections, which can be reflected in the organization of BPMN models [8].

- ICP and ECP: The organization of BPMN models with high ICP or ECP metrics may not be clear and thus difficult to understand. The coupling metrics detect models in which multiple processes depend on each other, which may influence the look of the whole design.
- CNC(G): In formal esthetics the coefficient of network complexity measure is considered with the notion of elegance [37].

Figure 2 presents a schema of the hierarchical structure of the extracted quality characteristics, quality subcharacteristics and quality metrics.

### 4.3. Selection of Quality Criteria

Very rarely does it happen that the literature indicates which values of metrics are good or bad, and an accurate analysis of the results is mostly left to the user. This is not a problem if the user

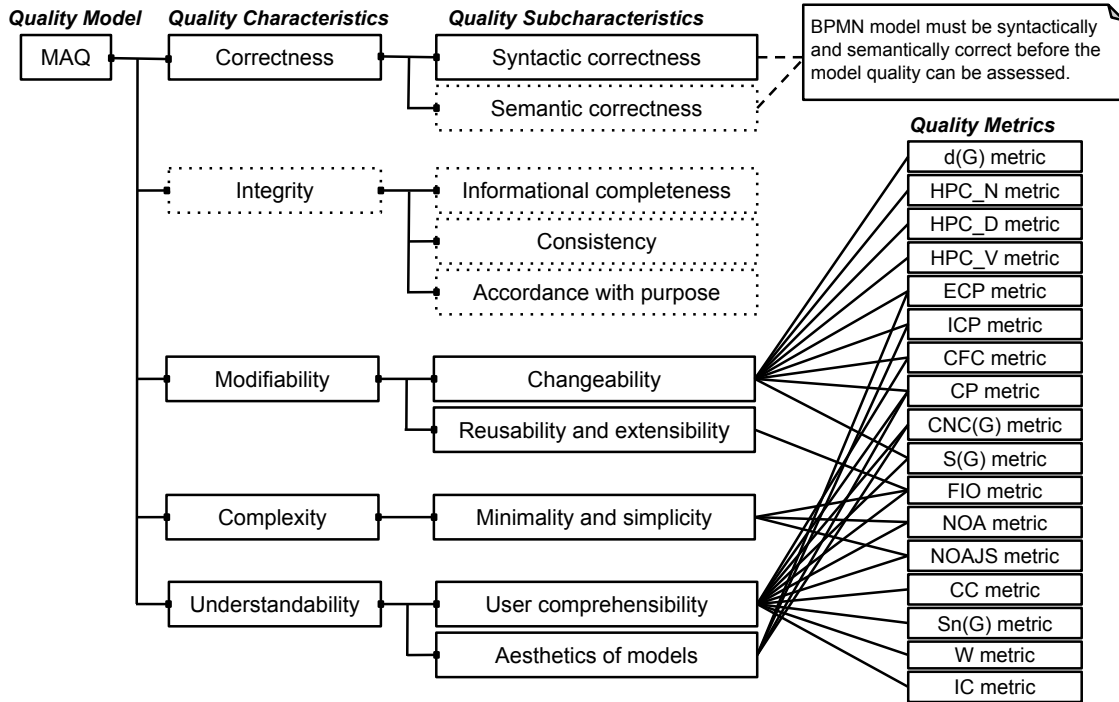


Figure 2. A schema of the hierarchical structure of MAQ

is an expert and a quick analysis of multiple metrics and models is not required. The purpose of MAQ is to automate the process of model quality assessment. Therefore, it is very important to define exact quality criteria as functions that appraise the results of quality metrics.

More often than specific numbers, the authors of metrics indicate the general trend of metrics' results, e.g. a lower (higher) value of a metric, a better model. Therefore, one of the selection criteria for the literature in Section 4.2 rejected literature and metrics for which this trend was not clear. With this knowledge in mind, quality criteria for metrics based on the results obtained from measuring models from a pre-prepared BPMN repository. The repository contained 57 business process models in BPMN; collected from five different Internet sources [13]. The identified BPMN models had varying quality in different sources because they were created by users with different levels of experience in BPMN. The repository contained officially correct BPMN examples given by the OMG, models from master and doctoral theses and models created by various individuals, who had less experience with BPMN. This variety of

models helped to define which results of metrics were obtained by high and low quality models. An effort was made to collect models of diverse quality, however it poses a threat to validity. In order to be able to examine the repository of BPMN models and to propose quality criteria, two tools were needed:

1. A tool that implements all the chosen quality metrics from Section 4.2. This tool was created and is reviewed in Section 5.
2. Additional statistical software which contains tools for clustering. In this case Weka software was chosen and simple  $k$ -means function was selected for clusterization.

The algorithm of  $k$ -means clustering was developed by Hartigan and Wong [39]. In MAQ, in the  $k$ -means clustering function, the  $k$  value was declared as equal 4 or 2, based on the results of metrics used on the repository. The seed value for each metric was chosen individually as integer value without rounding from the equation: maximal metric's value minus minimal metric's value divided by 2.

**Quality rating** in MAQ is defined as an ordinal scale that describes whether the result of the metric is of good or bad quality on a scale of

*Class A* (highest) through *Class E* (lowest). The chosen scale is ordinal since the quantitative levels of quality had varying distances between them for each metric. For example, the range of values obtained from the BPMN repository for CP metric had a range of 0 to 0.3 and the CFC metric had a range from 0 to 16. Clearly, the ranges between metrics were different in practice and not easily comparable without the use of an ordinal scale. The ordinal scale was created using results of measurements of the repository. The values for each entity of the ordinal scale were based on an interval of values which were relevant to each metric. For example, the CFC metric had an observed range from 0 to 16. The trend of values for each metric suggests what are good or bad values in terms of quality for each metric. This information was used to assign intervals to quality ratings. For example, in the case of CFC metric, the value should be low in order to attain a good quality. Clusterization of the metric was then used to create intervals of ordinal elements, e.g. zero to one for *Class A*, from one to four for *Class B*, etc.

Example calculations of quality criteria presented below are based on CP metric. A full list of the defined quality criteria is available in [13].

Summary of CP metric:

- Type of measurement method: *Objective*,
- Scale of theoretically possible results: *Real from zero to infinity*,
- Scale of results obtained by models from the repository: [0.0, 0.333333],
- *Low CP values are preferable* for high quality models.

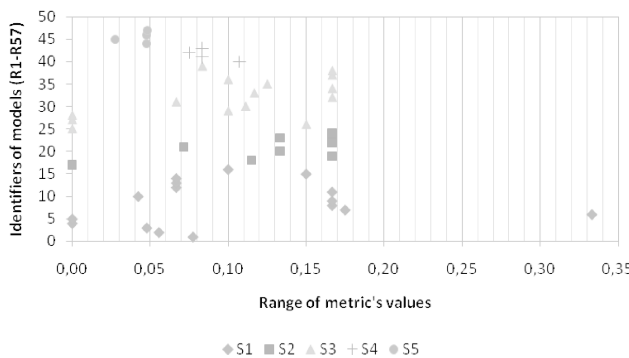


Figure 3. The result of measures of CP metric on the repository of BPMN models

Weka software settings: Simple  $k$ -means function, Number of clusters: 4, Seed: 0. Clusters obtained through the use of the software were as follows: *cluster 0*: 0.114167, *cluster 1*: 0.176786, *cluster 2*: 0.064361, *cluster 3*: 0.003934.

Table 2. Assignment of results to quality ratings

Range of results	Quality Rating
[0.0, 0.003934)	Class A
[0.003934, 0.064361)	Class B
[0.064361, 0.114167)	Class C
[0.114167, 0.176786)	Class D
[0.176786, $\infty$ )	Class E

The obtained quality criteria for CP metric:

$$QC() = \begin{cases} \text{Class A, if } CP \in [0.0, 0.003934) \\ \text{Class B, if } CP \in [0.003934, 0.064361) \\ \text{Class C, if } CP \in [0.064361, 0.114167) \\ \text{Class D, if } CP \in [0.114167, 0.176786) \\ \text{Class E, if } CP \in [0.176786, \infty) \end{cases}$$

#### 4.4. Selection of Quality Functions

Quality functions combine the results of quality criteria for both quality subcharacteristics and the overall quality of actual BPMN models. In this way, they indicate whether the model quality is good or bad. More specifically, the result of quality function for e.g. “*Minimality and Simplicity*” subcharacteristic has to combine the results of quality criteria for FIO, NOA and NOAJS metrics. Based on this example, the quality function determines what should be stated as an overall quality if, for example, criteria for FIO results in *Class B*, NOA in *Class C* and NOAJS in *Class B*.

There are many possible interpretations of how to propose quality functions. For example, the function for a subcharacteristic could be calculated as follows:

- The best quality rating obtained by any of metrics assigned to the subcharacteristic  $QF_{sch} = \min \{ \text{QualityRating} \}$
- The ceiling of the mean quality rating obtained by metrics assigned to the subcharacteristic. Let *Class A* = 1, *Class B* = 2, *Class C* = 3, *Class D* = 4, *Class E* = 5

$$QF_{sch} = (\text{QualityRating}) \lceil \sum_{m=1}^M QR_m / M \rceil$$

where M is a number of the assigned metrics  
– etc.

In MAQ, quality functions were proposed taking into account the following issues:

a) Not every quality metric can be calculated for each actual BPMN model. Some metrics may result in an “Undefined” value. Hence the need for differentiation in interpretation between metrics which always result in a real value (ECP, ICP, d(G), Sn(G), HPC(V), CNC(G), CFC, W, FIO, NOAJS, NOA, IC) and metrics that may result in an “Undefined” value (HPC(D), HPC(N), S(G), CC, CP).

b) Metrics that result in an “Undefined” value for an actual BPMN model should be excluded from the calculation of quality and only metrics which result in a non-undefined result should have influence on the quality.

c) The proposed quality functions use a Fibonacci sequence and the ceiling function. The Fibonacci sequence may help in addressing the differences between the results of metrics whose values are not easy to be directly compared. The distance between quality ratings varies depending on quality criteria. Fibonacci sequence seems to be relevant since the direct comparing of quality rating as ratios of each other (e.g. *Class E* as half of the quality of *Class D*) would overestimate the result. It seems to be relevant also because when the quality is low, it is important that this is clear for the user. The Fibonacci sequence increases rapidly from the initial value of 1 to 8. As a result, the quality rating for a bad quality model can be represented using higher values such as 8 in order to make the whole quality function more sensitive to bad quality. In order to make the rating more sensitive to a bad quality, the Fibonacci sequence is used starting from the third value. To summarize, Fibonacci sequence and ceiling function are chosen because they are more informative to have results that are sensitive to a low quality. These more sensitive results show clearly when the quality is low. Nevertheless, at this stage of research in the field it is difficult to assess if this interpretation is acceptable. Further research should inves-

tigate which interpretation of the quality function is best for combining metrics for models in BPMN.

Quality functions for quality subcharacteristics are defined as follows (the quality function for the whole BPMN model is analogical):

$$QF_{sch}() = (\text{QualityRating}) \lceil \frac{\sum_{m=1}^M QR_{value}(m)}{M} \rceil$$

where:

$$QR_{value}(m) = \begin{cases} 1, & \text{if } QC(m) = \text{Class } A \\ 2, & \text{if } QC(m) = \text{Class } B \\ 3, & \text{if } QC(m) = \text{Class } C ; \\ 5, & \text{if } QC(m) = \text{Class } D \\ 8, & \text{if } QC(m) = \text{Class } E \end{cases}$$

*m* – quality metric assigned to the quality subcharacteristic, which produced a non-undefined result for the measured BPMN model;

*M* – number of the assigned quality metrics;

(QualityRating) – The result of the equation is transferred into a adequate quality rating with casting so that the lower (worse) value of quality rating is assigned. An example: *Class A* = 1, *Class C* = 3, *Class E* = 8 results in:  $QF_{sch}() = (QR) \lceil \frac{1+3+8}{3} \rceil = (QR) \lceil 4 \rceil = \text{Class } D$ .

## 5. BPMN Quality Tool

*BPMN Quality Tool*<sup>1</sup> is a plug-in implemented in the Java language to Business Process Visual ARCHITECT (Simulacian) – well-known software for modeling in BPMN (tested in the 4.0 version of the software).

### 5.1. Initial Functionality of the Tool

*BPMN Quality Tool* in its initial functionality allows for measuring and displaying values of quality metrics for actual BPMN models (an example is presented on the left side of Fig. 4). This functionality was used to gather data needed to propose quality criteria for metrics (described in Section 4.3).

An additional functionality of the plug-in, available through a pop-up menu, shows relationships of BPMN elements in actual models.

<sup>1</sup> The plug-in is available online and can be found in: <<https://sourceforge.net/projects/bpmn-quality/>>.



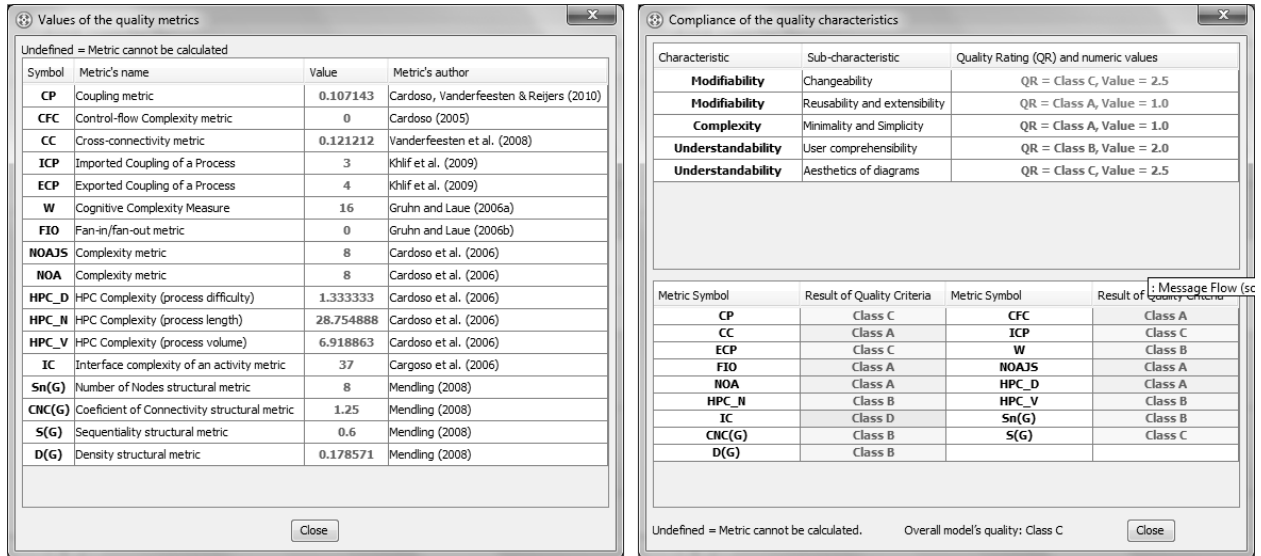


Figure 4. An example of measures of MAQ on a BPMN model

“Show Relationship of Element” option lists all relationships of the chosen element (example can be found in Fig. 5). It provides information especially about:

- the type of flow going to or from the chosen element (it distinguishes between a sequence flow and a message flow),
- the name of the flow (if the flow has a name, otherwise “Unnamed”),
- the direction of the flow (if flow goes “To” or “From” the chosen element),
- the icon of the BPMN model’s element to or from which the flow goes.

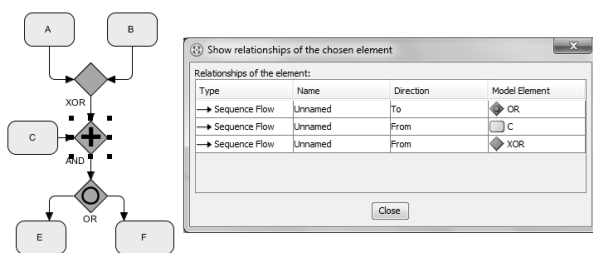


Figure 5. An example use of “Show Relationships of Element” option for the AND gateway element

This functionality may help to get more information about elements in complex BPMN models or in models with bad aesthetics, e.g. where arches cross. The analysis of the relationships of elements was a base to implementation of quality metrics presented in Section 4.2.

## 5.2. Implementation of MAQ

The final functionality of *BPMN Quality Tool* allows for assessing the quality of actual business process models in BPMN. “Quality Assessment” option is an implementation of the developed MAQ. The option shows a quality of sub-characteristics and an overall quality (example is presented on the right side of Fig. 4).

## 6. Preliminary Evaluation of MAQ

This section describes the process of how the preliminary evaluation of MAQ was conducted by a survey-based experiment and what results were obtained. The survey can be found in [13] in Appendix E.

### 6.1. Survey Study Design

*Research question:* The survey and survey-based experiment was aimed to provide an answer to the question “**Is the developed model for assessing the quality of business process models in BPMN considered useful?**”.

*Types of questions:* The survey was based on the questionnaire which consisted of closed questions. This form was chosen because it allowed for a more quantitative feedback. In some sur-

veys the respondents left additional comments. The comments mostly added details to the questions of the survey. The most interesting suggestions were about some additional functions the tool for assessing the quality of the models should have so that it would be useful for modelers. And there were also very important comments which helped to improve the definitions of the characteristics. These responses were later analyzed in a qualitative manner.

*Population of the survey:* The survey population consisted of experts on BPMN who used BPMN at work, for research or for private purposes. The author identified potential experts to be contacted, however, the final classification if someone is or is not an expert was based on how high the respondent rated his or her knowledge of BPMN (an additional question in the survey).

The initial question in the survey was: "Please select a number which best describes the level of your knowledge of BPMN notation" on a five point scale, where 1 meant a novice and 5 meant an expert. Experts in the survey were respondents who declared the level of their BPMN knowledge as 3 or more, as well as ticked that they used BPMN in their work, research or for private purposes.

In total 14 expert responses were obtained from 125 potential experts contacted.

## 6.2. Survey-based Experiment

*Objective and Design:* The aim was to evaluate practical usefulness of the MAQ model. This was done by comparing assessment given by the tool which implemented MAQ and the assessment given by the expert respondents. It was checked if the respondents' evaluation of the quality of BPMN models agreed with the results determined by the tool. This indicated how useful MAQ and the tool for automatic assessment of the quality of business process models in BPMN were. The respondents assessed three BPMN models based on the identified quality subcharacteristics. The subcharacteristics for each BPMN model were assessed using the previously introduced quality rating from *Class A* to *Class E*. The same process was conducted by

the tool. Later, results were compared and analysed. The goal of the experiment was defined according to the goal template in [40] as follows:

**analyse** the quality of BPMN models  
 [for the purpose of] the evaluation of MAQ  
**with respect to** its accuracy in the evaluation of quality of business process models in BPMN  
**from the point of view** of BPMN experts  
**in the context of** the business process modelling domain.

*Objects:* The objects were BPMN models.

*Subjects:* Responses from expert respondents.

*Independent variables:* Independent variables were three models in BPMN. The BPMN models for the survey were chosen in order to be of either good or bad quality. The chosen models of good quality were visually different. The selected models seemed to be relevant since they represented both good and bad quality models. The fact that the choice of the models was based on the author's knowledge and the chosen models could possibly not be representative for the whole population of business process models in BPMN posed a threat to validity (please refer to Section 7). Besides quality, a number of other concerns were taken into consideration when selecting the BPMN models for the survey. Firstly, the models needed to be non-trivial by representing processes which could be of importance. For example, the three BPMN models represented a trouble ticket system, a purchase ordering process and a software upgrade process. Secondly, the BPMN models needed to be reasonably complex, so that the respondents could easily understand them. This was seen in the models as they did not consist of more than 50 elements (including flows). Thirdly, the number of BPMN models chosen for the survey had to be limited in order to increase the response rate of the survey – more models could discourage the respondents. All of these reasons contributed to the choice of three non-trivial and appropriately complex BPMN models.

*Dependent variables:* The assessment by the tool and the assessment by experts who responded to the survey were the two dependent variables of the experiment. Dependent variables of the ex-

periment were selected in order to understand the correlation between the respondent's ratings of the quality versus the tool's use of the quality rating scale.

*Hypotheses:* Null hypothesis H0: The mean of the survey result for each characteristic was equal to the MAQ model's result.

Alternative hypothesis H1: The mean of the survey result for each characteristic was not equal to the MAQ model's result.

The hypotheses were tested using a student's *t*-test and the 95% confidence interval of the mean of the expert responses.

### 6.3. Results of the Preliminary Evaluation

The graphical charts that present the comparison of expert responses with tool response are available in Section 10.2 of [13]. In the survey-based experiment, the equality of the mean of expert's assessment of the quality of the surveyed BPMN models with that of the tool was mostly not rejected. The results of the MAQ model fell within the confidence interval for the characteristics of "Changeability" and "User comprehensibility" for all three models indicating that the equality of expert's responses and the responses of the tool cannot be rejected for the models surveyed. The hypothesis of the expert and tool agreement was rejected in one of the three models for "Aesthetics of the model." However, "Reusability and extensibility" was rejected in two of the three models and "Minimality and simplicity" was rejected in all models, indicating that they mostly did not equal the response of experts. The reason mostly lay in the fact that currently there is very little research on metrics that can calculate BPMN models and be proper indicators for the "Minimality and simplicity" subcharacteristic (only 3 relevant metrics were found) and the "Reusability and extensibility" subcharacteristic (only 1 relevant metric was found and the metric takes into consideration only the models with sub-processes). The MAQ is built in a way which is easily extensible if future research proposes updated or new metrics relevant for the subcharacteristics. Due to the fact that the number of the collected responses to the

survey cannot be accepted as representative, the obtained results may be used only as suggestions for the direction of the research, if it could be correct and helpful. It is identified as a threat to validity in Section 7.

## 7. Threats to Validity

The author has identified a number of threats to validity. The following is the explanation of them and their mitigating factors to the developed model. Threats to validity for research in the field of software engineering are presented as consisting of four types, which are: construct validity, conclusion validity, internal validity and external validity [40]. For each type of validity threat, risks which could pose a threat to the validity of MAQ are identified.

**Conclusion validity threats** are issues which affect the way conclusions are made from treatments.

*Reliability of measures:* When measurements are not consistently applied there is a possible risk of threatened validity. The created tool assured that all metrics, quality criteria and quality functions were calculated automatically so that the threat that calculations would be unreliable was mitigated.

*Random heterogeneity of BPMN models:* Variation poses a risk when the objects under study are heterogeneous. The BPMN models needed to be heterogeneous in terms of quality since good and bad quality was under assessment. In order to mitigate the consequences of heterogeneous BPMN models, the author narrowed the focus to Process Diagrams in BPMN 2.0 notation.

**Internal validity threats** are concerned with whether the relationship between the treatment and outcome is causal. This means that the relationship between the treatment and outcome cannot be caused by some unknown factor.

*Quality assurance of SLR:* A threat to validity is posed for the systematic literature review by the a lack of quality assurance activities. The activities could include reviewing the selected papers to see if they match the inclusion criteria by another reviewer, developing and verifying the pro-

tocol with another reviewer to make sure if the extracted data are correctly interpreted.

*Literature review for metrics:* The choice of the literature was restricted by the proposed selection criteria for the literature. There is a threat to validity that an important work could have been omitted.

*Selection of BPMN models:* BPMN models were selected by the author of the article. The models were selected from publicly available sources where the licensing allowed for them to be used for research. This poses a threat to validity since models which are licensed in a research-friendly way could be different than BPMN models in general. In order to mitigate this, 57 models from 5 different sources were collected so that a more general selection could be achieved. Furthermore, a number of models needed to be redrawn, so they could have a file format supported by a tool created by the author. The author tried to rewrite the models without defects, however, rewriting models always possibly may increase defects.

*Selection of BPMN models in survey:* There were three models chosen to be used in the survey. They were selected based on their varying quality in respect to the quality subcharacteristics measured. By selecting only three models of the 57 models there is a threat to validity that the models which were selected were not representative of the whole population of business process models.

*Selection of experts to the survey:* Naturally there is a variation in the level of expertise in the field of BPMN, and so the contacted experts may not be representative of the whole BPMN expert population. The author tried to mitigate this threat by contacting experts directly and also verifying through the survey whether they really were experts.

**Construct validity threat** refers to the theory and the observation if they are related in a causal way.

*Lack of metric validation:* If the metrics used in the paper have not been validated theoretically or empirically and were used in the MAQ model. Some metrics were additionally adjusted to the need of MAQ by the author. The selected metrics

come from scientific research and so that the selection was based on peer-reviewed metrics. Nevertheless, it poses a threat to the validity of the model.

**External validity threats** are concerned whether the result of the research is generalizable to a larger scope.

*Interaction of selection and treatment:* This is a threat when the subject of a study is not representative of the general population. Quality characteristics and subcharacteristics were selected using a SLR with a well-defined protocol that followed systematic guidelines [17]. Furthermore, the metrics were selected using a defined set of selection criteria that was applied to the literature. As a result, a well-defined methodology was applied in order to collect quality characteristics and metrics from the population of scientific literature relevant to BPMN models.

*Generalization of survey responses:* There is a serious risk that the survey respondents are not generalizable to the population of practitioners when the used sample used is not representative. The population used cannot be considered as representative for the general population. Therefore, the results obtained from the survey-based experiment may be used only as suggestions showing whether the direction of the research is correct and helpful for practitioners but the cannot be considered a statistically significant result.

*Generalization of BPMN models:* Threats to validity resulting from BPMN models not being generalized to the population pose a risk since the result can only be generalized to an appropriate scope. The repository of BPMN models consisted of only BPMN 2.0 Process Diagrams. This means that the conclusions cannot be generalized to other types of BPMN diagrams.

## 8. Conclusions

The quality of business process models is important in the area of model-based software development. The need for high quality of models is supported by many arguments both industrial and research based. This paper focuses on a practical

proposal of a model for quality assessment of actual models in BPMN called MAQ. The first part of the paper presents a metamodel of the MAQ. The metamodel defines a structure of the MAQ and is built upon the information presented in ISO/IEC 25010 [14] standard. Later on, all parts of the MAQ are described. The most important MAQ parts are: quality characteristics, quality subcharacteristics, quality metrics, quality criteria and quality functions. Later section shows BPMN Quality Tool which implements MAQ and can be helpful for modelers to ensure that the generated actual BPMN models are correct and properly built. The model was preliminarily evaluated for usefulness through a survey-based experiment in [13].

MAQ aims to assess only the quality characteristics which can be measured in isolation from any additional information about the domain, but the BPMN model itself. Therefore, MAQ only lists but is not designed to automatically assess subcharacteristic of *Syntactic correctness*, characteristic of *Integrity* and its subcharacteristics: *Informational completeness*, *Consistency* and *Accordance with purpose*. This might be considered as a limitation of MAQ.

Not all of the metrics chosen from the literature were validated. Additionally, some of the original metrics operated or were tested only on a subset of BPMN elements. Therefore, some additional changes to the original metrics metrics had to be applied in order to be able to measure actual models in BPMN. These changes have not been validated yet either. Future research may validate, change or extend the proposed metrics.

The MAQ and its implementation seem to be a good starting point for further development. The MAQ and *BPMN Quality Tool* can be further extended while new metrics will be introduced, existing metrics will be further developed in order to be able to measure actual models, and new quality criteria or quality functions will be suggested. This may lead to consideration of new perspectives and more compatible correlation between quality characteristics, quality subcharacteristics, quality metrics, quality criteria and quality functions.

## References

- [1] W. Khlif, L. Makni, N. Zaaboub, and H. Ben-Abdallah, "Quality metrics for business process modeling," in *Proceedings of the 9th WSEAS international conference on Applied computer science (ACS'09)*, R. Revetria, V. Mladenov, and N. Mastorakis, Eds. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2009, pp. 195–200.
- [2] Business process model and notation (BPMN) version 2.0. Object Management Group, Inc. (2011). [Online]. <http://www.omg.org/spec/BPMN/2.0/>
- [3] H. Reijers, J. Mendling, and J. Recker, "Business process quality management," in *Handbook on Business Process Management 1*, ser. International Handbooks on Information Systems, J. vom Brocke and M. Rosemann, Eds. Springer Berlin Heidelberg, 2015, pp. 167–185.
- [4] L. Sánchez-González, F. García, J. Mendling, F. Ruiz, and M. Piattini, "Prediction of business process model quality based on structural metrics," in *Conceptual Modeling – ER 2010*, ser. Lecture Notes in Computer Science, J. Parsons, M. Saeki, P. Shoval, C. Woo, and Y. Wand, Eds. Springer Berlin Heidelberg, 2010, Vol. 6412, pp. 458–463.
- [5] P. Mohagheghi, V. Dehlen, and T. Neple, "Definitions and approaches to model quality in model-based software development – A review of literature," *Information and Software Technology*, 2009, pp. 1646–1669.
- [6] T. Rozman, G. Polancic, and R.V. Horvat, "Analysis of most common process modelling mistakes in BPMN process models," 2007. [Online]. <http://www.slideshare.net/tomirozman/eurospi2007trozman>
- [7] I. Dubielewicz, B. Hnatkowska, Z. Huzar, and L. Tuzinkiewicz, "Quality-driven software development for maintenance," in *Emerging Technologies for the Evolution and Maintenance of Software Models*, J. Rech and C. Bunse, Eds. Hershey: Information Science Reference, 2012, pp. 1–31.
- [8] J. Cardoso, I. Vanderfeesten, and H.A. Reijers, "Computing coupling for business process models," 2006. [Online]. <https://eden.dei.uc.pt/~jcardoso/Research/Papers/Old%20paper%20format/Caise-19th-Coupling-Cardoso-Vanderfeesten.pdf>
- [9] G. Aagesen and J. Krogstie, "Analysis and design of business processes using BPMN," in

- Handbook on Business Process Management 1*, ser. International Handbooks on Information Systems, J. Brocke and M. Rosemann, Eds. Springer Berlin Heidelberg, 2010, pp. 213–235.
- [10] L. Makni, W. Khlif, Z.H. Nahla, and H. Ben-Abdallah, “A tool for evaluating the quality of business process models,” 2010. [Online]. <http://subs.emis.de/LNI%20/Proceedings/Proceedings177/234.pdf>
- [11] S. Overhage, D.Q. Birkmeier, and S. Schlauderer, “Quality marks, metrics, and measurement procedures for business process models,” *Business & Information Systems Engineering*, Vol. 4, No. 5, 2012, pp. 229–246.
- [12] J. Mendling, “Metrics for business process models,” in *Metrics for process models: empirical foundations of verification, error prediction*. Springer-Verlag, 2008, pp. 103–133.
- [13] M. Sadowska, “Quality of business models expressed in BPMN,” M.S. thesis, Wrocław University of Technology, Wrocław, 2013.
- [14] *Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models*, ISO/IEC Std. 25 010:2011(E), 2011.
- [15] *Information technology – Software product evaluation – Part 1: General overview*, ISO/IEC Std. 14 598-1:1999 (E), 1999.
- [16] S. Wagner, “Quality models,” in *Software product quality control*. Berlin: Springer, 2013, pp. 29–89.
- [17] B. Kitchenham and S. Charters, “Guidelines for performing systematic literature reviews in software engineering, v2.3,” Software Engineering Group at Keele University and Department of Computer Science at University of Durham, Tech. Rep. EBSE-2007-01, 2007.
- [18] T. Arendt and G. Taentzer, “UML model smells and model refactorings in early software development phases,” Universität Marburg, Tech. Rep. FB 12 - Mathematics and Computer Science, Nov 2010. [Online]. [http://spes2020.informatik.tu-muenchen.de/results/AT-AP4-D-AT-4\\_1\\_c.pdf](http://spes2020.informatik.tu-muenchen.de/results/AT-AP4-D-AT-4_1_c.pdf)
- [19] J. Becker, M. Rosemann, and C. Von Uthmann, “Guidelines of business process modeling,” in *Business Process Management*. Springer, 2000, pp. 30–49.
- [20] F. Fieber, M. Huhn, and B. Rumpe, “Modellqualität als indikator für softwarequalität: eine taxonomie,” *Informatik-Spektrum*, Vol. 31, No. 5, 2008, pp. 408–424.
- [21] A.A. Jalbani, J. Grabowski, H. Neukirchen, and B. Zeiss, “Towards an integrated quality assessment and improvement approach for UML models,” in *SDL’09 Proceedings of the 14th international SDL conference on Design for motes and mobiles*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 63–81.
- [22] J. Krogstie and A. Sølvberg, *Information systems engineering: Conceptual modeling in a quality perspective*. Trondheim, Norway: Kompendiumforlaget, 2003.
- [23] C. Lange and M. Chaudron, “Managing model quality in UML-based software development,” in *13th IEEE International Workshop on Software Technology and Engineering Practice*, 2005, pp. 7–16.
- [24] O. Lindland, G. Sindre, and A. Solvberg, “Understanding quality in conceptual modeling,” *IEEE Software*, Vol. 11, No. 2, March 1994, pp. 42–49.
- [25] J. Mendling, H. Reijers, and W. van der Aalst, “Seven process modeling guidelines (7PMG),” *Information and Software Technology*, Vol. 52, No. 2, 2010, pp. 127–136. [Online]. <http://www.sciencedirect.com/science/article/pii/S0950584909001268>
- [26] H.J. Nelson, G. Poels, M. Genero, and M. Piattini, “A conceptual modeling quality framework,” *Software Quality Journal*, Vol. 20, No. 1, 2012, pp. 201–228.
- [27] R. Schuette and T. Rotthowe, “The guidelines of modeling – an approach to enhance the quality in information models,” in *Conceptual Modeling – ER’98*, ser. Lecture Notes in Computer Science, T.W. Ling, S. Ram, and M. Li Lee, Eds. Springer Berlin Heidelberg, 1998, Vol. 1507, pp. 240–254.
- [28] S.S. Cherfi, J. Akoka, and I. Comyn-Wattiau, “Conceptual modeling quality – from EER to UML schemas evaluation,” in *Conceptual Modeling – ER 2002*, ser. Lecture Notes in Computer Science, S. Spaccapietra, S. March, and Y. Kambayashi, Eds. Springer Berlin Heidelberg, 2003, Vol. 2503, pp. 414–428.
- [29] D. Ssebuggwawo, S. Hoppenbrouwers, and E. Proper, “Assessing collaborative modeling quality based on modeling artifacts,” in *The Practice of Enterprise Modeling*, ser. Lecture Notes in Business Information Processing, P. van Bommel, S. Hoppenbrouwers, S. Overbeek, E. Proper, and J. Barjis, Eds. Springer Berlin Heidelberg, 2010, Vol. 68, pp. 76–90.
- [30] B. Unhelkar, “The quality strategy for UML,” in *Verification and Validation for Quality of UML 2.0 Models*. Hoboken, NY: Wiley-Interscience, 2005, pp. 1–26.

- [31] J. Cardoso, “How to measure the control-flow complexity of web processes and workflows,” in *Workflow Handbook*, 2005, pp. 199–212.
- [32] I. Vanderfeesten, H. Reijers, J. Mendling, W. van der Aalst, and J. Cardoso, “On a quest for good process models: The cross-connectivity metric,” in *Advanced Information Systems Engineering*, ser. Lecture Notes in Computer Science, Z. Bellahsene and M. Léonard, Eds. Springer Berlin Heidelberg, 2008, Vol. 5074, pp. 480–494.
- [33] V. Gruhn and R. Laue, “Adopting the cognitive complexity measure for business process models,” in *5th IEEE International Conference on Cognitive Informatics*, Vol. 1, 2006, pp. 236–241.
- [34] G. Muketha, A. Ghani, M. Selamat, and R. Atan, “A survey of business process complexity metrics,” *Information Technology Journal*, Vol. 9, No. 7, 2010, pp. 1336–1344.
- [35] E. Rolón, J. Cardoso, F. García, F. Ruiz, and M. Piattini, “Analysis and validation of control-flow complexity measures with BPMN process models,” in *Enterprise, Business-Process and Information Systems Modeling*, ser. Lecture Notes in Business Information Processing, T. Halpin, J. Krogstie, S. Nurcan, E. Proper, R. Schmidt, P. Soffer, and R. Ukor, Eds. Springer Berlin Heidelberg, 2009, Vol. 29, pp. 58–70.
- [36] V. Gruhn and R. Laue, “Complexity metrics for business process models,” in *9th International Conference on Business Information systems (BIS 2006)*, Vol. 85, 2006, pp. 1–12.
- [37] J. Cardoso, J. Mendling, and H.A. Reijers, “A discourse on complexity of process models,” in *Proceedings of the 2006 International Conference on Business Process Management Workshops (BPM’06)*, J. Eder and S. Dustdar, Eds. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 117–128.
- [38] L. Sánchez-González, F.G. Rubio, F.R. González, and M.P. Velthuis, “Measurement in business processes: a systematic review,” *Business Process Management Journal*, Vol. 16, No. 1, 2010, pp. 114–134.
- [39] J.A. Hartigan and M.A. Wong, “Algorithm AS 136: A k-means clustering algorithm,” *Journal of the Royal Statistical Society, Series C (Applied Statistics)*, Vol. 28, No. 1, 1979, pp. 100–108.
- [40] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Springer, 2012.





# Using the Cognitive Walkthrough Method in Software Process Improvement

Péter Balázs Polgár\*

*\*Eötvös Loránd University, Budapest*

sirpepe@elte.hu

## Abstract

In the past years, efforts in the field of Software Process Improvement were increasingly focusing on human aspects making one aware that people participating in the processes have a high impact on the success of any improvement. Applying the usability methodology to these problems is a promising new approach to dealing with the people issues in Software Process Improvement. This approach builds on the strengths of the usability perspective, most importantly its rich method library. One of these methods is the cognitive walkthrough method, used extensively by practitioners in software development projects.

**Keywords:** usability, software process improvement, cognitive walkthrough

## 1. Introduction

Recently, more and more Software Process Improvement (SPI) research studies the impact of people aspects on SPI projects, for example Korsaa et al. [1], Biró et al. [2, 3], Mahrin et al. [4], Kellner et al. [5], Prikladnicki [6], Siakas & Siakas [7] and Mumford [8] This impact stems from several factors. People taking ownership of the processes care more for the results and the proper execution, they are also more empowered for improvement and innovation, resulting in better processes and better products based on Messnarz et al. [9], O’Keeffe & Harington [10] and Christiansen & Johansen [11].

While SPI has an ever greater emphasis on people issues, another discipline, usability, is becoming more important as computers become ubiquitous. The usability methodology is about designing software and systems based on human needs, and as we are increasingly surrounded by computers, the ease of use of these devices becomes a major factor. Usability as a discipline has a history of helping to produce software,

and more recently systems which are suitable for users, thus resolving many people related problems other engineering fields are not suitable to handle. The usability methodology builds on a wide range of methods based on psychology and ergonomics principles helping practitioners to design systems which support users in their tasks. The user-centered design [12], forming the core value of the usability methodology, enables to view all development projects, including SPI projects with a fresh eye focusing on the humans involved in the systems.

We presented the usability approach to SPI in our previous paper [13], and while we discussed the application of some usability methods, more elaboration is needed to make this approach viable in practical work. Following this, I will describe the usage of the cognitive walkthrough method in SPI in this paper.

The remainder of this paper is structured as follows. The second section describes the people issues in SPI, the usability methodology and the usability approach to SPI. The third section introduces the cognitive walkthrough method

and discusses its use in usability projects. The fourth section presents the use of the cognitive walkthrough method in SPI. Finally section five I draw some conclusions.

## 2. The Usability Approach in Software Process Improvement

### 2.1. People Issues in Software Process Improvement

Processes are considered the cornerstone for many organizations as the most effective way of producing quality products. Organizations also realize the need to improve these processes to become more successful in their business, to be more competitive, to make products of higher quality and cheaper than their competitors. In the end processes are still carried out by people, so the effective process completion relies on the abilities, skills and motivation of individuals. While employing excellent team members certainly helps, personalities of people can still make or break a process influencing the end product. This inspires process improvement professionals to handle people issues.

The importance of people issues was realized gradually by practitioners. Korsaa et al. [1] describes how the focus got on people instead of the processes from the early days of process improvement. A study about organizational learning by O' Keeffe & Harington [10] showed evidence to support this shift of focus to people, stating that 58% of the success factors for the implementation of innovation and improvement are influenced by human and organizational aspects.

Recent models also address people issues as an important factor in improvement:

- In the ImprovAbility Model [11] by Christiansen and Johansen people aspects appear in most of the 20 parameters.
- In the Process and Enterprise Maturity Model [14] by Hammer people issues appear on most organizational and process maturity levels.
- In the team centered processes by Jacobson et al. [15] by looking at a processes from

a performer's perspective concludes that process needs to enable responses to situations.

Most recently, the SPI Manifesto [16] stated the principle: "We truly believe that SPI must involve people actively and affect their daily activities". This reinforces the focus on human aspects shifting from expert designers to the process applicators in defining and improving the processes. This principle is also supported by a number of values in the SPI Manifesto:

- "Know the culture and focus on needs": for the SPI to work, the organizational culture should be studied, as the people making up the organization carry values and practices. The SPI must consider these values to succeed.
- "Motivate all people involved": motivated people are more eager to participate in innovation and improvement, striving to look for solutions in their work.
- "Base improvement on experience and measurements": the SPI efforts must be based on the actual practices done by the organization, and all improvement activities should be based on quantifiable data.
- "Create a learning organization": the main benefit of this value is the culture supporting the continuous improvement.

Processes are represented by artifacts, namely the process descriptions. The ease of use or more specifically the usability of process descriptions was investigated by Mahrin et al. [4]. They found that there are usability related factors (for example understandable, tailorable, reusable, etc.), but their impact was not determined. Some of these factors were also proposed by Kellner et al. [5]. Other studies by Moe & Dybå [17], Scott [18] and Wang [19] showed that process descriptions have many usability problems impacting the application of the processes in a negative way.

### 2.2. Usability for Software Process Improvement

Usability is part of the software engineering quality model described in the ISO 9126 standard [20], and is often the most important attribute of a product from the user's point of

view. It also belongs to the broader field of Human-Computer Interaction studies on how humans use systems with software.

The most broadly accepted definition for usability is from the ISO 9241-11 standard [21]: “3.1 Usability: Extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.” This definition implies that we cannot produce a system that provides the same results under different contexts, with different users and different goals. Also the definitions state that with usability we are not just striving to get things done (effectiveness), but we need to do it with as little effort and resources consumed as possible (efficiency), while providing the users with a positive feeling and motivation (satisfaction).

Accurately describing the three product usage aspects (context, user and task) is an important part of usability engineering activities. Practitioners developed many methods, some of them coming from other disciplines (for example psychology, marketing and anthropology). Methods can be grouped based on the delivered data type (quantitative or qualitative), on the goal of the study (summative or formative) or on the persons involved (experts or experts and users). While not all methods produce quantitative data, most can produce easily measured values as described by Tullis [22].

For long, usability was mainly a software-engineering related discipline. With recent technological advancement and ubiquitous computing, usability is now considered in a much broader sense, also applicable to complex systems. This is reflected in the definition too, using product instead of software. The broader interpretation makes it possible to think in usability terms about complex themes like the interaction between citizens and the state (Citizen centered design, as presented by Hewitt [23]).

In discussing usability, there should be a clear distinction between the different meanings of the term. Practitioners use it to denote the quality of a software, the process of the design, and it is often hard to discern the ex-

act meaning. Keinonen has described all these meanings in [24]:

1. The development process of a product.
2. The attribute of a product
3. The use of a product
4. The user’s experiences while using a product
5. The user’s expectations about the usage of a product.

For the remainder of this paper, I will use the first and second meaning and use “usability” when referring to the product quality and “usability engineering” when referring to the development process.

Applying usability concepts in SPI has two advantages, the first one is focusing on the user and designing systems based on their needs. The resulting systems will have greater acceptance because of user involvement and will be more efficient because they more accurately capture the needs and expectations of the process performers. If this is an SPI related system, besides acceptance, the performers will have an easier time to follow it, as it was designed with the specific context in mind. Another advantage is the already established set of methods of usability engineering applicable to many kinds of tasks. While some of the methods need adaptation to be usable in an SPI environment, basic ideas stay the same.

For the usability approach to work, its concepts for the definitions have to be aligned to the SPI environment:

- Product: The system where the SPI is going to be applied, a set of processes, a process model.
- User: The performer of the process, the person doing the task.
- Context: Work conditions and situations, including the organizational and other levels of culture. Some elements of the cultural context may be strongly connected to the user (for example when having a strong national cultural background)
- Task: The process that the user performs. While the preconditions of a given process are defined by outlying elements (business goals, organizational needs, standards) the exact realization, the design of the task and

the task conditions are well within the scope of usability engineering.

- Effectiveness: The process has to come to an end with process goals successfully achieved.
- Efficiency: The process execution shall require as little resources and effort from the user as possible
- Satisfaction: The user’s experience of the executed process should be positive, empowering.

There is previous research mentioning the application of the usability methodology in the field of SPI, but these studies concentrate on the process descriptions, on the physical artifacts of the processes (for example by Mahrin et al. [4]). There is also some work concerning the usability of the tools used in SPI (for example by Al-Ani et al. [25]). While both of these fields are important, they represent just part of the scope of usability as they only deal with parts of the presentation and infrastructure layers.

Further details were presented on the usability approach in SPI by the author with Biró [13].

### 3. The Cognitive Walkthrough Method

The cognitive walkthrough method was described in detail by Nielsen and Mack [26]. It is one of the more widely used inspection methods. While it has its roots in the code-reviewing technique, the code walkthrough has been modified to identify usability issues in a product. An overview of the theory underlying the cognitive walkthrough method is provided by Rieman et al. [27].

The cognitive walkthrough is a quick and resource light method, and is usable even in the concept phase of development as it does not need a working code. The cognitive walkthrough is essentially based on the tasks of the user it tries to follow the user’s thinking while trying to learn a system through exploring the systems options.

A walkthrough is composed of six steps:

1. List the tasks the users of the system are expected to perform. If only a part of the

system is analyzed, a subset of these tasks should be chosen for evaluation.

2. Separate the tasks into intentions and goals (of the user). The intention is the overall end result the user is trying to achieve, while the goals are the result of the steps the user performs to arrive at the end result.
3. Decompose the tasks into steps. This helps to understand exactly where the system has problems.
4. The tasks and steps should be organized into evaluation sheets.
5. Perform the evaluation with chosen tasks. In each step the following questions should be asked (from [26]):
  - a) Will the user try to achieve the effect that the subtask has? Does the user understand that this subtask is needed to reach the user’s goal?
  - b) Will the user notice that the correct action is available? E.g. is the button visible?
  - c) Will the user understand that the wanted subtask can be achieved by the action? E.g. the right button is visible but the user does not understand the text and therefore will not click on it.
  - d) Does the user get feedback? Will the user know that they have done the right thing after performing the action? If one or more of these questions uncover issues, a weight should be added, and if necessary also notes describing the problem.
6. After the evaluation is complete a review should be held to decide how to act on the issues.
 

An example evaluation worksheet is shown in Table 1.

  - Step No.: The number of the current task step.
  - Task step: The name and short description of the current task step.
  - Operation: The operation the user has to perform in the current step.
  - Result: The expected result of the operation.
  - Aspect: The question that has uncovered some issues.

Table 1. Example of a cognitive walkthrough evaluation worksheet

<Task identifier>-<Task name>						
Step no.	Task step	Operation	Result	Aspect	Weight	Note
1.	...	...	...	...	...	...

- Weight: Weight given to the uncovered issues.
- Note: A short description of the found issue or anything else the evaluators found out or would like to note down.

While the cognitive walkthrough is a universal method (meaning its usage is not limited to a type of software systems), it has been adapted to specific types of software, for example Pinelle & Gutwin modified it to groupware [28], and Rowley & Rhoades presented a light weight modification the cognitive walkthrough [29]. These examples show the flexibility of the method.

Little research was made however on its applicability to processes. Novick describes a method to apply the cognitive walkthrough for operating procedures [30]. Operating procedures are similar to processes; they provide step by step instructions to follow to ensure a predefined, good outcome. As Novick states cognitive walkthrough for operating procedures provides insight into usefulness and safety beyond that associated with the cognitive walkthrough for physical interfaces. He changed the method for adaptation to procedures in five points:

1. As the steps are part of a procedure, some steps are not necessarily performed on an interface, for example when human–human interaction is concerned.
2. Procedures exist most of the time as artifacts informing the user what to do. This means that the form of these artifacts modifies the user’s understanding of the instructions.
3. At each step it has to be decided if training or experience needed for the step’s execution.
4. The correct execution of the steps should be identified not just from the user’s viewpoint but from the overall systems viewpoint too.
5. In safety critical systems (where operating procedures are often used) errors can af-

fect overall safety, so the error’s probability should be identified.

This application of the cognitive walkthrough method to operating procedures can be expanded to the SPI environment.

#### 4. Applying the Cognitive Walkthrough Method to Software Processes

Cognitive walkthroughs can be applied to SPI based on two observations:

- Novick’s work with operating procedures can be extended to the more general software processes improvement environment.
- Process can be viewed as a special type of software. Following this thought, the user interface through which the user works with the software is also the main concern of usability and the usability methods, or in this case the cognitive walkthrough. If we think of processes as software, there is an interface too, the process artifacts: descriptions, templates, tools, guidelines, standards, but also the activities and work product descriptions.

Based on these two observations we can use the cognitive walkthrough in a SPI environment.

To apply the cognitive walkthrough we first have to decide in which steps it can be used. The following generic steps of process improvement were described by Wang & King [31]:

1. Examine the needs for process improvement
2. Conduct a baseline assessment
3. Identify process improvement opportunities
4. Implement recommended improvement
5. Review process improvement achievement
6. Sustain improvement gains.

As cognitive walkthroughs are useful for evaluating design concepts, prototypes and finished products, they can be used for reviews in the second and fifth steps, their results present

issues for the third step and can evaluate improvement measures in the fourth step before executing them.

To adapt the method to SPI the changes made by Novick should be modified with process specific changes. The significant changes to the original method will be as follows:

- Using process steps instead of interface steps. Most of the time this involves the process performer interacting with a system or another human. While human–human interactions depend heavily on the individual and the organizational culture, the steps should be analyzed realistically (for example response times and schedules).
- How the process performer gets the information on the process should be evaluated too. This not only means the process descriptions should be inspected but more broadly the accessibility of these descriptions, the provided trainings etc.
- Each role involved in the process has to be evaluated separately, and also parallel to identify role interferences.
- The process achieves the results required by the overall system, the processes should be evaluated in the process environment.
- Determine if the errors found affect the process risk measures. Most projects include some kind of risk control, sometimes defined in processes. Risk should be evaluated at the process and also at the organizational level, which means that issues that may affect risk measures should be evaluated.
- Check if the step executions are aligned with the policies guiding the process.
- Key activities should be evaluated if they implement the overall goals of the process while they are executed as steps.
- Deliverables should be evaluated if they are accessible and understandable.
- Check if tools are used during the execution of the process, they should be inspected for potential issues.
- Other artifacts (guidelines, standards, templates and generally the contents of the process assets library) should be reviewed the same way as the deliverables.

With these changes to the original the cognitive walkthrough method is a viable method to apply in the SPI environment.

## 5. Conclusion

People issues in SPI are gradually recognized as an important success factor in improvement projects and the new approach of applying the usability methodology has a potential to handle these issues. This paper has introduced a practical aspect of this approach, the applications of the cognitive walkthrough usability inspection method to the SPI. I have shown how to execute the cognitive walkthrough method, and what are the significant changes needed for its application.

Cognitive walkthrough is a relatively quick and cheap method (in terms of resources, required staff and training), so its application should be viable in most organizations. While its benefits seem to be creating processes and process improvement more adaptable to the people using the processes, further research is needed on its performance under real project conditions.

## References

- [1] M. Korsaa, J. Johansen, T. Schweigert, D. Vohwinkel, R. Messnarz, R. Nevalainen, and M. Biró, “The people aspects in modern (S)PI management approaches,” 2010, presented at the EuroSPI 2010.
- [2] M. Biró, R. Messnarz, and A. Davison, “The impact of national cultural factors on the effectiveness of process improvement methods: The third dimension,” *Software Quality Professional*, Vol. 4, 2002.
- [3] M. Biró, R. Messnarz, and A. Davison, “Experiences with the impact of cultural factors on SPI,” 2001, presented at the EuroSPI 2001.
- [4] M. Mahrin, D. Carrington, and P. Strooper, “Investigating factors affecting the usability of software process descriptions,” in *Proceedings of the Software Process, 2008 International Conference on Making Globally Distributed Software Development a Success Story*, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 222–233.

- [5] M. Kellner, U. Becker, W. Riddle, J. Tomal, and M. Verlage, "Process guides: Effective guidance for process participants," in *Proceedings of the Fifth International Conference on the Software Process*, ISPA Press, Chicago, IL, USA, 1998, pp. 11–25.
- [6] R. Prikladnicki, "QUASE – A quantitative approach to analyze the human aspects of software development projects," in *Proceedings of the 2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering*, IEEE Computer Society, Washington, DC, USA, 2009, p. 78.
- [7] K.V. Siakas and E. Siakas, "The human factor deployment for improved agile quality," in *European Software Process Improvement and Innovation (EuroSPI 2006)*, 2006, pp. 11–23.
- [8] E. Mumford, "The ethics approach," *Communications of the ACM*, Vol. 36, No. 6, 1993, p. 82.
- [9] R. Messnarz, G. Spork, A. Riel, and S. Tichkiewitch, "Dynamic learning organisations supporting knowledge creation for competitive and integrated product design," in *Proceedings of the 19th CIRP Design Conference – Competitive Design*. Cranfield University Press, 2009.
- [10] T. O’Keeffe and D. Harington, "Learning to learn an examination of organisational learning in selected Irish multinationals," *Journal of European Industrial Training*, Vol. 25, No. 2/3/4, 2001, pp. 137–147.
- [11] M. Chirstiansen and J. Johansen, "ImprovAbility™ guidelines for low maturity organisations," *Software Process: Improvement and Practice*, Vol. 13, No. 4, 2008, pp. 319–325, presented at the EuroSPI 2007.
- [12] *Human-centred design processes for interactive systems*, ISO Std. 13 407:1999, 1999.
- [13] P. Balázs Polgár and M. Biró, "The usability approach in software process improvement," in *Systems, Software and Service Process Improvement*, ser. Communications in Computer and Information Science, R. O’Connor, J. Pries-Heje, and R. Messnarz, Eds. Springer Berlin Heidelberg, 2011, Vol. 172, pp. 133–142. [Online]. [http://dx.doi.org/10.1007/978-3-642-22206-1\\_12](http://dx.doi.org/10.1007/978-3-642-22206-1_12)
- [14] M. Hammer. The process and enterprise maturity model. Retrieved on 10.05.2011. [Online]. <http://www.hammerandco.com/HammerAndCompany.aspx?id=58>
- [15] I. Jacobson and I. Spence, "Enough of processes – lets do practices," *Journal of Object Technology*, No. 6, 2007, pp. 41–66.
- [16] J. Pries-Heje, J. Johansen, and R. Messnarz. SPI manifesto. Retrieved on 10.05.2011. (2010). [Online]. [http://www.iscn.com/Images/SPI\\_Manifesto\\_A.1.2.2010.pdf](http://www.iscn.com/Images/SPI_Manifesto_A.1.2.2010.pdf)
- [17] N.B. Moe and T. Dybå, "The use of an electronic process guide in a medium-sized software development company," *Software Process: Improvement and Practice*, Vol. 11, No. 1, 2006, pp. 21–34. [Online]. <http://dx.doi.org/10.1002/spip.250>
- [18] L. Scott, L. Carvalho, R. Jeffery, J. D’Ambra, and U. Becker-Kornstaedt, "Understanding the use of an electronic process guide," *Information and Software Technology*, Vol. 44, No. 10, 2002, pp. 601–616. [Online]. <http://www.sciencedirect.com/science/article/pii/S0950584902000800>
- [19] Y. Wang, *Software engineering processes: principles and applications*. Boca Raton, Fla: CRC Press, 2000.
- [20] *Software engineering – Product quality*, ISO/IEC Std. 9126:2001, 2001.
- [21] *Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability*, ISO Std. 9241-11:1998, 1998.
- [22] T. Tullis and B. Albert, *Measuring the user experience: collecting, analyzing, and presenting usability metrics*. Amsterdam, Boston: Elsevier/Morgan Kaufmann, 2008.
- [23] J.F. Hewitt, "Citizen-centered design (slowly) revolutionizes the media and experience of US elections," *interactions*, Vol. 16, No. 5, 2009, pp. 18–25.
- [24] T. Keinonen, "One-dimensional usability – influence of usability on consumers’ product preference," Master’s thesis, University of Art and Design Helsinki UIAH, 1998.
- [25] B. Al-Ani, E. Trainer, R. Ripley, A. Sarma, A. van der Hoek, and D. Redmiles, "Continuous coordination within the context of cooperative and human aspects of software engineering," 2008, pp. 1–4.
- [26] J. Nielsen and R. Mack, *Usability Inspection Methods*. Wiley, 1994.
- [27] J. Rieman, M. Franzke, and D. Redmiles, "Usability evaluation with the cognitive walkthrough," in *Conference companion on Human factors in computing systems*, 1995, pp. 387–388.
- [28] D. Pinelle and C. Gutwin, "Groupware walkthrough: Adding context to groupware usability evaluation," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI ’02. New York, NY,

- USA: ACM, 2002, pp. 455–462. [Online]. <http://doi.acm.org/10.1145/503376.503458>
- [29] D. Rowley and D. Rhoades, “The cognitive jogthrough: a fast-paced user interface evaluation procedure,” in *Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA: ACM, 1992, pp. 389–395.
- [30] D. Novick, “Using the cognitive walkthrough for operating procedures,” *interactions*, No. 6, 1999, pp. 31–37.
- [31] Y. Wang and G. King, “Philosophies and approaches to software process improvement,” in *Conference on Software Process Improvement (EuroSPI’99)*, Pori, Finland, 1999, pp. 7.24–7.38.



# Construction of Variable Strength Covering Array for Combinatorial Testing Using a Greedy Approach to Genetic Algorithm

Priti Bansal\*, Sangeeta Sabharwal\*, Nitish Mittal\*, Sarthak Arora\*\*

\**Netaji Subhas Institute of Technology, University of Delhi*

\*\**School of Computer Science and Engineering, Vellore Institute of Technology, Tamil Nadu*

bansalpriti79@gmail.com, ssab63@gmail.com, nitishmittal94@gmail.com,  
sarthak10193@gmail.com

## Abstract

The limitation of time and budget usually prohibits exhaustive testing of interactions between components in a component based software system. Combinatorial testing is a software testing technique that can be used to detect faults in a component based software system caused by the interactions of components in an effective and efficient way. Most of the research in the field of combinatorial testing till now has focused on the construction of optimal covering array (CA) of fixed strength  $t$  which covers all  $t$ -way interactions among components. The size of CA increases with the increase in strength of testing  $t$ , which further increases the cost of testing. However, not all components require higher strength interaction testing. Hence, in a system with  $k$  components a technique is required to construct CA of fixed strength  $t$  which covers all  $t$ -way interactions among  $k$  components and all  $t_i$ -way (where  $t_i > t$ ) interactions between a subset of  $k$  components. This is achieved using the variable strength covering array (VSCA). In this paper we propose a greedy based genetic algorithm (GA) to generate optimal VSCA. Experiments are conducted on several benchmark configurations to evaluate the effectiveness of the proposed approach.

**Keywords:** combinatorial testing, variable strength covering array, genetic algorithm, greedy approach

## 1. Introduction

The increasing dependence on software systems in every field, such as medicine, agriculture, communication systems has increased the need to perform software testing in an effective and efficient manner so as to ensure the delivery of reliable and quality software. In the case of a component based software system, interactions among components are often complex and they may cause interaction errors. It is therefore important to check all the possible interactions among various components to uncover faults caused by their interactions. As each component may have multiple configurations, testing all possible combinations

of components is practically impossible due to time and cost constraints. Furthermore, the number of test cases increases exponentially with the increase in number of components. A sampling strategy is therefore required to select a subset of configurations to be tested from the large interaction space. Combinatorial testing is a testing technique that samples the set of configurations in such a way that it covers all  $t$ -way ( $t$  denotes the strength of testing) interactions of components [1].

Covering arrays (CAs) and mixed covering arrays (MCAs) are combinatorial structures that have enjoyed a wide range of application in the field of software and hardware testing [2]. Due to

the importance of CAs, significant research has been carried out to construct CAs of optimal size by the researchers in the past. A CA constructed to perform  $t$ -way (2-way, 3-way, etc.) testing checks only all  $t$ -way interactions of components. Empirical studies show that a test set covering all possible 2-way combinations of input parameter values is effective for software systems [1, 3–5]. Dalal et al. [6] showed that testing all pair-wise interactions in a software system finds a large percentage of the existing faults. Kuhn et al. [7] examined fault reports for many software systems and concluded that more than 70% of the faults are triggered by a 2-way interaction of the input parameters. Faults can also be caused by the interaction of more than two parameters. In order to uncover faults caused by the interaction of more than two components, it is required to test higher strength interactions of components. Empirical studies in Kuhn et al. [7] and Kuhn and Reilly [8] show that most of the faults are triggered by a relatively low degree of interactions and suggest the need to perform testing up to  $t = 6$ .

Consider a Graphical User Interface (GUI) based on a windowing system which has five components, each with three possible values as shown in Table 1. For exhaustively testing the components' interactions in this system, 243 test cases are required whereas only 11 test cases for 2-way testing and 37 test cases for 3-way testing are required respectively. Evidently, the increase in strength of testing leads to the increase in number of test cases. However, it is quite often the case that certain components have stronger interactions while others may have few or none [9]. Hence, it is not desirable to perform higher strength interaction testing of all the components. A better way to test the system is to identify the subsets of components which are involved in stronger interactions and apply higher strength interaction testing only on these subsets to uncover the faults caused by their interactions. This is achieved using the variable strength covering array (VSCA), which is a CA or MCA of fixed strength  $t$  and also contains a set of disjoint CAs or MCAs of strength greater than  $t$ . As mentioned above, the example shown in Table 1 requires 11 test cases for 2-way testing.

Assume, first four components have stronger interactions compared to the fifth component. So it is feasible to perform 3-way testing only on the first four components, which additionally requires 16 test cases as illustrated in Figure 1 and Figure 2. Consequently, a total of 27 test cases are required for variable strength testing against 37 test cases required for a complete 3-way testing. We can see that VSCA achieves higher strength interaction coverage with the reduced number of test cases. So it is advantageous to find an effective technique to construct optimal VSCA to perform testing of a component based system efficiently.

Kernel	DS	WM	DSCP	GI
FreeBSD	Weston	Awesome	Wayland	KDE Plasma
FreeBSD	X.Org	Compiz	X11	Aqua
XNU	X.Org	OpenBox	Wayland	KDE Plasma
XNU	KWin	Awesome	X11	KDE Plasma
XNU	Weston	Compiz	X11	Gnome Shell
Linux	KWin	Compiz	Wayland	Aqua
XNU	Weston	Awesome	Quartz	Aqua
Linux	X.Org	Compiz	Wayland	KDE Plasma
Linux	X.Org	Awesome	Wayland	Gnome Shell
FreeBSD	KWin	OpenBox	Quartz	Gnome Shell
Linux	Weston	OpenBox	X11	Aqua

Figure 1. CA (11, 2, 3<sup>5</sup>)

Kernel	DS	WM	DSCP	GI
FreeBsd	X.Org	Compiz	Quartz	Aqua
XNU	Kwin	Awesome	Quartz	KDE Plasma
FreeBsd	Weston	Compiz	Wayland	KDE Plasma
Linux	Weston	Compiz	X11	Gnome Shell
Linux	Kwin	OpenBox	Wayland	KDE Plasma
FreeBsd	X.Org	Awesome	X11	Aqua
FreeBsd	Kwin	Compiz	X11	KDE Plasma
XNU	X.Org	OpenBox	Quartz	Gnome Shell
FreeBsd	Weston	Awesome	Quartz	Gnome Shell
FreeBsd	Kwin	Awesome	Wayland	Gnome Shell
XNU	Weston	Awesome	X11	Gnome Shell
FreeBsd	X.Org	OpenBox	Wayland	Gnome Shell
Linux	X.Org	OpenBox	X11	KDE Plasma
XNU	Weston	OpenBox	Wayland	Aqua
XNU	Weston	Compiz	Quartz	Gnome Shell
Linux	Kwin	Awesome	X11	Aqua
Linux	Weston	Awesome	Wayland	KDE Plasma
XNU	X.Org	Awesome	Wayland	Gnome Shell
Linux	X.Org	Compiz	Wayland	Gnome Shell
XNU	X.Org	Compiz	X11	Gnome Shell
XNU	Kwin	OpenBox	X11	Aqua
Linux	Weston	OpenBox	Quartz	Aqua
XNU	Kwin	Compiz	Wayland	Gnome Shell
Linux	X.Org	Awesome	Quartz	Gnome Shell
FreeBsd	Weston	OpenBox	X11	Gnome Shell
FreeBsd	Kwin	OpenBox	Quartz	Aqua
Linux	Kwin	Compiz	Quartz	KDE Plasma

Figure 2. VSCA (27; 2, 3<sup>5</sup>, (3, 3<sup>4</sup>))

Table 1. GUI based on a windowing system having five components, each with three values

Kernel	Display Server (DS)	Window Manager (WM)	Display Server Communication Protocol (DSCP)	Graphical Interface (GI)
Linux	Weston	Awesome	X11	KDE Plasma
FreeBSD	KWin	Compiz	Wayland	Aqua
XNU	X.Org	OpenBox	Quartz	Gnome Shell

The problem of constructing an optimal VSCA is NP-complete [10, 11]. Although many algebraic and computational construction methods have been proposed by the researchers to construct optimal CA/MCA, fewer strategies (greedy and meta-heuristic) exist to construct optimal VSCA. The amount of work that has been done to construct VSCA using meta-heuristic techniques such as Simulated Annealing (SA), Particle Swarm Optimization (PSO), Harmony Search (HS) and their impressive results has motivated us to explore GA to construct optimal VSCA.

To exploit the strength of both greedy and meta-heuristic techniques we present a technique that augments GA with a greedy technique to construct optimal VSCA efficiently. Experiments are conducted to evaluate the performance of the proposed technique with the existing techniques.

However, the problem that exists with the construction of VSCA is the existence of constraints or dependencies between components values in terms of restrictions or compulsion on components values that can coexist. For instance, in the example shown in Table 1, Quartz is a Mac technology and therefore cannot be run on Linux or FreeBSD. This constraint must be taken into account when generating test cases so that Quartz and Linux/FreeBSD do not appear in the same test case. Similarly, KDE Plasma and XNU cannot appear in the same test case as XNU does not support KDE Plasma. If constraints and dependencies are considered, then combinatorial testing becomes constrained combinatorial testing. In this paper, we focus on combinatorial testing and leave constrained combinatorial testing for future work.

The remainder of this paper is organized as follows. Section 2 gives the necessary back-

ground on combinatorial objects. Section 3 gives an overview of GA. Section 4 presents various methods available to construct VSCA. Section 5 describes the proposed strategy to generate VSCA for  $t$ -way testing. Section 6 describes the implementation and presents result of experiments performed to compare the effectiveness of the proposed approach with other existing approaches. Section 7 presents threats to validity. Section 8 concludes the paper and future plans are outlined.

## 2. Background

This section discusses the necessary background related to combinatorial objects.

### 2.1. Orthogonal Array

An orthogonal array  $OA_\lambda(N; t, k, v)$  is an  $N \times k$  array on  $v$  symbols such that every  $N \times t$  sub-array contains all ordered subsets of size  $t$  from  $v$  symbols exactly  $\lambda$  times and they have the property  $\lambda = N/v^t$  [12]. The use of OA in the field of software testing is limited due to the restrictions imposed on OA that all parameters have same number of values and that each pair of values can be covered the same number of times [13]. In general, OA is difficult to generate and its test suite is often quite large with  $\lambda > 1$ . However, OA has its advantages, such as making it relatively easy to identify the particular combination that caused a failure [11]. If an OA with  $\lambda = 1$  exists for some value of  $k$  and  $v$ , then it is an optimal array. To complement OA construction and to overcome its restrictions, CA and MCA have been introduced.

## 2.2. Covering Array

A covering array [12] denoted by  $CA_\lambda(N; t, k, v)$ , is an  $N \times k$  two dimensional array on  $v$  symbols such that every  $N \times t$  sub-array contains all ordered subsets from  $v$  symbols of size  $t$  at least  $\lambda$  times. If  $\lambda = 1$ , it means that every  $t$ -tuple needs to be covered only once and we can use the notation  $CA(N; t, k, v)$ . Here,  $k$  represents the number of values of each parameter and  $t$  is the strength of testing. An optimal CA contains a minimum number of rows to satisfy the properties of the entire CA. The minimum number of rows is known as covering array number and is denoted by  $CAN(t, k, v)$ . A CA of size  $N \times k$  represents a test set where each row corresponds to a test case, each column represents a component and the values in the column represent the domain of the respective component.

## 2.3. Mixed Covering Array

A mixed covering array [14], denoted by  $MCA(N; t, k, (v_1, v_2, \dots, v_k))$ , is an  $N \times k$  two dimensional array, where  $v_1, v_2, \dots, v_k$  is a cardinality vector which indicates the values for every column. An MCA has the following two properties: i) Each column  $i$  ( $1 \leq i \leq k$ ) contains only elements from a set  $S_i$  with  $|S_i| = v_i$  and ii) The rows of each  $N \times t$  sub-array cover all  $t$ -tuples of values from the  $t$  columns at least once. The minimum  $N$  for which there exists an MCA is called a mixed covering array number and is denoted by  $MCAN(t, k, (v_1, v_2, \dots, v_k))$ . A shorthand notation can be used to represent MCAs by combining equal entries in  $v_i : 1 \leq i \leq k$ . An  $MCA(N; t, k, (v_1, v_2, \dots, v_k))$  can be represented as  $MCA(N; t, k, (w_1^{q_1}, w_2^{q_2}, \dots, w_s^{q_s}))$ , where  $k = \sum_{i=1}^s q_i$  and  $w_j | 1 \leq j \leq s \subseteq \{v_1, v_2, \dots, v_k\}$ . Each element  $w_j^{q_i}$  in the set  $\{w_1^{q_1}, w_2^{q_2}, \dots, w_s^{q_s}\}$  means that  $q_i$  parameters can take  $w_j$  values each. A MCA of size  $N \times k$  represents a test set with  $N$ -test cases for a system with  $k$  components, each with varying domain size.

## 2.4. Variable Strength Covering Array

A variable strength covering array [9], denoted by  $VSCA(N; t, k, (v_1, v_2, \dots, v_k), C)$ , is

an  $N \times k$  CA or MCA of strength- $t$  containing  $C$  where,  $C$  is a set of disjoint CAs or MCAs each of strength greater than  $t$ . Each element of  $C$  is a subset of VSCA and they can have variable strength of testing. For example, a  $VSCA(N; 2, 4^3 5^3 6^2, \{CA(3, 4^3), MCA(4, 5^3 6^1)\})$  is shown in Fig. 3(a). Here, the overall array is an MCA having three components with four values, three with five values and two with six values each (the values of each component are labelled  $0, 1, 2, 3, \dots$ ). It covers all 2-way interactions among components. In addition to this, it contains two sub arrays: a CA of strength-3 that covers all 3-way interactions among first three components with four values and an MCA of strength-4 covering all 4-way interactions among three components with five values and one component with six values. The order of columns in the sub arrays in  $C$  is important and they are listed consecutively from left to right. If a VSCA contains  $n$  identical sub arrays with the same  $t, k$  and  $v$ , they can be represented as  $CA(t, v^k)^n$ . For instance,  $VSCA(N, 2, 3^{11}, (3, 3^4)^2)$  shown in Figure 3(b) represent a CA that covers all 2-way interactions among eleven components with three values and contains two disjoint sub arrays, each of which covers all 3-way interactions among four components with three values each.

## 3. Genetic Algorithm

The basics of GA were first proposed by Holland [15]. GA is a meta-heuristic search based optimization technique originating from the Darwinian theory of evolution by natural selection where fitter individuals are more likely to survive in a competing environment [16]. It is a global search technique characterized by evolution in every generation, starting with a randomly generated initial population. The initial population represents potential solutions to the given problem. Each individual in the population is associated with a fitness value that is calculated using a fitness function. The fitness function is a function of the objective that we want to optimize (maximize or minimize). The fitness value of an individual appraises us of the merit of a solution

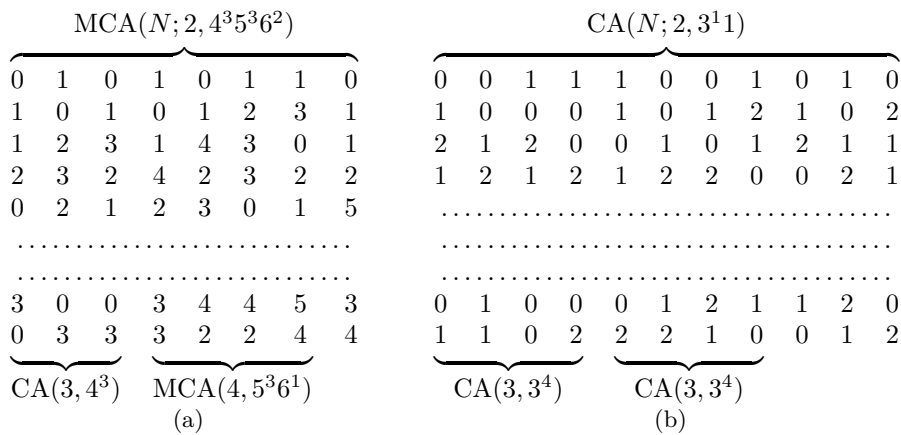


Figure 3. Representation of VSCA

for the given problem. In each generation, the population evolves towards better solutions by means of evolutionary operators such as selection, crossover and mutation. This process continues until a satisfactory solution is found or the maximum number of generations is reached. As compared to other meta-heuristic techniques, GA starts with a population of solutions instead of a single solution that helps GA to cover the solution search space more thoroughly and avoid its chances of getting stuck in the local minima. Moreover, GA is easy to understand and can be applied to an optimization problem which can be described with chromosome encoding. On the contrary, the complexity of crossover and mutation operations is attributed to longer run time and, furthermore, GA cannot assure a constant optimization response time which limits its use in real time applications. The basic outline of GA is shown in Fig. 4.

Having described the notations, in the next section we will briefly discuss the existing state-of-the-art algorithms for constructing optimal VSCA for pair-wise testing.

#### 4. Related Work

In an extensive survey performed by Khalsa and Labiche in [17], it has been found that 75 tools/algorithms exist to generate CA/MCA for combinatorial testing but not all of them sup-

port construction of VSCA. The strategies that support the construction of VSCA are broadly classified into computational strategies and artificial intelligence based strategies. Computational strategies use greedy approach to construct VSCA by using either one-test-at-a-time or one-parameter-at-a-time approach. The strategies based on one-test-at-a-time approach use a greedy heuristic and try to select a test case that covers the maximum number of uncovered interactions from a pool of candidate test cases. However, selecting such a test case itself is an NP-complete problem [18]. Some of the strategies that use one-test-at-a-time approach are the Test Vector Generator (TVG) [19], Pairwise Independent Combinatorial Testing (PICT) [20], Intelligent Test Case Handler<sup>1</sup> (ITCH), Density [21], DA-RO and DA-FO [22], and TSG [23]. The strategies that use one-parameter-at-a-time approach are ACTS [24, 25] and ParaOrder [21].

Recently the search based software testing (SBST) is increasingly gaining importance and is been used to solve a wide range of problems in software testing. Meta-heuristic techniques are being used by the SBST community to find an optimal solution for software testing problems. The problem of generating an optimal CA/MCA/VSCA is also considered as a SBST problem [26, 27]. Meta-heuristic techniques start by searching over a large set of feasible solutions and can often find better solutions with

<sup>1</sup> IBM Intelligent Test Case Handler

```

generate an initial population P randomly
evaluate fitness of each individual in P using the fitness function
while ((generation ≤ maximum generation) and solution not found)
    select a subset of individuals P' from current generation for offspring production
    apply crossover to P'
    apply mutation to P'
    replace low fitness individuals in P by offspring in P'
    evaluate P
end while
return individual with highest fitness

```

Figure 4. Outline of basic GA

fewer computational efforts as compared to other algorithms, iterative methods or simple heuristics [28]. To the best of our knowledge, only five meta-heuristic based strategies to generate VSCA exist in the literature. Table 2 list features of all tools/algorithms that use meta-heuristic techniques and some selected tools that use greedy techniques to construct CA/MCA/VSCA.

## 5. The Proposed Approach for Construction of VSCA

In this section, we present our proposed strategy of VSCA-GA that aims to generate an optimal VSCA to cover all (100%)  $t$ -way and  $t_i$ -way interactions between components in a component based system. Here,  $t$  denotes the overall strength of VSCA and  $t_i$  denotes the strength of sub arrays. VSCA-GA uses a greedy based approach to GA to generate optimal VSCA. Let  $VSCA(N; t, k, (v_1, v_2, \dots, v_k), C)$  represent a VSCA configuration. VSCA-GA starts by creating an initial population of  $P_{\text{size}}$  individuals where each individual chromosome represents a candidate solution which is a VSCA of size  $N \times k$ . Here,  $N$  the number of rows of VSCA corresponds to test cases and  $k$  represents the number of components in a component based system. At the start of the search process  $N$  is unknown, so we use the method suggested by Stardom [39], where we start with a large random array and apply binary search repeatedly until a solution is found. In case the size of  $N$  is known in advance, i.e. best bound achieved in the existing state-of-the-art, we can start with the known

size and try to optimize it further. An individual chromosome in the population is represented by  $VSCA_f | 1 \leq f \leq P_{\text{size}}$  and each  $VSCA_f$  in the population has a fitness associated with it which is defined as the total number of distinct variable strength interactions covered by it. It is calculated as

$$\text{Fitness}(VSCA_f) = \sum_{i=t, t_1, \dots, t_n} \text{number of distinct } i\text{-way interactions covered by } VSCA_f \quad (1)$$

Where,  $t$  is the overall strength of VSCA,  $t_1, t_2, \dots, t_n$  are the strength of sub-arrays.

After initialization, GA searches the solution space by applying genetic operators such as selection, crossover and mutation repeatedly to find the best solution. The process continues until a solution is found or the maximum number of iterations is reached. In case VSCA-GA starts by taking  $N$  from the existing literature and a solution is found at this  $N$ , then the size of VSCA is decreased by one, otherwise the size of VSCA is increased by one and VSCA-GA is executed again in both cases. When VSCA-GA is re-executed, we seed the initial population by supplying the best VSCA generated in the previous run. If the size of VSCA in the current run is less than that in the previous run, we decrease the size of seeded VSCA by one by removing the test case that contributes least to the fitness of VSCA whereas, if the size of VSCA in the current run is greater than the size in the previous run, then we add a randomly generated test case to the existing VSCA. The various steps of VSCA-GA strategy are explained below.

Table 2. Comparison of various tools/algorithms for constructing CA/MCA/VSCA for CIT

S. No.	Tool / Algorithm	Variable Strength	Maximum Strength Support(t)	Technique Employed	Test Generation Strategy	Constraint Handling
1.	AETG [1]	✗	2			✓
2.	ITCH <sup>1</sup>	✓	6			✓
3.	TVG [19]	✓	6			✓
4.	PICT [20]	✓	6			✓
5.	Density [21]	✓	3		One Test at a Time	✗
6.	DA-RO [22]	✓	3	Greedy		✗
7.	DA-FO [22]	✓	3			✗
8.	TSG [23]	✓	3			✗
9.	Jenny [29]	✗	8			✓
10.	ACTS (IPOG) [24, 25]	✓	6		One Parameter at a Time	✓
11.	ParaOrder [21]	✓	3			✗
12.	SA [9]	✓	3	Simulated Annealing		✗
13.	GA [30]	✗	3	Genetic Algorithm		✗
14.	ACA [30]	✗	3	Ant Colony Optimization		✗
15.	ACS [31]	✓	3	Ant Colony Optimization		✗
16.	TSA [32]	✗	6	Tabu Search		✗
17.	GAPTS [33]	✗	2	Genetic Algorithm	One Test at a Time	✗
18.	PWiseGen [34]	✗	2	Meta-heuristic Genetic Algorithm		✗
19.	VS-PSTG [35]	✓	6	Particle Swarm Optimization		✗
20.	HSS [36]	✓	15	Harmony Search		✓
21.	HSTCG [37]	✓	7	Harmony Search		✓
22.	CASA [27]	✗	3	Simulated Annealing		✓
23.	PSO [38]	✗	2	Particle Swarm Optimization	One Parameter at a Time	✗
24.	PSO [38]	✗	2	Particle Swarm Optimization		✗

### 5.1. A Greedy Approach to Generate Initial Population

When GA is used to construct VSCA, the role of initial population on the performance of GA cannot be ignored as it can affect the convergence speed and quality of the final solution [40, 41]. Generally, initial population is generated randomly. However, recognizing the effect of initial population on GA performance, several population initialization methods for GA have been proposed in the past by the researchers [41–46]. Here, we present a greedy approach for generating a good quality initial population of VSCA, which is achieved by focusing on the coverage of maximum number of possible uncovered interactions.

Let us consider the system under test consisting of  $k$  components where a component is represented by  $C_m | 1 \leq m \leq k$  and each component  $C_m$  can take values from 0 to  $(v_m - 1)$  ( $v_m$

is the number of possible values of component  $C_m$ ). The  $j^{\text{th}} | 1 \leq j \leq v_m$  value of component  $C_m$  is represented by  $\text{val}_{mj}$ . To generate an initial population of VSCAs, VSCA-GA starts by computing and storing all the possible  $t$ -way and  $t_i$ -way interactions between the values of all the components in an interaction list  $L$ , based on the configuration of VSCA. Then, it calculates the number of uncovered interactions of each value  $\text{val}_{mj}$  of every component  $C_m$ , stores it in a variable  $N_{\text{uncovered}}(\text{val}_{mj})$  and assigns a probability of selection denoted by  $P(\text{val}_{mj})$  to each of them. The probability of selection assigned to a value  $\text{val}_{mj}$  of component  $C_m$  represents its chances of getting selected when a test case is created. In our case, the probability assigned to a value  $v_{mj}$  of component  $C_m$  depends upon the number of uncovered interactions of  $\text{val}_{mj}$  as well as the total number of uncovered interactions of component  $C_m$ . Initially all values  $\text{val}_{mj}$  of a component  $C_m$  are involved in an equal number of uncovered

interactions, therefore each of them will have an equal probability of getting selected. For instance, if a component has four possible values then initially each one of them will have the probability of selection equal to 0.25. VSCA-GA generates the first test case  $tc_{f1}$  in  $VSCA_f | 1 \leq f \leq P_{size}$  by selecting a value of each component randomly as each one of them have an equal probability of selection. After the generation of first test case, VSCA-GA updates the interaction list  $L$  by eliminating interactions that are covered in  $tc_{f1}$ . Let the value  $val_{ms}$  of component  $C_m$  be selected in  $tc_{f1}$  and the number of interactions covered by  $val_{ms}$  in  $tc_{f1}$  is  $N_{covered}(val_{ms})$ , then the number of interactions of  $val_{ms}$  left uncovered is denoted by  $N'_{uncovered}(val_{ms})$  and is calculated as:

$$N'_{uncovered}(val_{ms}) = N_{uncovered}(val_{ms}) - N_{covered}(val_{ms}) \quad (2)$$

Let  $P_{old}(val_{ms})$  denote the probability of selection of value  $val_{ms}$  before selection, then after selection the probability of  $val_{ms}$  becomes:

$$P_{new}(val_{ms}) = P_{old}(val_{ms}) \times \frac{N'_{uncovered}(val_{ms})}{N_{uncovered}(val_{ms})} \quad (3)$$

The decrease in the probability of value  $val_{ms}$  is calculated using Equation 4 and is distributed among the remaining values  $val_{mj} | 1 \leq j \leq v_m$  and  $j \neq s$  of component  $C_m$  according to Equation 5.

$$P_{decrement}(val_{ms}) = P_{old}(val_{ms}) \times \left( 1 - \frac{N'_{uncovered}(val_{ms})}{N_{uncovered}(val_{ms})} \right) \quad (4)$$

$$P_{new}(val_{mj}) = P_{old}(val_{mj}) + \left( \frac{N_{uncovered}(val_{mj})}{\sum_{j=1}^{j \neq s} \text{to } v_m} N_{uncovered}(val_{mj})} \times P_{decrement}(val_{ms}) \right) \quad (5)$$

Equation 5 increases the probability of the value  $val_{mj}$  of component  $C_m$  based on the number of its uncovered interactions and the total number of uncovered interactions of the remaining values (except  $val_{ms}$ ) of component  $C_m$ . Hence,

the higher the number of remaining uncovered interactions of a value, the higher will be the increase in its probability and vice versa, thereby getting greedy by extending a higher opportunity of selection to the values with maximum uncovered interactions. Once the probability of each value of every component is updated, the number of uncovered interactions of the selected value  $val_{mj} \forall m$  is updated by assigning the value of  $N'_{uncovered}(val_{ms})$  to  $N_{uncovered}(val_{ms})$ . The succeeding test cases  $tc_{fi} | 2 \leq i \leq N$  are generated by selecting a value for each component based on the probabilities that are updated after the generation of every test case. Since each value  $val_{mj}$  of a component  $C_m$  may have different probability of selection, to select a value of a component, a random number is generated in the range  $[0, 1]$  and based on the interval in which the random number falls; the value  $val_{mj}$  of the component  $C_m$  is selected. For instance, consider a component  $C_m$  having four possible values and assume that at some point of time during the test case generation process, the probability of selection of each of the four values  $val_{m1}$ ,  $val_{m2}$ ,  $val_{m3}$  and  $val_{m4}$  becomes 0.20, 0.35, 0.35 and 0.10 respectively. If the generated random number lies in the range  $[0, 0.2]$  then value  $val_{m1}$  is selected, if it lies in the range  $(0.2, 0.55]$  then value  $val_{m2}$  is selected, if it lies in the range  $(0.55, 0.9]$  then value  $val_{m3}$  is selected otherwise  $val_{m4}$  is selected. An example to illustrate the greedy approach to generate initial population is given below.

Example: Let us consider a component based a system having configuration  $(N; 2, 2^3 3^1, CA(3, 2^3))$  as shown below:

$C_1$	$C_2$	$C_3$	$C_4$
a1	a2	a3	a4
b1	b2	b3	b4
			c4

To construct a VSCA in the initial population, VSCA-GA assigns a probability of selection to each value of every component. Initially, each value of a component has an equal number of uncovered interactions; therefore each of them will have an equal probability of selection. The



probability of selection of each value of every component is shown below:

$C_1$	$C_2$	$C_3$	$C_4$
$P(a1)=0.5$	$P(a2)=0.5$	$P(a3)=0.5$	$P(a4)=0.333$
$P(b1)=0.5$	$P(b2)=0.5$	$P(b3)=0.5$	$P(b4)=0.333$
			$P(c4)=0.333$

The first test case  $TC_1$  is constructed by selecting a value for each component randomly from their respective input domain. Let  $TC_1$  be: a1, b2, a3, b4.

Now, VSCA-GA changes the probability of selection of each value of every component based on the number of their uncovered interactions using Equations 2–5. The new probabilities become:

$C_1$	$C_2$	$C_3$	$C_4$
$P(a1)=0.32$	$P(a2)=0.68$	$P(a3)=0.32$	$P(a4)=0.166$
$P(b1)=0.68$	$P(b2)=0.32$	$P(b3)=0.68$	$P(b4)=0.417$
			$P(c4)=0.417$

Subsequently, for generating the next test case  $TC_2$ , VSCA-GA generates random numbers. Let the random numbers generated be 0.2, 0.5, 0.2 and 0.3 for each component respectively. Therefore,  $TC_2$  will be: a1, a2, a3, b4.

Now, VSCA-GA again changes the probability of selection of each value of every component based on the number of their uncovered interactions using Equations 2–5. The new probabilities become:

$C_1$	$C_2$	$C_3$	$C_4$
$P(a1)=0.18$	$P(a2)=0.43$	$P(a3)=0.18$	$P(a4)=0.235$
$P(b1)=0.82$	$P(b2)=0.57$	$P(b3)=0.82$	$P(b4)=0.209$
			$P(c4)=0.556$

The same procedure is repeated to construct the remaining  $(N - 2)$ -test cases. Once a VSCA is generated, the same procedure is repeated to generate all the remaining VSCAs in the initial population. Notably, every time a new VSCA is generated, the interaction list  $L$  is reinitialized to store all the possible  $t$ -way and  $t_i$ -way interactions between the values of all the components based on the configuration of VSCA.

### 5.2. A Greedy Approach to Perform Crossover

The next step after initialization is the application of selection, crossover and mutation operators repeatedly to generate optimal VSCA that covers all possible  $t$ -way and  $t_i$ -way interactions. The crossover operator combines the genes of two or more parents to generate an offspring. It is based on the idea that the exchange of information between good chromosomes will generate even better offspring [47]. There are many variations of the crossover method, namely single-point crossover, two-point crossover, multi-point crossover, uniform crossover, etc. The number of crossover points determines how many segments are exchanged between the parents. The length (number of genes) of each segment may vary and it depends on the position of crossover points. VSCA-GA performs a crossover at the boundaries of test cases and the length of a segment is always equal to one (i.e. one test case). When a crossover is performed, it is quite possible that during the exchange of information between parents, some good features of a parent may get lost. In our case, based on the configuration of VSCA each test case  $tc_{f_i} | 1 \leq i \leq N$  in  $VSCA_f$  covers some fixed number of  $t$ -way and  $t_i$ -way interactions, out of which some interactions are distinctly covered by  $tc_{f_i}$  only. When a random crossover is performed, it may happen that during the exchange of information between two parent VSCAs say  $VSCA_1$  and  $VSCA_2$ , the best test case  $tc_{1_i}$  covering maximum number of distinct interactions in  $VSCA_1$  may get exchanged with the test case  $tc_{2_i}$  of  $VSCA_2$ . This may result in the gain of new interactions as well as loss of existing distinct interactions covered by  $tc_{1_i}$  in  $VSCA_1$  thereby, reducing the net gain in fitness after the crossover. The net gain in fitness is calculated using Equation 6.

$$\begin{aligned} \text{Net gain in fitness}(VSCA_f) = & \\ & \text{Number of new interactions gained} - \\ & \text{Number of existing distinct interactions lost} \end{aligned} \quad (6)$$

In order to minimize the loss of existing distinct interactions and to maximize the net gain in fitness during a crossover, VSCA-GA uses a greedy approach to perform a crossover. It takes the number of test cases which are to be exchanged during the crossover as input (NTC) instead of the number of crossover points, which helps it in selecting the test cases greedily for crossover. VSCA-GA starts by selecting VSCAs using roulette wheel selection to become parents during the crossover. In the roulette wheel selection, a probability is being assigned to each individual in the population. This probability is calculated on the basis of the fitness of the individual and thus the individuals with higher fitness have better chances of getting selected for reproduction. Out of the two parents selected using roulette wheel for crossover, VSCA-GA chooses a parent with higher fitness. Let the higher fitness parent be  $\text{parent}_1$  then VSCA-GA calculates the number of distinct  $t$ -way and  $t_i$ -way interactions covered by each test case of  $\text{parent}_1$ . Subsequently, it checks whether the number of test cases to be exchanged (NTC) is equal to the number of test cases covering least number of distinct interactions. There are three possibilities:

1. The number of test cases covering the least number of distinct interactions is equal to NTC – In this case VSCA-GA performs crossover by exchanging the test cases that cover the least number of distinct interactions in  $\text{parent}_1$  by the respective test cases of  $\text{parent}_2$ . For instance, consider a system  $A$  having configuration  $(N; 2, 3^5, CA(3, 3^4))$  (the value of each component is labelled 0, 1, 2) and let  $N$  be 7 which means that the VSCA will consist of 7 test cases represented by  $\text{TC}_1, \text{TC}_2, \dots, \text{TC}_7$ . Each test case  $\text{TC}_i | 1 \leq i \leq 7$  contains a value 0/1/2 corresponding to each component. Let NTC be 2. After calculating the number of distinct interactions covered by each of the 7 test cases in  $\text{parent}_1$ , it has been found that two test cases  $\text{TC}_2$  and  $\text{TC}_5$  cover the least number of distinct interactions (i.e. 4) in  $\text{parent}_1$ . Hence, the number of test cases covering the least number of distinct interactions in  $\text{parent}_1$  is equal to NTC. Accordingly, a crossover is performed by exchanging  $\text{TC}_2$  and  $\text{TC}_5$  in  $\text{parent}_1$  with the respective test cases  $\text{TC}_2$  and  $\text{TC}_5$  of  $\text{parent}_2$  as shown in Figure 5(a).
2. NTC is greater than the number of test cases covering least number of distinct interactions. Here VSCA-GA selects first NTC test cases in  $\text{parent}_1$  when sorted in the ascending order by the number of distinct interactions covered by them and applies a crossover at these positions. For instance, in the aforementioned system  $A$ , let NTC be 3. Here, NTC is greater than the number of test cases covering the least number of interactions, so the crossover is performed by exchanging  $\text{TC}_2, \text{TC}_5$  and  $\text{TC}_4$  (which covers next least number of interactions i.e. 7 after  $\text{TC}_2$  and  $\text{TC}_5$ ) with the respective test cases of  $\text{parent}_2$  as shown in Figure 5(b).
3. The number of test cases covering the least number of distinct interactions is greater than NTC: Here VSCA-GA calculates all the  $t$ -way and  $t_i$ -way interactions covered by the respective test cases of  $\text{parent}_2$  and performs crossover by exchanging test cases that cover the maximum number of interactions not covered by  $\text{parent}_1$ . Again, for the aforementioned system  $A$ , two test cases  $\text{TC}_2$  and  $\text{TC}_5$  in  $\text{parent}_1$  cover the least number of distinct interactions (i.e., 4). Let NTC be 1, which is less than the number of test cases covering the least number of distinct interactions in  $\text{parent}_1$ . In this case, VSCA-GA calculates the number of interactions covered by the respective test cases of  $\text{parent}_2$ , in our case  $\text{TC}_2$  and  $\text{TC}_5$ . It is clear from Figure 5(c), that  $\text{TC}_5$  covers 5 interactions as compared to  $\text{TC}_2$  which covers only 1 interaction, not covered in  $\text{parent}_1$ . Hence, our strategy performs a crossover by exchanging test cases  $\text{TC}_5$  in  $\text{parent}_1$  and  $\text{parent}_2$ . By choosing the test case in  $\text{parent}_1$  that covers the least number of distinct interactions and exchanging it with a test case of  $\text{parent}_2$  that covers maximum number of interactions not covered by  $\text{parent}_1$ , we ensure that the resulting offspring is of better quality than its parent by

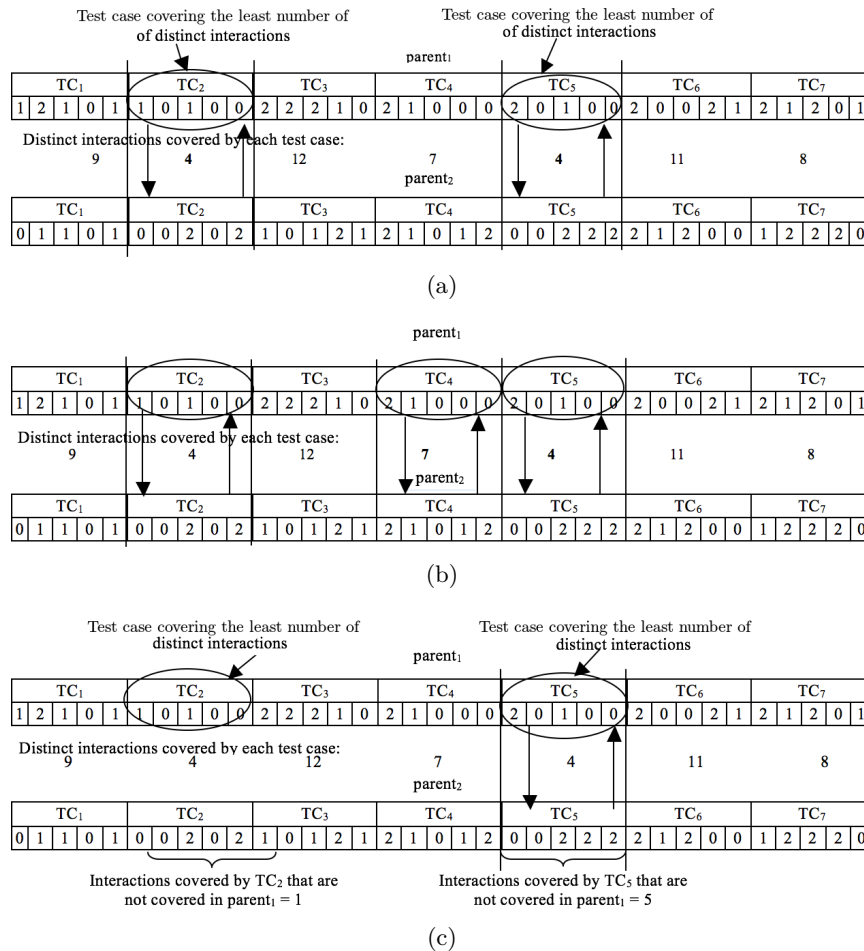


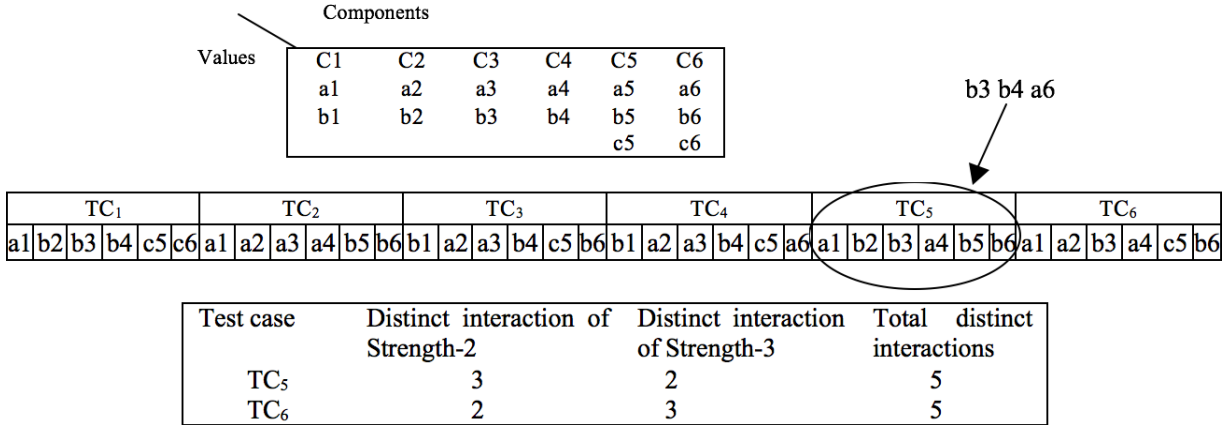
Figure 5. Multipoint crossover VSCA( $N; 2, 3^5, CA(3, 3^4)$ )

minimizing the loss of existing interactions and maximizing the gain.

### 5.3. A Greedy Approach to Perform Mutation

Mutation has a significant effect on the performance of GA as the mutation operator randomly modifies, with a given probability, one or more genes of a chromosome, thus increasing the diversity of the population and avoids getting stuck in the local minima. In traditional GA, every individual has an equal probability of getting mutated irrespective of their fitness [48]. Thus the probability of an individual with the highest fitness to be disrupted by a mutation is equal compared to the one with the lowest fitness. Hence a mutation strategy is needed to mutate an individual to maximize improvement in fitness

by minimizing fitness loss due to the mutation. Here, we present a greedy mutation strategy to perform a mutation. First, we select an individual VSCA<sub>f</sub> for a mutation and list all the  $t$ -way and  $t_i$ -way interactions left uncovered by the selected individual. Subsequently starting from the highest strength ( $t_h$ ) uncovered interaction, we check interactions of strength  $t_h$  that occurs multiple times in VSCA<sub>f</sub> and replace one of its occurrences with the uncovered  $t_h$  interaction in an attempt to increase its overall fitness. However, when an existing interaction is replaced with an uncovered interaction, then in addition to the gain of new interactions some old distinct interactions may get lost. Hence, to maximize the net gain after mutation, we calculate the number of distinct interactions covered by the multiple occurring interactions in the respective test cases and replaces the one which covers the

Figure 6. Greedy Mutation VSCA( $N; 2, 2^4 3^2, MCA(3, 2^2 3^2)$ )

least number of distinct interactions. In case more than one test case covers the least number of distinct interactions we replace the one which covers the least number of higher strength interactions. For instance, consider a system having configuration  $(N; 2, 2^4 3^2, MCA(3, 2^2 3^2))$  as shown in Figure 6. It is evident from Figure 6 that the VSCA selected for mutation does not cover the triplet ‘b3 b4 a6’. When examining the VSCA, it is found that the triplet ‘b3 a4 b6’ is covered by both TC<sub>5</sub> and TC<sub>6</sub>. Hence, one occurrence of ‘b3 a4 b6’ can be replaced by ‘b3 b4 a6’. To replace an occurrence of ‘b3 a4 b6’ by ‘b3 b4 a6’, the proposed greedy approach to mutation calculates the total number of distinct interactions of strength-2 and strength-3 covered by b3, a4 and b6 in TC<sub>5</sub> and TC<sub>6</sub>. In our case both TC<sub>5</sub> and TC<sub>6</sub> cover an equal number of distinct interactions, so the proposed approach replaces ‘b3 a4 b6’ in TC<sub>5</sub> which covers a smaller number of distinct interactions of higher strength ‘3’ by the uncovered triplet ‘b3 b4 a6’.

The overall VSCA-GA strategy can be found in Appendix.

## 6. Experimental Results

To assess the effectiveness of VSCA-GA strategy, we implemented the proposed strategy by extending an open source tool PWISEGen [49]. It is an open source tool written in Java to generate pair-wise (2-way) test set using GA. It does not provide support for the construction of CA of

strength  $t > 2$  as well as VSCA construction. We have extended PWISEGen by adding the capability to generate VSCA of strength up to 6 using the greedy strategies proposed in Section 5 to generate the initial population, crossover and mutation. We call it PWISEGen-VSCA.

To compare the performance of PWISEGen-VSCA with the existing greedy based strategies such as IPOG, PICT, ITCH, TVG, DA-RO, DA-FO, ParaOrder, TSG and AI based strategies such as SA, ACS, VS-PSTG, HSS and HSTCG based on VSCA size, we performed experiments on a set of four benchmark problems taken from Cohen et al. [9], Ahmed et al. [35] and Alsewari and Zamli [36]. As the VSCA size is not dependent on the execution environment, we compare our result directly with the results published in literature [9, 21–23, 31, 35, 36] with respect to VSCA size.

The results of experiments conducted to compare VSCA size on four VSCA configurations with various sub-configuration settings are shown in Table 3, Table 4, Table 5 and Table 6 respectively. Cells marked NA (not available) in the table signify that the results are not available in the publications and the cells marked ‘-’ signify that the tool/algorithm does not support the specified strength. As VSCA-GA produces non-deterministic results, we ran each configuration 30 times on PWISEGen-VSCA and reported the best VSCA size obtained over 30 runs. It can be observed from Table 3, Table 4, Table 5 and Table 6 that AI-based strategies generally perform better than their greedy counterparts.

Table 3. VSCA Size for VSCA configuration  $VSCA(N; 2, 3^{15}, C)$

{C}	No. of interactions	PICT	ITCH	DA-RO	DA-FO	Para Order	TVG	TSG	IPOG	SA	ACS	VS-PSTG	HSS	PWiseGen-VSCA Best	VSCA Average
$\phi$	945	35	31	21	20	33	22	20	21	16	19	19	20	16	16.33
CA(3, 3 <sup>3</sup> )	972	81	48	28	29	27	27	27	27	27	27	27	27	27	27
CA(3, 3 <sup>3</sup> ) <sup>2</sup>	999	729	59	28	29	33	30	27	28	27	27	27	27	27	27
CA(3, 3 <sup>3</sup> ) <sup>3</sup>	1026	785	69	28	30	33	30	28	29	27	27	27	27	27	27
CA(3, 3 <sup>4</sup> )	1053	105	59	32	34	27	35	33	38	27	27	30	27	27	27.9
CA(3, 3 <sup>5</sup> )	1215	131	62	40	42	45	41	40	41	33	38	38	38	33	34.13
CA(3, 3 <sup>6</sup> )	1485	146	61	46	46	49	48	48	48	34	45	45	45	40	42.13
CA(3, 3 <sup>7</sup> )	1890	154	68	53	53	54	54	51	51	41	48	49	51	47	48.2
CA(3, 3 <sup>9</sup> )	3213	177	94	60	60	62	62	59	63	50	57	57	60	57	57.33
CA(3, 3 <sup>15</sup> )	13230	83	132	70	78	82	81	82	83	67	76	74	77	74	75.8
CA(3, 3 <sup>4</sup> ), CA(3, 3 <sup>5</sup> ), CA(3, 3 <sup>6</sup> )	1863	1376	114	46	46	44	53	48	48	34	40	45	45	40	41.5
CA(4, 3 <sup>4</sup> )	1026	245	103	-	-	-	81	-	81	-	-	81	81	81	81
CA(4, 3 <sup>5</sup> )	1350	301	118	-	-	-	103	-	100	-	-	97	94	91	91
CA(4, 3 <sup>7</sup> )	3780	505	189	-	-	-	168	-	165	-	-	158	159	158	158.3
CA(5, 3 <sup>5</sup> )	1188	730	261	-	-	-	243	-	243	-	-	243	243	243	243
CA(5, 3 <sup>7</sup> )	6048	1356	481	-	-	-	462	-	461	-	-	441	441	441	441
CA(6, 3 <sup>6</sup> )	1674	2187	745	-	-	-	729	-	729	-	-	729	729	729	729

Table 4. VSCA Size for VSCA configuration  $VSCA(N; 2, 3^{20}10^2, C)$

{C}	No. of interactions	PICT	ITCH	DA-RO	DA-FO	Para Order	TVG	TSG	IPOG	SA	ACS	VS-PSTG	HSS	PWiseGen-VSCA Best	VSCA Average
$\phi$	3010	100	NA	100	100	100	101	100	102	100	100	102	106	100	100.33
CA(3, 3 <sup>20</sup> )	33790	940	NA	100	105	103	103	100	102	100	100	105	109	100	100
MCA(3, 3 <sup>20</sup> 10 <sup>2</sup> )	73990	423	NA	401	409	442	423	411	442	304	396	481	450	440	446
CA(4, 3 <sup>3</sup> 10 <sup>1</sup> )	3280	810	NA	-	-	-	270	-	270	-	-	270	270	270	274.53
MCA(5, 3 <sup>3</sup> 10 <sup>2</sup> )	5710	2430	NA	-	-	-	2700	-	2700	-	-	2700	2700	2700	2700
MCA(6, 3 <sup>4</sup> 10 <sup>2</sup> )	11110	7290	NA	-	-	-	8100	-	8100	-	-	8100	8100	8100	8100

Table 5. VSCA Size for VSCA configuration  $VSCA(N; 2, 4^35^36^2, C)$

{C}	No. of interactions	PICT	ITCH	DA-RO	DA-FO	Para Order	TVG	TSG	IPOG	SA	ACS	VS-PSTG	HST-CG	HSS	PWiseGen-VSCA Best	VSCA Average
$\phi$	663	43	48	41	40	49	44	39	40	36	41	42	43	42	37	38.93
CA(3, 4 <sup>3</sup> )	727	384	97	64	64	64	67	64	67	64	64	64	64	64	64	64
MCA(3, 4 <sup>3</sup> 5 <sup>2</sup> )	1507	781	164	131	132	141	132	125	132	100	104	124	120	116	120	121.3
CA(3, 5 <sup>3</sup> )	788	750	145	125	125	126	125	125	126	125	125	125	125	125	125	125
MCA(4, 4 <sup>3</sup> 5 <sup>1</sup> )	983	1920	354	-	-	-	320	-	320	-	-	320	320	320	320	320
CA(3, 4 <sup>3</sup> ), CA(3, 5 <sup>3</sup> )	852	8000	194	125	125	129	125	125	126	125	125	125	NA	125	125	125
MCA(4, 4 <sup>3</sup> 5 <sup>1</sup> ), MCA(4, 5 <sup>2</sup> 6 <sup>2</sup> )	1883	288000	1220	-	-	-	900	-	900	-	-	900	NA	900	900	900
CA(3, 4 <sup>3</sup> ), MCA(4, 5 <sup>3</sup> 6 <sup>1</sup> )	1477	48000	819	-	-	-	750	-	750	-	-	750	NA	750	750	750
MCA(4, 4 <sup>3</sup> 5 <sup>2</sup> )	2503	2874	510	-	-	-	496	-	479	-	-	472	454	453	458	459.23
MCA(3, 4 <sup>3</sup> 5 <sup>3</sup> 6 <sup>1</sup> )	4290	1266	254	207	211	247	237	197	215	171	201	206	NA	212	204	206.76
MCA(3, 5 <sup>1</sup> 6 <sup>2</sup> )	843	900	188	180	180	180	180	180	180	180	180	180	180	180	180	180
MCA(3, 4 <sup>3</sup> 5 <sup>3</sup> 6 <sup>2</sup> )	7080	261	312	256	261	307	302	239	263	214	255	260	264	263	260	260.33
MCA(5, 4 <sup>3</sup> 5 <sup>2</sup> )	2263	9600	1639	-	-	-	1600	-	1600	-	-	1600	NA	1600	1600	1600
MCA(5, 4 <sup>3</sup> 5 <sup>3</sup> )	11463	15048	2520	-	-	-	2583	-	2487	-	-	2430	2430	2430	2434	2436.53

Table 6. VSCA Size for VSCA configuration  $VSCA(N; 2, 10^1 9^1 8^1 7^1 6^1 5^1 4^1 3^1 2^1, C)$ 

{C}	No. of interactions	PICT	ITCH	Density	Para Order	TVG	IPOG	SA	ACS	VS-PSTG	HSS	PWiseGen-VSCA Best	VSCA Average
$\phi$	1266	102	119	NA	NA	99	90	NA	NA	97	94	92	93.96
MCA(3, $10^1 9^1 8^1$ )	1986	31256	765	NA	NA	720	720	NA	NA	720	720	720	720
MCA(3, $7^1 6^1 5^1$ )	1476	19515	301	NA	NA	210	211	NA	NA	210	210	210	210
MCA(3, $4^1 3^1 2^1$ )	1290	2397	140	NA	NA	99	90	NA	NA	97	94	92	92.6
MCA(3, $10^1 9^1 8^1 7^1$ )	3680	22878	806	NA	NA	784	772	NA	NA	742	740	740	745.03
MCA(3, $10^1 9^1 8^1$ ), MCA(3, $7^1 6^1 5^1$ )	2196	NA	947	NA	NA	720	720	NA	NA	720	720	720	720
MCA(3, $10^1 9^1 8^1$ ), MCA(3, $7^1 6^1 5^1$ ), MCA(3, $4^1 3^1 2^1$ )	2220	NA	968	NA	NA	720	720	NA	NA	720	720	720	720
MCA(4, $5^1 4^1 3^1 2^1$ )	1386	1200	237	-	-	123	142	-	-	120	120	120	120
MCA(5, $10^1 9^1 4^1 3^1 2^1$ )	3426	124157	2276	-	-	2160	2160	-	-	2160	2160	2160	2160
MCA(6, $7^1 6^1 5^1 4^1 3^1 2^1$ )	6306	NA	5157	-	-	5040	5043	-	-	5040	5040	5040	5040

Table 7. VSCA generation time (in seconds) for VSCA configuration  $VSCA(N; 2, 3^{15}, C)$ 

{C}	IPOG	TVG	PWiseGen-VSCA
$\phi$	0.077	0.056	2.976
CA(3, $3^3$ )	0.009	0.071	1.32
CA(3, $3^3$ ) <sup>2</sup>	0.025	0.062	13.5
CA(3, $3^3$ ) <sup>3</sup>	0.023	0.076	5.424
CA(3, $3^4$ )	0.012	0.088	60.042
CA(3, $3^5$ )	0.03	0.098	11.4
CA(3, $3^6$ )	0.013	0.141	48.06
CA(3, $3^7$ )	0.023	0.161	57.6
CA(3, $3^9$ )	0.019	0.304	97.8
CA(3, $3^{15}$ )	0.048	2.008	211.08
CA(3, $3^4$ ), CA(3, $3^5$ ), CA(3, $3^6$ )	0.008	0.302	30.78
CA(4, $3^4$ )	0.025	0.108	11.4
CA(4, $3^5$ )	0.011	0.189	5431.8
CA(4, $3^7$ )	0.013	0.862	9003.6
CA(5, $3^5$ )	0.015	0.499	6.6
CA(5, $3^7$ )	0.046	3.853	12035.4
CA(6, $3^6$ )	0.093	1.388	19.44
CA(6, $3^7$ )	0.078	11.685	21183.6

Table 9. VSCA generation time (in seconds) for VSCA configuration  $VSCA(N; 2, 4^3 5^3 6^2, C)$ 

{C}	IPOG	TVG	PWiseGen-VSCA
$\phi$	0.002	0.035	2.37
CA(3, $4^3$ )	0.002	0.041	2.106
MCA(3, $4^3 5^2$ )	0.002	0.156	433.8
CA(3, $5^3$ )	0.005	0.077	0.39
MCA(4, $4^3 5^1$ )	0.016	0.189	18.4704
CA(3, $4^3$ ), CA(3, $5^3$ )	0.001	0.082	0.5772
MCA(4, $4^3 5^1$ ), MCA(4, $5^2 6^2$ )	0.047	1.136	52.6344
CA(3, $4^3$ ), MCA(4, $5^3 6^1$ )	0.032	0.699	35.9112
MCA(4, $4^3 5^2$ )	0.023	0.917	6992.4
MCA(3, $4^3 5^3 6^1$ )	0.015	0.733	1173.6
MCA(3, $5^1 6^2$ )	0.003	0.089	0.45
MCA(3, $4^3 5^3 6^2$ )	0.011	1.621	1579.2
MCA(5, $4^3 5^2$ )	0.11	2.84	30.6
MCA(5, $4^3 5^3$ )	0.296	26.193	7485.6

Table 8. VSCA generation time (in seconds) for VSCA configuration  $VSCA(N; 2, 3^{20} 10^2, C)$ 

{C}	IPOG	TVG	PWiseGen-VSCA
$\phi$	0.012	0.636	8.9544
CA(3, $3^{20}$ )	0.039	5.972	915
MCA(3, $3^{20} 10^2$ )	0.085	13.559	3813.84
CA(4, $3^3 10^1$ )	0.061	1.491	1546.8
VSCA(5, $3^3 10^2$ )	0.343	27.409	274.8
VSCA(6, $3^4 10^2$ )	1.684	208.681	378

Table 10. VSCA generation time (in seconds) for VSCA configuration  $VSCA(N; 2, 10^1 9^1 8^1 7^1 6^1 5^1 4^1 3^1 2^1, C)$ 

{C}	IPOG	TVG	PWiseGen-VSCA
$\phi$	0.003	0.414	7.8
MCA(3, $10^1 9^1 8^1$ )	0.003	0.865	5.148
MCA(3, $7^1 6^1 5^1$ )	0.007	0.241	6.72
MCA(3, $4^1 3^1 2^1$ )	0.002	0.131	37.518
MCA(3, $10^1 9^1 8^1 7^1$ )	0.044	2.169	2586.24
MCA(3, $10^1 9^1 8^1$ ), MCA(3, $7^1 6^1 5^1$ )	0.031	0.893	6.0684
MCA(3, $10^1 9^1 8^1$ ), MCA(3, $7^1 6^1 5^1$ ), MCA(3, $4^1 3^1 2^1$ )	0.028	0.894	7.7376
MCA(4, $5^1 4^1 3^1 2^1$ )	0.003	0.021	635.22
MCA(5, $10^1 9^1 4^1 3^1 2^1$ )	0.234	7.504	85.8
MCA(6, $7^1 6^1 5^1 4^1 3^1 2^1$ )	0.733	38.548	484.02

When AI-based strategies are compared to each other, we can see that SA and ACS support construction of VSCA of strength  $t \leq 3$  only whereas VS-PSTG supports construction of VSCA of strength up to 6. The published results [36] show that unlike other greedy and AI-based strategies, HSS support construction of VSCA of strength up to 15 but nothing is mentioned about the efficiency of HSS in terms of VSCA generation time. From Table 3, we can infer that PWISEGen-VSCA outperforms ACS whereas the results in Table 4 and Table 5 are comparable. Although PWISEGen-VSCA supports construction of higher interaction strength VSCA however, VSCA generation time increases with the increase in interaction strength which makes it infeasible to generate higher strength VSCAs. It is evident from Tables 3-6 that PWISEGen-VSCA generates better results as compared to VS-PSTG, HSTCG and HSS. From Tables 3-6, it is clear that SA outperforms existing state-of-the-art strategies for lower interaction strength ( $t \leq 3$ ), however, the results generated by PWISEGen-VSCA are equal or close to SA.

Finally from Tables 3-6, we can conclude that PWISEGen-VSCA generates optimal VSCA most of the time as compared to other greedy and meta-heuristic techniques for strength  $\leq 6$ .

It is difficult to compare PWISEGen-VSCA with the existing state-of-the-art algorithms in terms of VSCA generation time, as the generation time is dependent on the running environment and most of the algorithm implementations are not publicly available. To perform a fair comparison, we restrict the comparison of VSCA generation time against publicly available algorithm implementation: ACTS (IPOG) and TVG. These tools are run on Windows using an INTEL Pentium Dual Core 1.73 GHZ processor with 1.00 GB of memory. The results of comparison made on the dataset of Tables 3-6 with respect to VSCA generation time (in seconds) are shown in Tables 7-10 respectively. It is evident from Tables 7-10 that PWISEGen-VSCA requires more time to construct VSCA as compared to ACTS (IPOG) and TVG, however, the extra

time consumed by PWISEGen-VSCA allowed the construction of VSCAs of smaller size.

## 7. Threats to Validity

One important threat to validity of the effectiveness of our approach is that we could not use any sophisticated statistical hypothesis tests such as Welch's t-test to assess and compare PWISEGen-VSCA with the existing meta-heuristic techniques for constructing VSCA as we do not have access to the source code of any of them. Also, we could not compare the efficiency of PWISEGen-VSCA in terms of VSCA generation time with the existing meta-heuristic techniques because of the above mentioned reason.

## 8. Conclusion and Future Work

In this paper we have presented and evaluated VSCA-GA, a strategy based on GA to construct optimal VSCA for  $t$ -way testing. The strategy is implemented in PWISEGen-VSCA. Our strategy exploits the strength of both greedy and meta-heuristic techniques by integrating greedy technique with GA. The experiments conducted on a set of benchmark problems show that PWISEGen-VSCA outperforms the existing state-of-the-art algorithms except SA in terms of VSCA sizes. However, our results are comparable to SA which generates VSCA for strength  $t$  up to 3 whereas VSCA-GA constructs VSCA for strength  $t$  up to 6.

In future, we plan to construct VSCA to handle feature constraints and try to improve the efficiency of PWISEGen-VSCA to construct higher strength VSCA.

## References

- [1] D.M. Cohen, S.R. Dalal, M.L. Fredman, and G.C. Patton, "The AETG system: An approach to testing based on combinatorial design," *IEEE Transactions on Software Engineering*, Vol. 23, No. 7, 1997, pp. 437-444.

- [2] A. Hartman, "Software and hardware testing using combinatorial covering suites," in *Graph Theory, Combinatorics and Algorithms*. Springer, 2005, pp. 237–266.
- [3] D.M. Cohen, S.R. Dalal, A. Kajla, and G.C. Patton, "The automatic efficient test generator (AETG) system," in *5th International Symposium on Software Reliability Engineering*. IEEE, 1994, pp. 303–309.
- [4] D.M. Cohen, S.R. Dalal, J. Parelius, and G.C. Patton, "The combinatorial design approach to automatic test generation," *IEEE software*, No. 5, 1996, pp. 83–88.
- [5] K. Burr and W. Young, "Combinatorial test techniques: Table-based automation, test generation and code coverage," in *Proc. of the Intl. Conf. on Software Testing Analysis & Review*. San Diego, 1998.
- [6] S.R. Dalal, A. Jain, N. Karunanithi, J. Leaton, C.M. Lott, G.C. Patton, and B.M. Horowitz, "Model-based testing in practice," in *Proceedings of the 21st international conference on Software engineering*. ACM, 1999, pp. 285–294.
- [7] D.R. Kuhn, D.R. Wallace, and A.M. Gallo Jr, "Software fault interactions and implications for software testing," *IEEE Transactions on Software Engineering*, Vol. 30, No. 6, 2004, pp. 418–421.
- [8] D.R. Kuhn and M.J. Reilly, "An investigation of the applicability of design of experiments to software testing," in *27th Annual NASA Goddard/IEEE Software Engineering Workshop*. IEEE, 2002, pp. 91–95.
- [9] M.B. Cohen, P.B. Gibbons, W.B. Mugridge, C.J. Colbourn, and J.S. Collofello, "A variable strength interaction testing of components," in *27th Annual International Computer Software and Applications Conference*. IEEE, 2003, pp. 413–418.
- [10] Y. Lei and K.C. Tai, "In-parameter-order: A test generation strategy for pairwise testing," in *Third IEEE International High-Assurance Systems Engineering Symposium*. IEEE, 1998, pp. 254–261.
- [11] C. Nie and H. Leung, "A survey of combinatorial testing," *ACM Computing Surveys (CSUR)*, Vol. 43, No. 2, 2011, p. 11.
- [12] A. Hedayat, N. Sloane, and J. Stufken, *Orthogonal Arrays*, ser. Springer Series in Statistics. Springer, New York, 1999.
- [13] R. Mandl, "Orthogonal latin squares: an application of experiment design to compiler testing," *Communications of the ACM*, Vol. 28, No. 10, 1985, pp. 1054–1058.
- [14] M.B. Cohen, P.B. Gibbons, W.B. Mugridge, and C.J. Colbourn, "Constructing test suites for interaction testing," in *25th International Conference on Software Engineering*. IEEE, 2003, pp. 38–48.
- [15] J.H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control and artificial intelligence*. MIT press, 1992.
- [16] K.F. Man, K.S. Tang, and S. Kwong, "Genetic algorithms: concepts and applications," *IEEE Transactions on Industrial Electronics*, Vol. 43, No. 5, 1996, pp. 519–534.
- [17] S.K. Khalsa and Y. Labiche, "An orchestrated survey of available algorithms and tools for combinatorial testing," in *IEEE 25th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2014, pp. 323–334.
- [18] C.J. Colbourn, M.B. Cohen, and R. Turban, "A deterministic density algorithm for pairwise interaction coverage," in *IASTED Conf. on Software Engineering*. Citeseer, 2004, pp. 345–352.
- [19] J. Arshem. TVG download web page. (2009). [Online]. <http://sourceforge.net/projects/tvg>
- [20] J. Czerwonka, "Pairwise testing in real world: Practical extensions to test case generator," in *PNSQC'06: Proceedings of 24th Pacific Northwest Software Quality Conference*, 2006, pp. 419–430.
- [21] Z. Wang, B. Xu, and C. Nie, "Greedy heuristic algorithms to generate variable strength combinatorial test suite," in *The Eighth International Conference on Quality Software*. IEEE, 2008, pp. 155–160.
- [22] Z. Wang and H. He, "Generating variable strength covering array for combinatorial software testing with greedy strategy," *Journal of Software*, Vol. 8, No. 12, 2013, pp. 3173–3181.
- [23] S.A. Abdullah, Z.H. Soh, and K.Z. Zamli, "Variable-strength interaction for  $t$ -way test generation strategy," *International Journal of Advances in Soft Computing & Its Applications*, Vol. 5, No. 3, 2013.
- [24] M. Forbes, J. Lawrence, Y. Lei, R.N. Kacker, and D.R. Kuhn, "Refining the in-parameter-order strategy for constructing covering arrays," *Journal of Research of the National Institute of Standards and Technology*, Vol. 113, No. 5, 2008, pp. 287–297.
- [25] L. Yu, Y. Lei, R.N. Kacker, and D.R. Kuhn, "ACTS: A combinatorial test generation tool," in *IEEE Sixth International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2013, pp. 370–375.



- [26] S. Ali, L.C. Briand, H. Hemmati, and R.K. Panesar-Walawege, "A systematic review of the application and empirical investigation of search-based test case generation," *IEEE Transactions on Software Engineering*, Vol. 36, No. 6, 2010, pp. 742–762.
- [27] B.J. Garvin, M.B. Cohen, and M.B. Dwyer, "Evaluating improvements to a meta-heuristic search for constrained interaction testing," *Empirical Software Engineering*, Vol. 16, No. 1, 2011, pp. 61–102.
- [28] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Computing Surveys (CSUR)*, Vol. 35, No. 3, 2003, pp. 268–308.
- [29] B. Jenkins. Jenny download web page. (2005). [Online]. <http://burtleburtle.net/bob/math/jenny.html>
- [30] T. Shiba, T. Tsuchiya, and T. Kikuno, "Using artificial life techniques to generate test cases for combinatorial testing," in *Proceedings of the 28th Annual International Computer Software and Applications Conference*. IEEE, 2004, pp. 72–77.
- [31] X. Chen, Q. Gu, A. Li, and D. Chen, "Variable strength interaction testing with an ant colony system approach," in *APSEC'09. Asia-Pacific Software Engineering Conference*. IEEE, 2009, pp. 160–167.
- [32] L. Gonzalez-Hernandez, N. Rangel-Valdez, and J. Torres-Jimenez, "Construction of mixed covering arrays of variable strength using a tabu search approach," in *Combinatorial Optimization and Applications*. Springer, 2010, pp. 51–64.
- [33] J.D. McCaffrey, "An empirical study of pairwise test set generation using a genetic algorithm," in *Seventh International Conference on Information Technology: New Generations (ITNG)*. IEEE, 2010, pp. 992–997.
- [34] P. Flores and Y. Cheon, "P WiseGen: Generating test cases for pairwise testing using genetic algorithms," in *IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, Vol. 2. IEEE, 2011, pp. 747–752.
- [35] B.S. Ahmed and K.Z. Zamli, "A variable strength interaction test suites generation strategy using particle swarm optimization," *Journal of Systems and Software*, Vol. 84, No. 12, 2011, pp. 2171–2185.
- [36] A.R.A. Alsewari and K.Z. Zamli, "Design and implementation of a harmony-search-based variable-strength t-way testing strategy with constraints support," *Information and Software Technology*, Vol. 54, No. 6, 2012, pp. 553–568.
- [37] J. LI, D. XING, and Y. ZHAO, "Combinatorial test suite generation of variable strength based on harmony search," *Journal of Network & Information Security*, Vol. 4, No. 2, 2013, pp. 177–188.
- [38] X. Chen, Q. Gu, J. Qi, and D. Chen, "Applying particle swarm optimization to pairwise testing," in *IEEE 34th Annual Computer Software and Applications Conference (COMPSAC)*. IEEE, 2010, pp. 107–116.
- [39] J. Stardom, "Metaheuristics and the search for covering and packing arrays," Ph.D. dissertation, Simon Fraser University, 2001.
- [40] *Genetic Algorithms in Search, Optimization, and Machine Learning*, 1st ed. Reading, Mass: Addison-Wesley Professional, Jan. 1989.
- [41] H. Maaranen, K. Miettinen, and M.M. Mäkelä, "Quasi-random initial population for genetic algorithms," *Computers & Mathematics with Applications*, Vol. 47, No. 12, 2004, pp. 1885–1895.
- [42] Y.W. Leung and Y. Wang, "An orthogonal genetic algorithm with quantization for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, Vol. 5, No. 1, 2001, pp. 41–53.
- [43] H. Maaranen, K. Miettinen, and A. Penttinen, "On initial populations of a genetic algorithm for continuous optimization problems," *Journal of Global Optimization*, Vol. 37, No. 3, 2007, pp. 405–436.
- [44] S. Rahnamayan, H.R. Tizhoosh, and M.M. Salama, "A novel population initialization method for accelerating evolutionary algorithms," *Computers & Mathematics with Applications*, Vol. 53, No. 10, 2007, pp. 1605–1614.
- [45] H. Wang, Z. Wu, J. Wang, X. Dong, S. Yu, and C. Chen, "A new population initialization method based on space transformation search," in *Fifth International Conference on Natural Computation*, Vol. 5. IEEE, 2009, pp. 332–336.
- [46] P. Bansal, S. Sabharwal, S. Malik, V. Arora, and V. Kumar, "An approach to test set generation for pair-wise testing using genetic algorithms," in *Search Based Software Engineering*. Springer, 2013, pp. 294–299.
- [47] D. Ortiz-Boyer, C. Hervás-Martínez, and N. García-Pedrajas, "CIXL2: A crossover operator for evolutionary algorithms based on population features." *J. Artif. Intell. Res. (JAIR)*, Vol. 24, 2005, pp. 1–48.
- [48] S.M. Libelli and P. Alba, "Adaptive mutation in genetic algorithms," *Soft Computing*, Vol. 4, No. 2, 2000, pp. 76–80.
- [49] P. Flores. P WiseGen download web page. (2010). [Online]. <https://code.google.com/p/pwisegen/>

## Appendix

**Input:** VSCA configuration:  $(t, k, (v_1, v_2, \dots, v_k), C)$ , VSCA size:  $N$ , Population Size:  $P_{size}$ , Maximum Number of Generations:  $NOG$ , Number of Reproductions:  $NOR$ , Number of Test Cases for Crossover:  $NTC$

**Output:** Optimal VSCA

Algorithm 1. VSCA-GA

```

1: procedure VSCA-GA
2:   Initialize  $G := 0$  ▷ generation number
3:   ▷ Generate initial population  $pop_1$ 
4:   for each VSCA $_f$  in  $pop_1$  do ▷  $1 \leq f \leq P_{size}$ 
5:     Create an interaction list L of all  $t$ -way and  $t_i$ -way interactions between all components  $C_m$  ▷  $1 \leq m \leq k$ 
6:     for  $m = 1$  to  $k$  do
7:       for  $j = 1$  to  $v_m$  do
8:         Store the number of uncovered interactions of value  $val_{mj}$  of component  $C_m$  in  $N_{uncovered}(val_{mj})$ 
9:       end for
10:    end for
11:    for each component  $C_m$  do
12:      Assign each value  $val_{mj}$  an equal probability of selection P ( $val_{mj}$ )
13:    end for
14:    ▷ Generate first test case
15:    Create the first test case  $tc_{f1}$  by selecting a value for each component  $C_m$  randomly
16:    Let  $val_{ms}$  is the selected value of component  $C_m$ 
17:    ▷ Generate remaining  $(N - 1)$  test cases
18:    Initialize  $i := 2$ 
19:    for  $i = 2$  to  $N$  do
20:      for  $m = 1$  to  $k$  do
21:        Update interaction list L by eliminating the interactions covered by  $val_{ms}$  in test case  $tc_{f(i-1)}$ 
22:        Store the number of remaining interactions of  $val_{ms}$  in  $N'_{uncovered}(val_{ms})$ 
23:        Decrease probability of selection of value  $val_{ms}$  of  $C_m$  selected in test case  $tc_{f(i-1)}$  according to equation 3
24:        Update probability of remaining values of  $C_m$  according to equation 5.
25:      end for
26:      for each component  $C_m$  do
27:        Generate a random number between 1 to 100
28:        Create test case  $tc_{fi}$  by selecting a value of  $C_m$  based on the interval in which the random number falls
29:      end for
30:    end for
31:  end for
32:  Calculate fitness of each VSCA $_f$  in  $pop_1$  using equation 1
33:   $G \leftarrow G + 1$ 
34:  while solution not found and  $G \leq NOG$  do
35:    ▷ Perform Crossover
36:    Initialize  $counter := 1$ 
37:    while  $counter \leq NOR$  do
38:      Select two parent VSCA from population  $pop_{G-1}$  using Roulette Wheel Selection
39:      Let  $parent_1$  is the parent VSCA having higher fitness among the two parents
40:      Calculate the number of distinct pairs covered by each test case of  $parent_1$ 
41:      if  $NTC <$  number of test cases covering least number of distinct interactions then
42:        Calculate the number of interactions covered by respective test cases of  $parent_2$ 
43:        Select  $NTC$  test cases of  $parent_2$  that cover maximum number of interactions not covered by  $parent_1$ 
44:      else if  $NTC >$  number of test cases covering least number of distinct interactions then

```

```
45:         Select NTC test cases in  $parent_1$  when sorted in ascending order by the number of distinct
           interactions covered by them
46:     else
47:         Select NTC test cases that covers least number of distinct interactions in  $parent_1$ 
48:     end if
49:     Perform crossover between  $parent_1$  and  $parent_2$  by exchanging selected test cases to generate
           offsprings  $os_1, os_2$ 
50:      $\triangleright$  Perform Mutation
51:     Apply greedy mutation on  $os_1$  and  $os_2$  as discussed in Section 5.3
52:     Replace weaker VSCA in  $pop_{G-1}$  by  $os_1, os_2$  to form new population  $pop_G$ 
53:      $counter \leftarrow counter + 1$ 
54: end while
55: Calculate fitness of each  $VSCA_f$  in the  $pop_G$  using equation 1
56: if solution found then
57:     break
58: else
59:      $G \leftarrow G + 1$ 
60: end if
61: end while
62: if generations >  $NOG$  then
63:     return (solution not found)
64: else
65:     return  $VSCA_f$   $\triangleright$  VSCA with 100% fitness
66: end if
67: end procedure
```



# Model Driven Web Engineering: A Systematic Mapping Study

Karzan Wakil\*, Dayang N. A. Jawawi\*\*

\**Software Engineering Department, Faculty of Computing, University of Technology Malaysia*

\*\**Software Engineering Department, Faculty of Computing, University Technology Malaysia*

karzanwakil@gmail.com, dayang@utm.my

## Abstract

Background: Model Driven Web Engineering (MDWE) is the application of the model driven paradigm to the domain of web software development, where it is particularly helpful because of the continuous evolution of Web technologies and platforms. Objective: In this paper, we prepare a survey of primary studies on MDWE to explore current work and identify needs for future research. Method: Systematic mapping study uses for finding the most relevant studies and classification. In this study, we found 289 papers and a classification scheme divided them depending on their research focus, contribution type and research type. Results: The papers of solution proposal (20%) research type are majority. The most focused areas of MDWE appear to be: Web Applicability (31%), Molding and Notation (19%), and Services and Oriented (18%). The majority of contributions are methods (33%). Moreover, this shows MDWE as a wide, new, and active area to publications. Conclusions: Whilst additional analysis is warranted within the MDWE scope, in literature, composition mechanisms have been thoroughly discoursed. Furthermore, we have witnessed that the recurrent recommendation for Validation Research, Solution Proposal and Philosophical Papers has been done through earlier analysis.

**Keywords:** model driven web engineering, MDWE, web engineering, systematic mapping study

## 1. Introduction

MDWE is the application of the model driven paradigm in the web domain [1–5]. The advent of a new area of software engineering, focusing on the special features of the web environment, was undertaken by the research community at the beginning of the 1990s. At the beginning, this research focused on new methods, models and notations which were used in hypermedia systems. However, later the target were web-based systems that were presented through some approaches which included the Hypermedia Design Model (HDM) [6] and the Object-Oriented Hypermedia Design Method (OOHDM) [7]. There are a number of comparative studies and sur-

veys which investigate the evolution of this area and have drawn attention to areas where further research is needed to address a number of clearly-identified gaps and shortcomings. Within the Web engineering community, a number of research groups are working towards suitable resolutions to these gaps, which can be broadly classified within three areas: 1) There is a wide variety of Web development methodologies, using a multiplicity of different notations, models and techniques. 2) No single Web development approach provides coverage for the whole life cycle. 3) There still remains a lack of tool supports for Web development methodologies [8–12]. Instead of traditional or conventional methods, specialized web development methods were used [13].

The application of the Model-Driven Architecture (MDA) initiative has been applied to numerous domains since 2001. In general, it works better than those areas controlled by functional requirements, well-structured models, and accurate separation of concerns and standard platforms. MDA has created potent advantages in which web engineering has essentially been shown to be an application domain. As new platforms emerge and changes in technologies occur continuously in this area, MDA mainly permits successful highlighting of interoperability, model evolution and adaptation issues of web systems [14]. Due to the rapid evolution of web technologies and platforms, MDWE was also developed by applying independent models, such as the content, navigation, process, and presentation issues possessing various issues of web applications. Moreover, these models are unified and changed to codes, conversely. These codes consist of web pages, configuration data for web frameworks, and also traditional program codes [1].

For the design and advancement of many types of web applications, MDWE approaches already offer outstanding methodologies and tools. By applying independent models (including navigation, presentation, data and others), these approaches reveal diverse issues, and are sustained by model compilers that generate a vast majority of the application's web pages and the logic centered on these models [15].

The specification of the application is built up step by step by alternating automatic generation and manual elaboration steps, from the Computational Independent Model (CIM), to a Platform Independent Model (PIM), to a Platform Specific Model (PSM), to code. Today, most approaches based on MDA are 'elaboration' approaches, which have to deal with the problems of model and code synchronization. Some tools support the regeneration of the higher-level models from the lower-level models [1].

A systematic mapping study is a way of identifying and classifying research related to the topic, it has been adapted from other disciplines to software engineering by Kitchenham

and Charters [16]. When used for a specific research area, it categorizes different types of research reports in various dimensions and often provides a map of its results. Systematic mapping studies have been recommended mostly when little relevant evidence is found during the initial study of the domain, or if the topic to be investigated is very broad [16]. In contrast to systematic literature reviews, systematic mapping studies are conducted at a coarse-grained level. They aim only to find and identify evidence relating to research questions, and to identify research gaps in order to direct future research. In this context, we believed it would be appropriate to conduct a systematic mapping study, since model-driven web engineering appears to be a broader concept with multiple research focus areas. In this paper, a Systematic Mapping Study for MDWE is presented from the perspective of the guidelines extracted from the reports published by Kitchenham and Charters [16] and Biolchini et al. [17].

There are a great number of journals, conferences and workshops within the web engineering area and MDWE fields that were published. These included the Journal of Web Engineering (JWE) [18], the International Journal of Web Engineering (IJWE) [19], and the International Conference on Web Engineering (ICWE) [20]. Wherever this topic is mentioned, it is hard to get a comprehensive overview of the state of the research. For controlling the review papers and understanding the subjects of the papers, we need a systematic mapping study in MDWE.

Following this introduction, this paper has been structured as follows: In Section 2, we present a short overview of the context in which the current study has been conducted, and we justify its needs. Section 3 describes how the systematic mapping methodology has been applied. The classification schemes and their various dimensions are discussed in Section 4. Section 5 is dedicated to presenting the results of mapping the selected primary studies, and the discussion of research questions. We discuss the overall results and identify the potential limitations of our study in Section 6. Section 7 consists of a conclusion and suggestions for future work.

## 2. Background and Motivation

There already exist literature surveys and systematic review works in this field resulting from the swift progressed in web engineering and MDWE. Some investigators completed going through MDWE methodologies [21], introducing a crucial assessment of earlier studies of traditional web methodologies and highlighting the capability of the MDWE paradigm [2] as well as systematic review of web engineering research [22].

Several of the MDWE methods that have been suggested are presented by Jesús and John, 2012 [21], who consider and investigate the strengths and weaknesses of such methods associated with the present trends and best practices on Model-Driven Engineering (MDE). Introducing every approach and investigating the models, they suggest signifying web applications, the architectural factors in the changes, and the application of present web user interface technologies in the code outcome are their aim. This is accomplished for the purpose of creating potential research strategies for upcoming works on the MDWE area [21].

A crucial review of the earlier studies of classical web methodologies is presented by Aragón et al. 2012 [2], who highlights the capability of the MDWE paradigm to highlight lengthy overdue issues of web development, encompassing research and enterprise. With respect to the terms extracted from the literature, the chosen key MDWE development approaches are investigated and matched. The paper argues that certain classical gaps can be enhanced with MDWE and shows that this new tendency introduces a stimulating as well as novel method to create web systems inside practical projects. However, this paper presents a general assessment of the situation and investigates how MDE can overcome the classical issues identified in web development in the past years [2], as can be concluded from this introduction.

For the purposes of investigating the rigor of claims ascending from web engineering research, Mendes, 2005, applies a systematic literature review. The rigor is measured by applying a stan-

dard spooled from software engineering research. The outcomes have indicated that just 5% of 173 papers reviewed by them could be considered methodologically rigorous. On top of showing their outcomes, they offer proposals for the betterment of web engineering research founded on lessons picked up by the software engineering fraternity [22].

In many areas, systematic review has achieved great attention amongst researchers these days. In the application investigating statistical sciences, psychology sciences, industrial-organizational psychological sciences, education, medicine, health sciences domain, and software engineering, it is extensively used. The idea of Evidence-based software engineering founded on medical practice by applying systematic review was assessed by [23], and presents a guideline for a systematic review that is conducive for software engineering investigators [17]. As a result, numerous systematic reviews were carried out in software engineering after words and several article were published in the web engineering domain, such as: Mendes reviewed 173 papers, only 5% of all papers reviewed were designed properly, were based on a real scenario [22], Alfonso at al. to create a comprehensive review and synthesis of the current state of the art in the literature related to the engineering requirements in the web domain. To do this, a total of 3059 papers published in the literature and extracted from the most relevant scientific sources were considered, of which 43 were eventually analyzed in depth in accordance with the adopted systematic review process [24]. Insfran and Fernandez presented a systematic review of usability evaluation methods for Web development; total of 51 research papers have been reviewed from an initial set of 410 papers; the results show that 45% of the reviewed papers reported the use of evaluation methods [25].

Where continued investigation is required to highlight a number of visibly recognized gaps, and weaknesses, a few comparative studies and reviews of web development methodologies have gained attention in these areas. Several investigative groups within the web engineering fra-

ternity are pushing towards appropriate solutions to these gaps which, already laid out in the previous section, can be categorized into three parts [2]:

- Applying a diversity of dissimilar notations, models and techniques, there is a vast range of web development methodologies.
- The non-presence of a single all-in-one answer because no single web development approach offers coverage for the entire life cycle, which means that web developers need to mix-and-match factors from diverse approaches.
- Web development methodologies remain inadequately supported via tool support. On the contrary, there are inadequate methodical investigation and design components by way of the majority of development tools.

By implementing a Model-Driven Development (MDD) paradigm, for instance MDWE, these problems can be highlighted to a certain degree. Investigating approaches adapted to the model-driven paradigm is the chief focus which makes an innovative input from the review paper. Concepts play the utmost significance in MDWE, free of their representations. MDWE suggests applying metamodels that are platform-independent together with the representation of ideas. A set of transformations and relations among ideas that facilitate active development and guarantees uniformity between models supports the development process. In some regions of software engineering and development, the model-driven paradigm is being applied with outstanding outcomes. This indicates it could also be adapted for web engineering. For example, MDE offers an appropriate way to guarantee traceability and product derivation in software product's lines [2, 26, 27]. Several articles on the secondary study in the area of web engineering, readied by the earlier reviewer, with different sides of web engineering methodologies and MDWE, presented certain problems and methods for the development of web applications. At times, they did not present a systematic mapping for MDWE as it seemed a concrete work for MDWE.

Systematic mapping studies belong to the Evidence-based Software Engineering (EBSE) paradigm [28]. They provide new, empirical and systematic methods of research. Although several studies have been reported in the broader MDWE (e.g. [2, 14, 15, 21]), we are not aware of any systematic mapping study that has been conducted in this field. Given the fact that various types of research have appeared addressing varying focus areas at different levels of granularity related to a broader topic of MDWE, there is a need for a more systematic investigation of the topic. Therefore, the current study is intended to contribute to MDWE through a systematic and evidence-based approach. This study may help researchers in the field of MDWE through providing an overview of the current research in the area. Furthermore, it may serve as the first step towards more thorough examination of the topics addressed in it with the help of systematic literature reviews.

### 3. Research Method

The process of continuing a systematic mapping study in software engineering was expounded by Petersen et al. [29]. By taking into account their guidelines, we carried out the present study. Referring to our subject matter, we discovered demarcating certain explicit schemes apart from utilizing the classification schemes suggested in their task for some areas. As highlighted in Figure 1, it is based on the crucial process steps of (1) Defining research questions. (2) Defining search strategy. (3) Screening of primary studies. (4) Defining classification schemes. (5) Mapping of studies.

#### 3.1. Research Questions

In web engineering, acquiring a general idea of the present analysis within the scope of the model driven is the objective of this study. To clarify this aim, we demarcated three research questions:

- RQ1: What MDWE subject matters are the most analyzed ones and how far have these



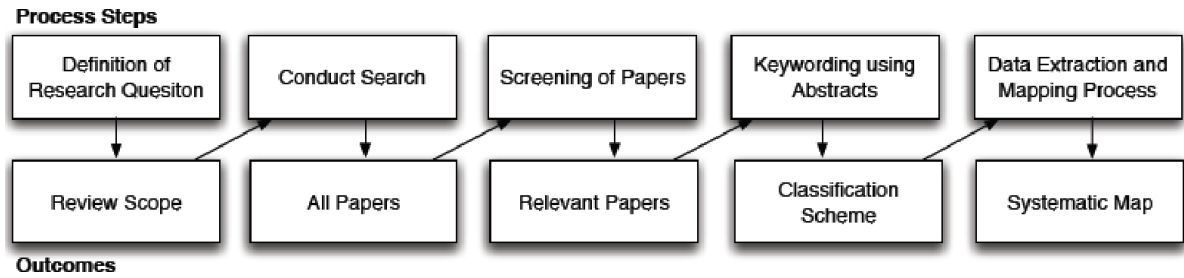


Figure 1. The Systematic Mapping Process [29]

subject matters been explored? In addition, until now what kinds benefits have been highlighted? At the design stage, by utilizing various modeling illustrations, MDWE can be supported in different ways. Which modelling illustration has constructed MDWE forms, the demarcation of our question. The probability of recognizing complementary research requirements would be the solution to this question. Besides, based explicitly explicitly on the kind of contributions, this question is meant to observe how far the-seapproaches provide for the overall goals at present.

- RQ2: To publish research on MDWE, which methods are normally utilized? Early analysis revealed that web engineering was the subject matter of certain meetings devoted to specificities and international journals whilst MDWE was a workshop topic. By our intention to observe through the question demarcation, we seek other forums that are utilized to publish the investigation in this field.
- RQ3: What diverse kinds of investigation in this literature has been highlighted and how far has it gone? As explained in SWE-BOK and MDWE workshop guides [30, 31], to heighten the integrity of the investigation, the utilization of empirical studies and enhanced proven approaches is encouraged [32]. In this perspective, with regards to the particular scope of MDWE, we want to categorize various research types available.

### 3.2. Search Strategy

With the purpose of ascertaining the largest number of significant chief studies, we created a definite pursuit approach. We label it from

three viewpoints: search scope, search method, and search strings utilized.

As far as the scope is concerned, to identify the highest quantity of the associated investigative tasks, we did not limit the scope of our search to any specific research locations. But, the investigative outcomes are narrowed to publications dated between January 2000 and January 2014. We selected this commencement date because the highest publication regarding this area commenced post-end 1999. Conversely, the search scope for manual search (highlighted below) is restricted to the periods indicated for each location as follows.

In view of search techniques, manual as well as automatic searches were carried out. The search carried out by manually going through journals or meeting events is our idea of a manual search. At the same time, through the amalgamation of pre-demarcated search strings to locate the prime electronic dates is an automatic search. As the manual search for certain journals and meeting events published on those areas was forecasted to be immensely time consuming, we carried out automatic search for the bulk of locations.

Based on Table 2, we chose a number of journals and meetings for the manual tasking majority of the studies were MDWE, discovered there during preliminary investigative searches. We utilized the search string highlighted in Table 1 for the automatic searches, being the former which is characteristic of four rudimentary ideas connected to MDWE. By conducting a number of initial searches on chosen electronic data sources, the concluding string was created. ACM Digital Library, IEEEEXplorer, Science Direct, Springer Link, Scopus, Engineering Village, ProQuest, and Google scholar, as per Table 3,

Table 1. Search String Used for Automatic Searches

Concept	Alternative Used
Model Driven Web Engineering	(model driven OR model-driven OR model-driven development OR MDD OR MD OR Modeling OR meta model OR meta-model OR model transformation) AND (web engineering OR web engineering methods OR web based OR web application)

Table 2. Overview of publication forums for selected studies

Sources	Name	No.
Journals	Journal of Web Engineering	7
	International Journal of Web Engineering and Technology	3
	International Journal of Information Technology and Web Engineering	2
	ACM Transactions on Internet Technology journal	3
	international journal of web information system	5
	Global Journal on Technology	1
Conferences	International conferences web engineering	58
	International Conference web information system engineering	8
	International Conference On Web Information Systems And Technologies (WEBIST)	14
	International World Wide Web Conferences	14
	International Conference Model-Driven Engineering Languages and Systems	3
	The Unified Modelling Language Conference	8
	Proceedings edition of the Educators' Symposium	3
	International Conference on Information Integration and Web-based Applications Services	14
	International Conference on Software and Data Technologies	7
	Hawaii International Conference on System Sciences (HICSS)	5
	Symposium on User Interface Software and Technology (UIST)	1
	IEEE International Symposium on Web Systems Evolution	4
International Journal of Computer Information Systems and Industrial Management Applications	1	
Workshops	Model Driven Web engineering workshop	69
	international workshop Model-Driven Security	1

were the primary digital sources that were utilized to carry out automatic searches.

The string provided in Table 1, utilized to structure an accordingly equivalent string explicit to each source based on the point that since the tools furnished by different sources, including the precise syntax of search strings to be used differ between each source. For the application of the search string for safeguarding uniformity, a duplicate set of metadata values (i.e. title, abstract and keywords) covering all sources was chosen.

### 3.3. Selection of Primary Studies

As mentioned earlier, we utilized an amalgamation of manual and automatic searches. The sys-

tem of choosing chief studies is highlighted in Figure 2. To ascertain a preliminary set of publications, we started by conducting a number of investigative searches on digital libraries provided earlier. In addition, we utilized six previously known papers [11, 21, 234, 258, 289] as the initial point and according to the references and citing publications. As a result, this step produced 14 publications [1, 4, 15, 34, 35, 38, 46, 56, 67, 71, 139, 204, 245, 253]. To aid us in ascertaining certain journals and meeting events pertinent to our study; we utilized this preliminary set of publications. Hence, since they were acknowledged to be famous among web engineering investigators and publications associated with our study and probably were to be located

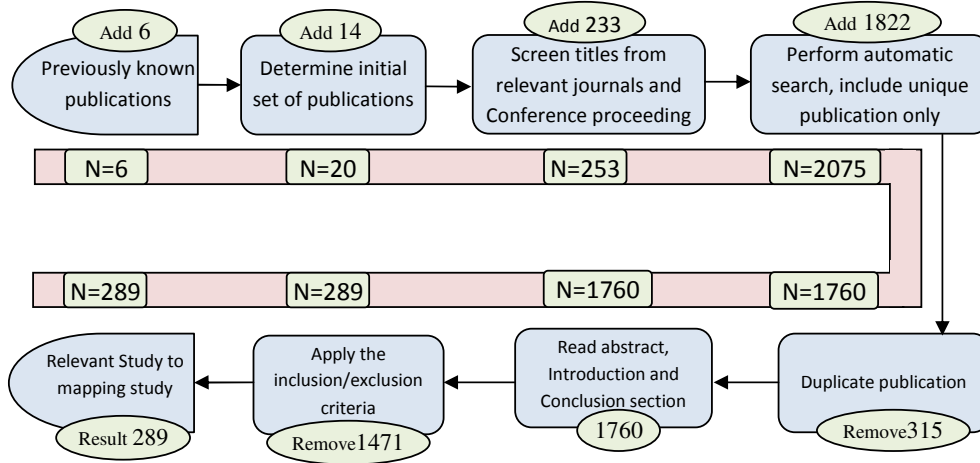


Figure 2. Study Selection Process

there as highlighted in Table 2, we made up our minds to manually search for transactions on Model Driven in Web Development, events of the annual conference models and metamodels, events of the transformation model conference and MDWE workshop. We acquired additional significant studies by screening titles in these areas, and the overall number of studies was 233. For the purposes of obtaining a general view of the area and to demarcate initial classification plans, these publications were screened.

We discovered 253 publications, six from previous known papers that very relevant papers in this area also any person can find it easily, 14 from references of six papers, 233 from journals, conferences and workshops through the manual step in total as shown in Figure 2.

Utilizing the search engines of electronic data sources i.e. IEEEExplore, Science Direct, ACM Digital Library, and Springer Link, we conducted automatic searches in the following phase. The search string provided in Table 1 was utilized by us. Table 4 represents a general view of outcomes taken from the manual and automatic searches. In addition, we performed the search string to Google Scholar. As a result, as shown in Table 3, we acquired additional significant studies, and the overall number of studies is 1822.

Eventually, we discovered 2075 papers: 253 from manuals, 1822 from the automatic search after merging manual search and automatic search.

Table 3. Digital Libraries Used in Automatic Search

Library	No.
ACM Digital Library	77
IEEE Xplorer	646
Science Direct	72
Springer Link	347
Scopus	115
Engineering Village	214
Google scholar	120
ProQuest	231
Total	1822

After conducting manual and automatic searches, we did not include the duplicate publications. By matching results acquired in this step, we discovered 315 papers were duplicated. Hence, the remaining papers total 1760.

To resolve about its inclusion or exclusion, the authors took into account the Abstract, Keywords, Introduction and Conclusion of each of these 1760 studies acknowledged to this stage, for the second time. Because of their shortfall in significance or fulfilling one of the other exclusion conditions, a total of 1471 studies were not included either. Based on our selection criteria, which are utilized for the mapping study, we discovered that the the remaining number of papers that were ready for systematic mapping is 289 papers. A general view of outcomes acquired from manual and automatic searches is presented in Table 4.

Table 4. Presents Overview of Results Obtained from Manual and Automatic Searches

Sources	Study re- trieved	Duplicate	Exclusion	Inclusion	Ready to mapping
<b>Manual Search:</b>					
Previously known publications	6				
Determine initial set of publications	14				
Journals, Conferences and Workshops	233				
<b>Online Search:</b>					
ACM Digital Library	77				
IEEE Xplorer	646	315	1471	289	289
Science Direct	72				
Springer Link	347				
Scopus Link	115				
Engineering Village Link	214				
Google scholar Link	118				
ProQuest Link	231				
Total	2075	1760	289	289	289

A listing of all criteria on the foundation of which studies were included or excluded is given below.

- Inclusion: We highlight some points to inclusion of the papers that answer our research questions.
  - Studies that clearly present an MDWE, demarcating new structures into UML or by utilizing its extension mechanisms.
  - Papers that demonstrate a distinctive answer to certain metamodeling or model transformation problem, or MDD, or MDA.
  - Papers that create a current MDWE in practice and assess it.
  - Studies that suggest methods to mapping MDWE.
  - Studies that merged the model driven in web application's scope.
  - Papers that suggest rudimentary outlines such as typical case studies for demonstration or substantiation of MDWE.
- Exclusion: We highlight some points to exclude the papers that do not answer our research questions.
  - Based on abstract, papers which mentioned MDWE. This was needed because in spite of the studies indicating MDWE in their introductory sentences as a chief concept, we found that these studies fell short of highlighting it. Other concepts

such as MDD, MDA and MDSD were also subjected to the same criterion.

- Papers that address only recommendations, guidelines or principles, rather than highlighting a useful approach to MDWE.
- Initial papers for books.
- Editorials, keynotes, tutorial outlines, tool demonstrations and panel deliberations, books, technical reports and other non-peer-reviewed publications.
- Identical reports of the same study discovered in various sources.
- Papers from industrial meetings, posters, and non-English publications.
- Papers unable to solve our research questions.

A general view of studies acquired by way of manual and automatic searches is presented in Table 4. The number of studies that were chosen in accordance to the inclusion criteria highlighted in Figure 2 is shown as well.

### 3.4. Defining a Classification Scheme

The classification schemes suggested by Petersen et al. [29] were utilized by us (Fig. 3), and we classified the publications into categories from three viewpoints: (1) focus area, (2) type of contribution and (3) research type. But, these categories were altered to match the details

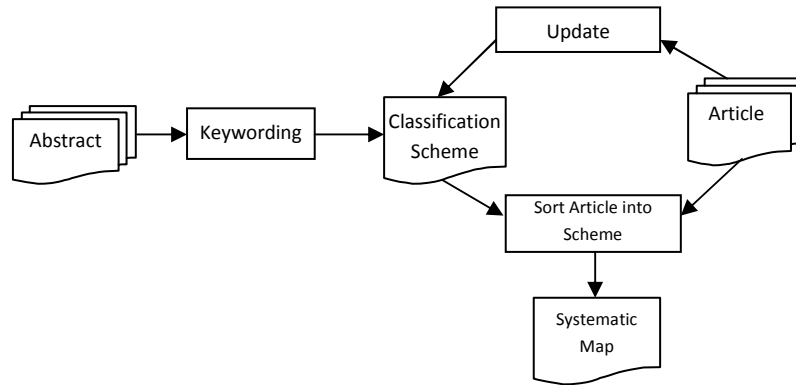


Figure 3. Building Classification Scheme [29]

of our mapping study. We utilized an iterative strategy while categorizing and mapping the studies into classification schemes. The concluded classification schemes are shown in Section 4.

Techniques to lessen the time required in creating a classification scheme and making sure that the scheme takes the current studies into consideration is key wording. Key wording is completed in two steps. At the beginning, the reviewers go through abridgements and search for keywords and ideas that showcase the input paper. In the process, the reviewer confirms the framework of the research. Following this, a comprehensive grasp about the nature and input into research is created through a set of keywords from various papers merged together. This aids the reviewers in demarcating a set of categories that is characteristic of the core population. In addition, reviewers can choose to study the opening or closing segments of the paper when abridgments are found to be of terrible quality to permit important keywords to be selected. When an absolute set of keywords has been selected, they can be gathered and utilized to create the categories for the map [29].

### 3.5. Mapping of Studies

As demarcated in Section 4, the real mapping was undertaken by mapping each involved study to a specific intersection set in the classification schemes. Section 5 shows the resultant mapping.

## 4. Classification Schemes

As deliberated earlier in Section 3, publications are categorized from three diverse approaches: focus scope, contribution and research type as shown in Figures 4, 7 and 8.

### 4.1. Focus Area

Chosen studies were separated into five research focus scopes based on specific research subjects, they addressed based on a broader outlook. Identifying these research focus areas was achieved through the key wording method shown in [29]. The eight categories of research focus areas are concisely described below and as well as in Figure 4.

**Web Applicability:** This category includes studies that present software applications that run in a web browser and Rich Internet applications (RIA). Furthermore it presents articles when related to the Web Information System (WIS), Search engine, Semantic web and cloud application. Furthermore, represent any articles that are related to MDWE with web applications.

**Testing and Quality:** This category reflects papers that present web system qualities, such as QoS, testing web software and web security. It also shows the papers that are related to the quality of websites.

**Service and Oriented:** One of the most popular fields in web software is web service. This category includes studies that present web services with Model-driven web services or partially re-

lated to MDWE ones, such as web services with UML, Metamodel, and workflow in the web domain. It also represents studies related to Service Oriented Architecture (SOA) with Model Driven in the web engineering.

**Requirements and Design:** Requirements and design are the software engineering steps; this category presents studies that are related to UML design and some steps in the design process; also, report studies that are related to functional and non-functional requirements.

**Web Economics:** This Category presents studies of software economics; moreover it includes articles focused on e-commerce, e-business, social web and social mashup websites.

**Modeling and Notation:** This category includes studies that present a modeling and a notation on its own, or in some way, contribute to the modeling process which uses some existing notation. This category reflects papers in the fields: Metamodels (presentation model, navigation model and user interface), model transformation (CIM, PIM models), code generation and adaptively, or other studies with the same concept as MDWE.

**Methodologies and Development Process:** While some studies focused on the methodologies or web development process, this category reflects papers that study the web engineering methodologies or the concepts of methodologies. On the other hand, it represents papers that focus on the web development process such a business process or an agile process .

**Web Management:** Studies present a novel method of weaving models, or present some solutions related to management of the model for websites. In the future, they will include more papers that work on Content Management System (CMS) or data management in the websites.

The following figure (Fig. 4) shows the topics of focus areas for MDWE with the percentage value of each of them . In this classification of finding topics, we use the SWEBOK guideline and the guide call paper at the workshop of the model driven web engineering [30,31].

Figure 4 shows the classification 289 papers of MDWE for eight topics of research focus; we found most of the papers in Web Applicability

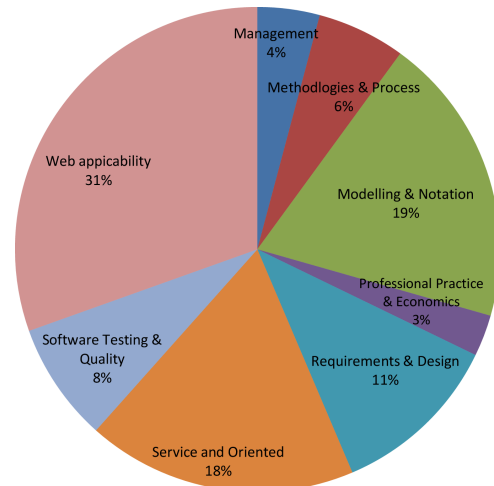


Figure 4. Distribution of Research Focus

(31%), followed by modeling and notation (19%) and service & oriented (18%). However, some categories were very important in software design, but we could not find more of them, such as: Requirements & Design (11%), Testing & quality (8%) whereas development processes covered only (6%) and some categories have few publications, such as Web management (4%), and Web Economics (3%). However, we classified our research focus on eight topics, but it was not easy to select the research focus because the eight topics were very general; so we classified each topic into several subtopics by using SWEBOK and MDWE workshop guides [30, 31], as shown in Figure 5.

Figure 5 classifies 8 topics of research focus into 26 subtopics: (1) Web applicability subtopics (Web Application, RIA, Semantic Web, WIS, Search Engine, and Cloud Application), (2) Testing & Quality has three subtopics (Security, QoS, Testing), (3) Service & Oriented that has only two subtopics (Web Service and SOA), (4) Requirements & Design subtopics are (Functional & non-functional requirements, UML & Design), (5) Web Economics subtopics (business, social web, evolution) (6) Modeling & Notation subtopics (Model transformation, metamodel, adaptivity, code generation), (7) Methodologies & Development Process subtopics (Methodology, agile, Development Process), (8) Management sub topics (CMS, Weaving, data-intensive). Figure 6 shows the 26 sub

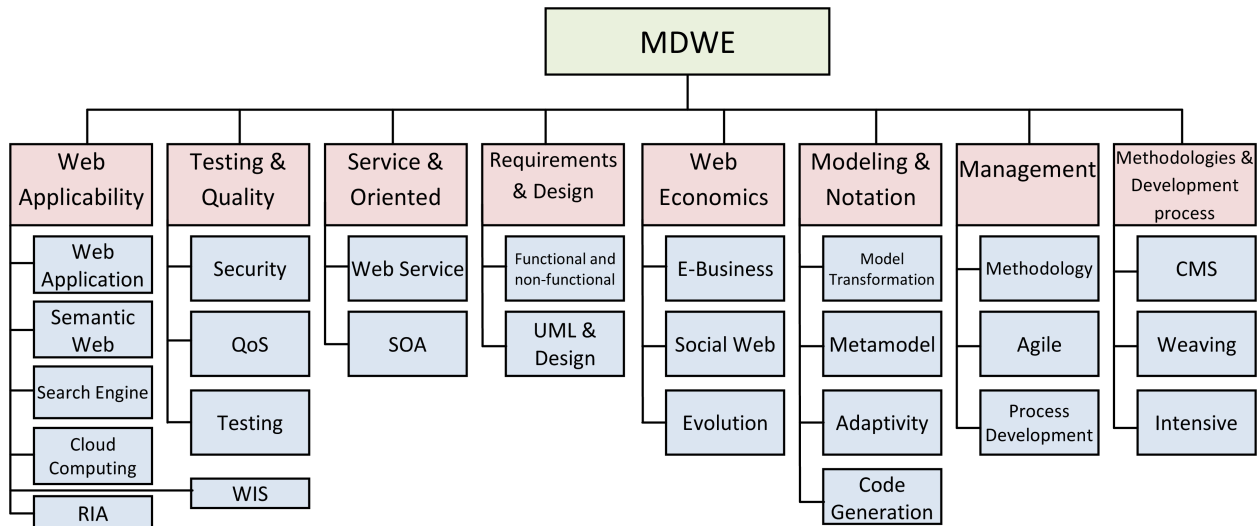


Figure 5. Classification of the Research Related to MDWE

topics with a number of publications and references.

Figure 6 explains the number of publications per subtopics. The figure shows the majority of publications in Web Applications (15.9%), Web Services (13.5%), Model Transformations (8.7%), minor publications in Cloud Application (0.3%), Evolution (0.3%), and Data-insensitive (0.3%). Other subtopics (between 0.7% to 6.6%) on the other side of this figure represent the reference of publications for e.g. RIA has 19 publications where the references are [77–95], CMS has 9 publications where the references are [304–312], but Cloud Application has only one publication where the reference is [118], and so on.

#### 4.2. Contribution Type

The contribution type is divided into five categories (see Fig. 7) described below:

**Metric:** The suggestion or application of metrics to effectiveness of MDWE is emphasized through this contribution.

**Tool:** In the design of a prototype or a device which can be assimilated with current outlines is based on contributions that target on supplying tool support for MDWE.

**Method:** Modeling, approaches, model changes and model structure, which are provided explicitly through contributions.

**Model** Based on papers that theoretically deliberate or create contrasts, investigate associations, seek challenges, or create classifications, etc.

**Process:** The papers contribute to the process which is characterized through papers that explain the MDWE and furnish a depiction on their assimilation in the general software development process. Furthermore, certain specific issues which are settled through these contributions are associated with MDWE.

Figure 7 shows major publications in the contribution type which are related to the Method (33%) which minor in Metric (2%), between minor and major there is Model (24%), Process (23%), and Tool (18%).

#### 4.3. Research Type

The research strategy utilized in the main study is reflected through research type. For the classification of research types (RQ3), we have utilized a scheme suggested by Wieringa et al. [32]. A concise depiction of research kinds are as follows (see Fig. 8):

**Evaluation Research:** Comparison with validation research, evaluation research focuses on analyzing the answer which has been essentially applied by now. It examines the practical application of the solution.

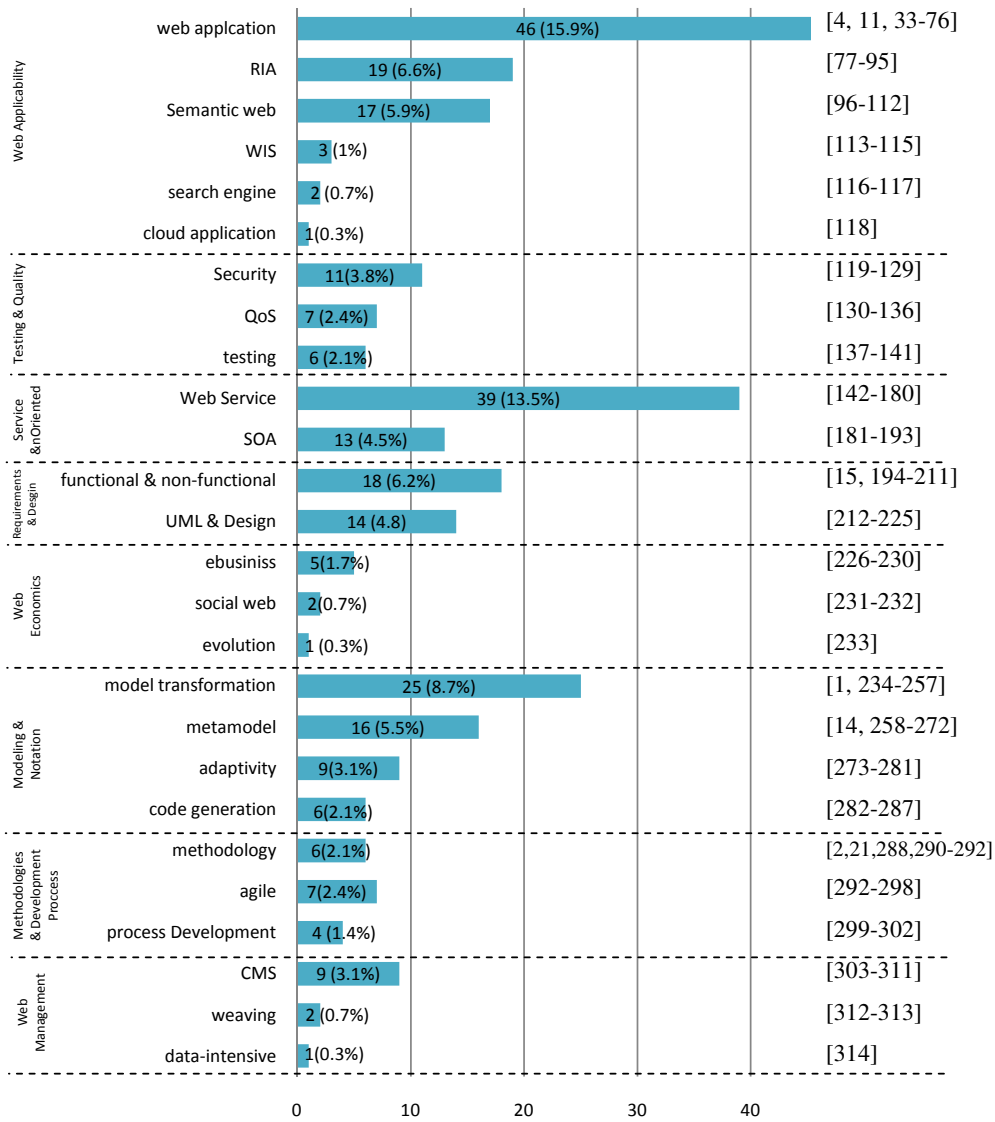


Figure 6. Number of Papers per Research Topic and References

**Experience Paper:** The personal experiences of the author from a single or more real life projects are reported through an experience paper. It normally explains what was achieved in the project and also how it was accomplished.

**Opinion Paper:** The author’s own ideas on the aptness or inaptness of a certain method or instrument are reported through these papers. Likewise, on the basis on explanations how certain methods or instruments should have been developed etc., these papers are sometimes used to share personal opinion.

**Philosophical Paper:** To observe things that are already present in a novel way through an ar-

rangement presented via theoretical suggestions. However, it does not accurately overcome a specific issue. Taxonomies, theoretical outlines, etc. will be maybe added to theoretical suggestions.

**Solution Proposal:** By providing either an innovative answer or a significant extension of an existing technique, a solution proposal overcomes a problem. In addition, its advantages are highlighted by either a case in point or in-depth reasoning.

**Validation Research:** The investigation of the solution proposal that has not been essentially put into use is the chief reason for validation research. By way of systematic manner, valida-



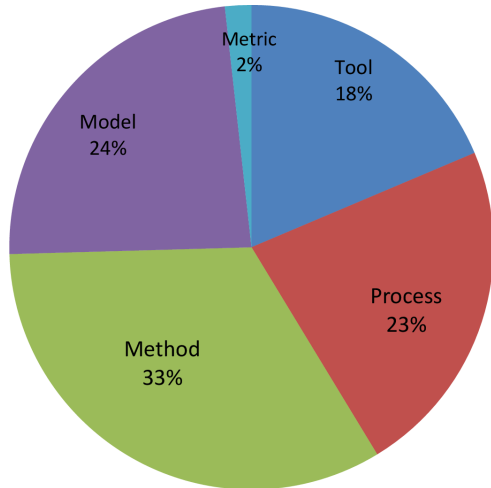


Figure 7. Distribution of Contribution Type

tion research is carried out and may pose any of these: experiments, prototypes, simulations, mathematical analysis, etc.

Chief and minor publications in the research type are Validation Research (24%) and Opinion Papers (7%) respectively; other publications are divided into Solution Proposal (20%), Philosophical Paper (19%), Experience Paper (17%), and Evaluation Research (13%).

#### 4.4. Scheme Mapping Study

In this study, we have 289 papers that are ready for systematic mapping, after the extraction of papers to form categories of the Research Focus (Fig. 4), Contribution Type (Fig. 7), and Research Type (Fig. 8); we designed a mapping study with a number of publications, as shown in Figure 9.

The Map (Fig. 9) shows the classification mapping study of 289 papers; these papers show the number of applications with a focus on research type and contribution type. We will discuss this in Section 5. For more information about our papers, we designed a bar chart of publications per year as shown in Figure 10.

Figure 10 shows 289 papers per year between 2000 and 2014; the result of a bar chart is the publication of continual MDWE growth. In 2000, only one paper was found but in 2013, there were 29 papers, with most publications between 2007 and 2013. However, the result for



Figure 8. Distribution of Research Type

2013 was such because probably, our search in January 2014 found some unpublished papers. Hence, these results show this area is a new and active area, which means that in the last decade the researchers focused on this area in publications.

### 5. Mapping and Discussion of Research Questions

With regards to research type and contribution type, a map covering eight current research target scopes within the setting of MDWE was created in order to provide an overview of the field (see Fig. 9). The framework of the focus of the current investigation, together with a suggestion of investigative divisions in the area, is provided on the map. Most of the research papers are particularly devoted to furnishing model driven development, and clarifying the related processes as shown through mapping outcomes. A higher degree of investigation has been undertaken regarding the structure of web engineering methods, model driven development and model driven architecture, within the scope of web development. However, we highlight our findings in two divergent dimensions to the extent to which analysis of MDWE subjects in current research is concerned: (1) main subjects in the area together with the magnitude of their coverage and contribution types (RQ1) and research type (RQ3),

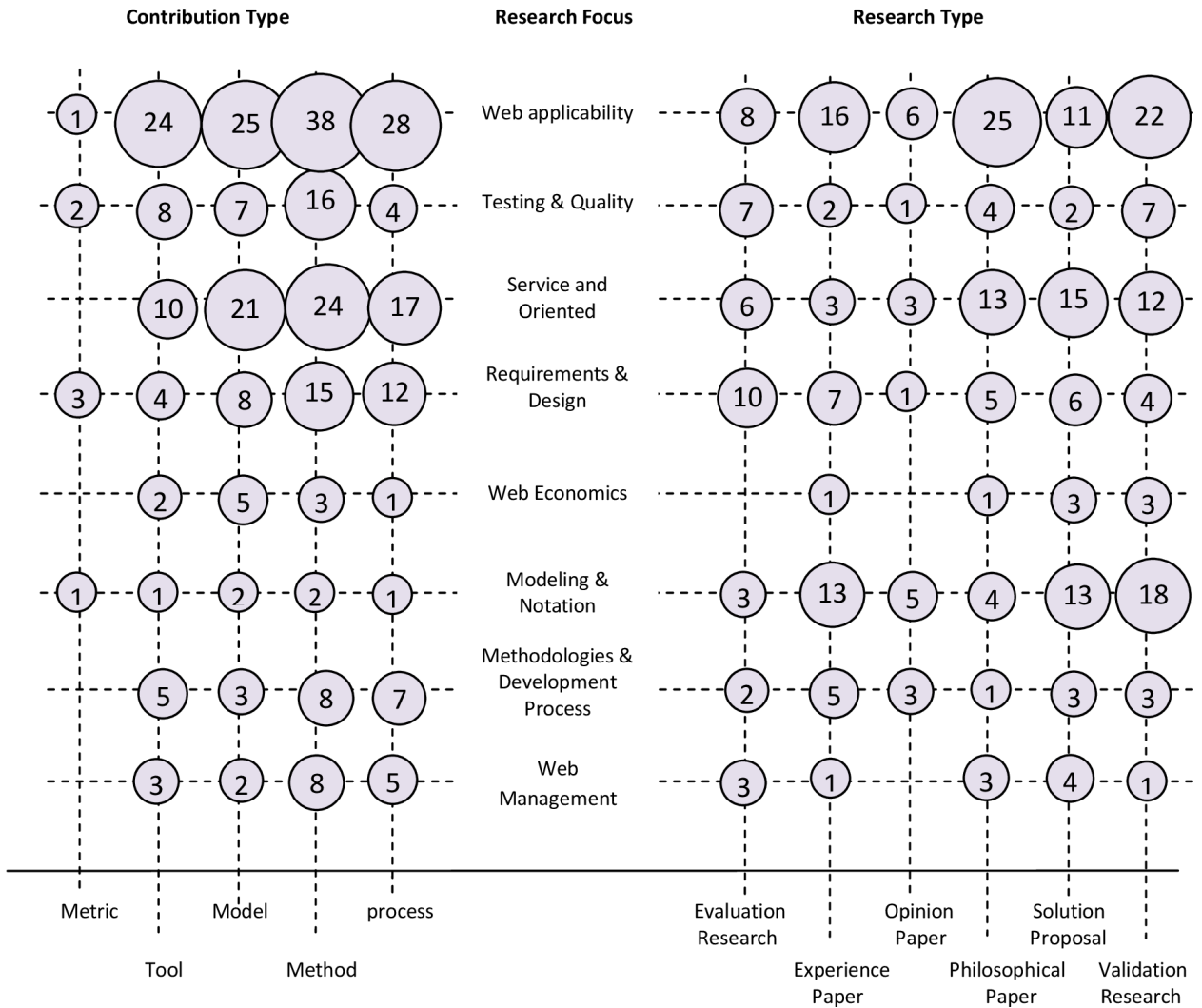


Figure 9. Map of Research Focus on MDWE

and (2) forums utilized for publishing the associated research (RQ2).

The first dimension of our results, including the major topics along with specifications of research types, has been covered in the Sections 5.1–5.8. We have organized each subsection in a way that briefly describes the studies selected for each topic, while highlighting the extent and nature of research. Furthermore, it identifies the types of contribution made by each selected study. The publications in this area can be divided into eight major focus areas (see Fig. 4), including Web Applicability, Service and Orientation, Modeling and Notation, Requirements and Design, Testing and quality, methodologies and process, management and Economics. Figure 5 also shows the major topics addressed

by the existing research, divided into related subtopics where possible. Figure 6 shows a summary of groups of papers identified per research subtopic.

An overview of the volume of research selected by major research focus areas is shown in Figure 4. It shows that most publications are covered by Web Applicability, at 31%, followed by modeling and notation at 19% and web services at 18%. Another level is software quality, which has a good coverage rate in the publications, but 11% of publications cover requirements and design, while 8% of publications cover software testing and quality, and 6% cover methodologies and processes. A very small number of publications cover management (4%) and economics (3%). Figure 7 shows the contribution

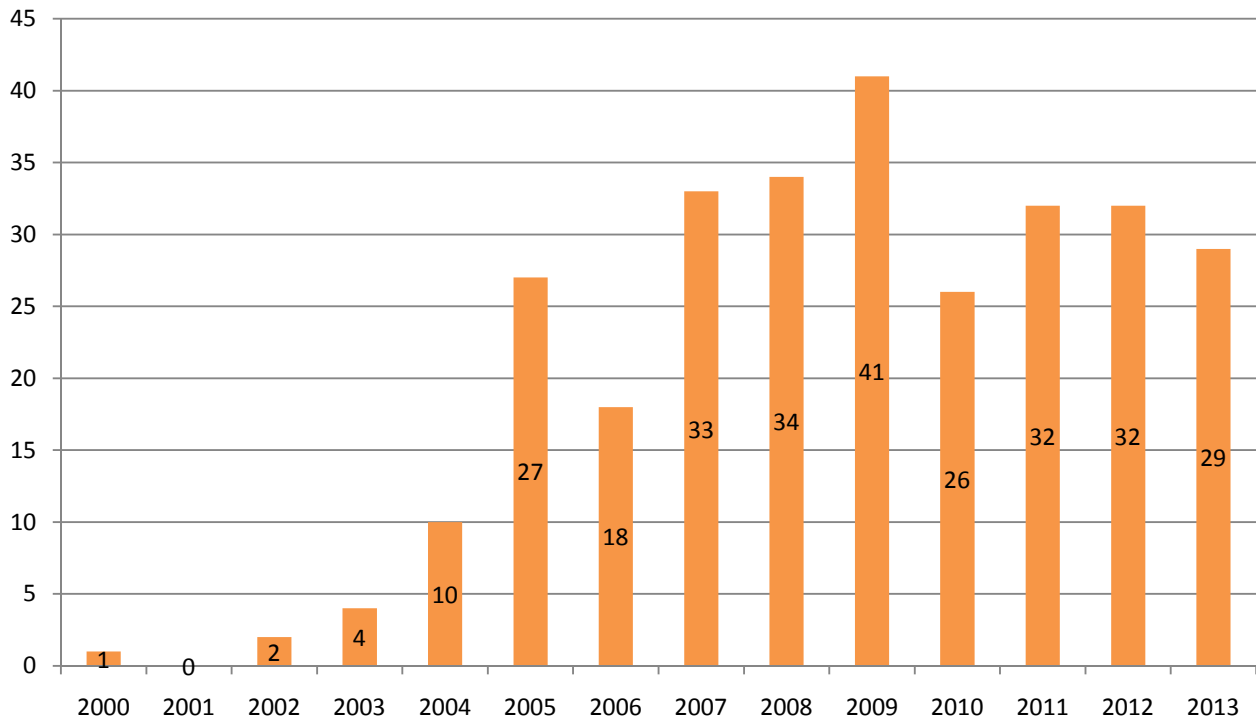


Figure 10. Publication per Years

Table 5. Research and contribution types presented by 88 papers on Web applicability

Contribution Type	Research Type				Metric
	Process	Method	Model	Tool	
Evaluation Research	[41, 48, 76]	[47, 48, 103, 117]	[47, 53, 85]	[103]	–
Experience Papers	[40, 55, 82]	[33, 40, 61, 66, 92, 100]	[11, 39, 57, 90, 92]	[33, 57, 59, 61, 71, 72, 105]	–
Opinion Papers	[93, 113]	[52, 83, 93]	[69]	[82]	–
Philosophical Paper	[4, 42, 43, 45, 46, 79, 80, 87–89, 99, 102, 106, 118]	[35, 45, 95, 96, 99, 110, 115]	[4, 56, 65, 80, 88, 89, 91, 98, 109, 115]	[4, 43, 56, 67]	[89]
Solution Proposal	[58, 75, 84, 97]	[36, 64, 78, 101]	[58, 64, 86]	[97, 104, 116]	–
Validation Research	[50, 73]	[34, 38, 44, 49, 50, 62, 63, 68, 70, 74, 108, 111, 112]	[60, 94, 107]	[37, 38, 51, 54, 63, 77, 107, 114]	–

type of publications, where 33% contribute to method, but the model, process and tool, have near percentages, which are 24%, 23% and 18% respectively. A small percentage of publications returned to metrics, specifically 2% of publications. Figure 8 shows that, based on research type, only 7% of publications reported opinions, but 24% reported validation research, 19% reported philosophical content, 17% reported real-life experiences, and 13% reported evaluation research. Furthermore, Figure 10 shows the bar chart of publications per year, but with most

publications released in 2009 starting with 2005 it is still an active field for publication.

### 5.1. Web Applicability

In this section, we briefly discuss different studies related to Web Applicability. Table 5 lists the papers that focus on this topic. This is an area where most research effort is spent. Also in this section we discuss the sub-topics which consist of web applications, Rich Internet appli-

cations, semantic web, search engine, and cloud computing.

In the MDWE web applications there is an application program that is stored on a remote server and delivered over the Internet through a browser interface that is driven by software engineering methods. With the publications' growth in this field, Cheung [37] developed a web application design framework through a tool and in [55] used a model driven process for the development of web applications.

Rich Internet applications (RIAs) offer rich, engaging experience that improves user satisfaction and increases productivity. Using the broad reach of the Internet, RIAs can be deployed across browsers and desktops. In [80], RIA was defined as a new approach and was developed through model driven architecture, while [86] presents a RIA metamodel to deal with the new technological challenges that have arisen with Web 2.0 development [86].

Another type of web application is the Semantic web that is represented in [96–112], web information system that is represented in [113–115] and search engine that is represented in [116, 117], while a new field is clouding, as shown in [118]. In this paper Kumar et.al. used the Model Driven Approach for Developing Cloud Application. This paper was published in 2013.

## 5.2. Testing and Quality

This category includes papers related to Model Driven, with software testing, quality of service and security. Escott, E. [141] focused on Model Driven in the development of testing web applications, Ortiz, G. in [134] presents a model-based approach to the implementation of QoS monitors, by describing them as platform-independent models. On the other hand, Nakamura, Y. [123] describes a tooling framework to generate Web services security configurations, using a model driven architecture (MDA) as shown in Table 6.

## 5.3. Services and Oriented

One of the most popular fields in MDWE are web services, usually with some combination of programming and data, and possible inclusion of human resources as well. Table 7 shows the papers related to web services and Service Oriented Architecture (SOA), for example, Achilleas A. et al. [151]. They propose a Model-Driven Web Service oriented framework that combines MDE with Web Services, to automate the development of platform-specific web-based applications. In another paper, Bajohr and Margaria [189] address the high availability of model-driven SOAs for applications that are orchestrations of services and are defined by their (behavioral) models.

## 5.4. Requirements and Design

This category includes papers that explain functional and non-functional requirements that support Model Driven in the web domain, and also papers that focus on the UML design in web domains. Table 8 classifies requirements and design publications. Aguilar et al. [197] prepared an algorithm that has been defined in order to analyze dependencies among functional and non-functional requirements, and Guzman et al. [222] showed Web 2.0 patterns requirements in MDWE.

## 5.5. Web Economics

Software engineering economics is about making decisions related to software engineering in a business context. The success of a software product, service and solution depends on good business management. Yet, in many companies and organizations, software business relationships to software development and engineering remain vague. Table 9 has all the publications that were founded on MDWE. Guotao and Du [227] implemented e-commerce on the web application.

Table 6. Research and Contribution Types Presented by 23 Papers on Software Testing and Quality

Contribution Type	Process	Method	Research Type		Tool	Metric
			Model			
Evaluation Research	[121, 135]	[121, 126, 130, 135]	[134, 136]		[126, 132, 134–136]	[135]
Experience Papers	–	[139]	–		[123]	–
Opinion Papers	–	[122]	–		–	–
Philosophical Paper	–	[128, 131, 140, 141]	[128, 131]		[131]	[141]
Solution Proposal	–	[120, 125]	[120]		–	–
Validation Research	[137, 138]	[127, 129, 133, 138]	[124, 137]		[119]	–

Table 7. Research and Contribution Types Presented by 52 Papers on Services and Oriented

Contribution Type	Process	Method	Research Type		Tool	Metric
			Model			
Evaluation Research	[144, 182, 187]	[153, 187]	[155, 181]		[144, 182]	–
Experience Papers	–	[160, 176, 189]	[189]		–	–
Opinion Papers	[167]	[171]	[156, 167]		–	–
Philosophical Paper	[152, 159, 165, 193]	[142, 159, 185, 188, 190, 193]	[161, 164, 165, 174, 180]		[147, 188]	–
	[145, 177, 178, 183, 186]	[146, 150, 157, 158, 169, 170, 172, 173, 175, 177]	[150, 170, 175, 178, 186, 191]		[145, 178]	–
Solution Proposal						
Validation Research	[148, 166, 184, 192]	[143, 149]	[163, 166, 168, 179, 192]		[18, 151, 154–162]	–

Table 8. Research and Contribution Types Presented by 33 Papers on Requirements and Design

Contribution Type	Process	Method	Research Type		Tool	Metric
			Model			
Evaluation Research	[195, 198, 200, 201, 212]	[195, 200, 206, 211]	[206, 211, 218]		–	[201, 209, 210]
Experience Papers	[194, 199, 208, 222]	[222, 225]	–		[202, 204, 208]	–
Opinion Papers	[197]	–	–		–	–
Philosophical Paper	–	[205, 216, 224]	[203, 220]		–	–
Solution Proposal	[213, 214]	[217, 219, 221]	[221, 223]		–	–
Validation Research	[207]	[15, 196, 215]	[207]		[196]	–

Table 9. Research and Contribution Types Presented by 8 Papers on Economics

Contribution Type	Process	Method	Research Type		Tool	Metric
			Model			
Evaluation Research	–	–	–		–	–
Experience Papers	–	[230]	–		–	–
Opinion Papers	–	–	–		–	–
Philosophical Paper	–	[232]	[232]		–	–
Solution Proposal	–	[231]	[226, 233]		[231]	–
Validation Research	[227]	–	[227, 229]		[228]	–

### 5.6. Modeling and Notations

In this section, we briefly discuss different studies related to modeling notations and the associated notations. Table 4 lists the papers that focus on this topic. This topic consists of metamodels, model transformations, adaptive and code generation. Jiang et al. [273] propose MAWA, a method for model-driven development of adaptive Web applications. Koch and Kraus [268] present a first step towards such a common metamodel by defining first a metamodel for the UML-based Web Engineering (UWE) approach. [235, 236, 257] are papers that focused on model transformation, but [282–287] are papers that focused on code generation in MDWE, as shown in Table 10.

### 5.7. Methodologies and Process

This topic includes papers related to web engineering methodologies and processing, a list of which can be seen in Table 11. Andrés and Duitama [21] present some web engineering methodologies. In Rivero et al. [296] proposed an agile approach to MDWE methodologies (called Mockup-Driven Development, or MockupDD) by inverting the development process. This can be seen in Table 11.

### 5.8. Web Management

The last topic under MDWE is management websites through different models. This topic covers papers that are related to CMS, weaving and data management in this area. Table 12 lists management papers in MDWE. Joao and Alberto in [306] proposed the creation of a model-driven approach for the development of web-applications, based on Content Management Systems.

## 6. Discussion

In this part, based on findings on future examination, we provide a summary of the legitimacy of threats, related to the crucial findings

of this systematic mapping study, and deliberate regarding certain consequences of these findings. We also highlight the limitations of this mapping study that may represent threats to its validity.

In this paper we propose a systematic mapping study for MDWE, the primary studies on MDWE to explore current work, and we identify needs for future research. A systematic mapping study is used for finding the most relevant studies and classification. In this study, we found 289 papers and classification schemes divided them into classification schemes on the basis of research focus, contribution type and research type. The majority of 20% of the papers were on the solution proposal type of research. The most common areas in MDWE appear to be: Web Applicability at 31%, Molding and Notation at 19%, and Services and Oriented at 18%. The majority of contributions are methods, at 33%. Moreover, this shows the MDWE as a wide, new, and active area for publications. Whilst additional analysis is warranted within the MDWE scope, in literature composition mechanisms have been thoroughly discussed. Furthermore, we have observed that a recurrent recommendation for validation research, solution proposals and philosophical papers has been presented through earlier analysis.

### 6.1. Threats to Legitimacy

The outcomes of a systematic mapping study may be affected by diverse factors, for example, the researchers who conducted the study, the databases and the search string developed, as well as the time limits chosen. As it will be shown in the following paragraphs, when these threats to legitimacy are taken into account, the outcomes become more satisfactory and precise.

We conducted a systematic mapping study and every stage was explicitly defined. The other investigators were permitted to reprise the mapping study, since each step was shown explicitly. However, it is probable that certain articles that were omitted would be counted in, and vice versa, as a result of choosing articles which have been conducted by diverse investigators, because the decision about the exclusion or inclusion of

Table 10. Research and Contribution Types Presented by 56 Papers on Modeling and Notations

Contribution Type	Process	Research Type		Tool	Metric
		Method	Model		
Evaluation Research	[259]	–	–	[259, 275]	[238]
Experience Papers	[14, 239, 244, 285]	[254, 260, 287]	[244, 255, 266, 284]	[14, 243, 249, 254, 256, 260, 287]	–
Opinion Papers	–	[241, 277]	[1, 267, 270]	[270]	–
Philosophical Paper	[236, 257]	[250, 252, 257]	[250, 252]	–	–
Solution Proposal	[273]	[248, 258, 268, 269, 276]	[235, 247, 268, 269, 278, 279, 281, 286]	[263, 279]	–
Validation Research	[234, 237, 246, 265, 274, 280, 282, 283]	[240, 242, 251, 253, 262, 271, 272]	[245, 246, 251, 264, 271, 280]	[240, 253, 261, 262, 264, 282]	–

Table 11. Research and Contribution Types Presented by 17 Papers on Methodologies and Process

Contribution Type	Process	Research Type		Tool	Metric
		Method	Model		
Evaluation Research	[2]	[289]	–	[289]	–
Experience Papers	[290, 294]	[297]	[290]	[291, 294, 302]	–
Opinion Papers	–	[21, 293, 300]	–	–	–
Philosophical Paper	[292]	[292]	–	–	–
Solution Proposal	[295, 296]	[296]	[299]	–	–
Validation Research	[298]	[301]	[301]	[287]	–

Table 12. Research and Contribution Types Presented by 12 Papers on Management

Contribution Type	Process	Research Type		Tool	Metric
		Method	Model		
Evaluation Research	[314]	[310]	–	[309]	–
Experience Papers	[311]	–	–	[311]	–
Opinion Papers	–	–	–	–	–
Philosophical Paper	[306]	[303, 306, 313]	[313]	–	–
Solution Proposal	[304, 312]	[304, 305, 312]	[307]	–	–
Validation Research	–	[308]	–	[308]	–

a specific article is based on the investigators who conducted the mapping study. Yet, it is highly improbable that the main conclusions derived from the recognized set of articles would be altered by these diversities, based on personal assessments, which is a general categorization of approaches.

Acquiring a set of significant articles encompassing the said research subject was the target of the conducted mapping study. The outcome set should be completed as soon as possible. Based on this motive, we derived the search string in a systematic fashion. Because of the number of significant articles discovered utilizing a search string, not all appropriate words are used whilst creating a search string. For instance, the word 'Model-Driven' was added in the search string, and 'Web Engineering' was not included in the papers which have model-driven in software engineering only, including model-driven in web engineering. Moreover, some terms, for instance 'Web Application', are used incongruously in literature. At times, MDWE is not the only factor in 'Web Application'. In conclusion, a diverse set of final articles might have been the outcome of diverse or added terms utilized in the search string, but this would only pose a negligible effect on the general classification obtained, and added articles could be easily categorized based on the given classification.

## 6.2. General Findings

MDWE, as the main target in modern software development, is endorsed by results of the systematic mapping study, as there are many publications on this subject. As was proven by the number of new publications, the topic has received greater attention in recent years.

The advent of several suggestions, as a result of recent suggestions in the sphere of MDWE has focused mainly on the development of web applications. Nevertheless, there is still a relevant task to be undertaken if we look at the overall issues related to amalgamating MDWE into an MDE setting. Models are the main aim for envisaging an operable outlook of the system

in an MDE framework, and essentially acquiring working software systemically, in an automated manner. Therefore, the scope of modeling precise and comprehensive behaviors of factors requires more care, along with resolutions to identification models so as to amalgamate factors into this broader context. Up till now, minimal tasks have been described in the literature (e.g. [21]) that have highlighted methodologies substantiation, but even these substantiation methods pose a restricted infrastructure to substantiating methodologies, through execution only. Notwithstanding the fact that a systematic substantiation system cannot be replaced via verification undertaken in this manner, it can cause other issues. For example, it necessitates designers to be aware of the exact details of advice transformations, hence leading to usability issues.

## 6.3. Limitations of Review

It has to be mentioned that this review has certain restrictions. These restrictions are comparable to those of other systematic reviews. There is some probability that certain significant materials were not added to the review, for example dissertations, related books or white papers, and some significant papers might not have been discovered in the digital databases, by means of our search and selection protocol. The latter is more of an issue regarding how investigators write their abridgments, and how digital databases categorize and locate published work. The former is a restriction of our review, and could be highlighted the following works. There is actually no reason why a keyword search would not return the entire published significant material, if abridgments were prudently written and keywords were inserted. Sometimes, categorization schemes in the literature are already present, which can preferably be used again or enhanced. Nevertheless this seldom happens, and worst still, a sound categorization scheme may often not be the case. On top of that, for the currently published material, a thoroughly planned categorization scheme might not be the best option. Slowly developing the categorization scheme when running through the abridg-



ments of all the papers was the approach taken in this review. One direct issue with this strategy is that it might fall short of discovering some breaks in the field. For instance, in a certain categorization scheme there could be a missing category. Had the category been inserted, the significant breaks would clearly be noticed. Investigators are encouraged to be very conversant with the scope under assessment, creating the categorization schemes that diminish the threat of legitimacy from this restriction.

## 7. Conclusion and Future Work

A relevant progression in the development of web software systems that are more maintainable, extensible and reusable is an outcome of the investigation in the area of MDWE. We demarcated some research questions and launched a systematic mapping study, in order to acquire an overall view of the present investigation in this field. To satisfy the goals of the study, we discovered 289 publications that retained highest significance.

The chosen papers appeared between 2000 and 2014. The findings of this study show that MDWE is a somewhat underdeveloped area. In 2001, the preliminary relevant contributions to this area were shown (i.e. [63]). Most papers come out in workshops and meetings, while some have come out in journals.

As far as the answer to our first research question is concerned, the main research topics identified are: (1) Web Applicability, (2) Service and Oriented, (3) Modeling and Notation, (4) Requirements and Design, (5) Testing and quality, (6) methodologies and process, (7) management, and (8) Economics of MDWE. To respond to our second research question, we have determined that most research has appeared at conferences (63%) and workshops (32%). Relatively fewer publications (9%) have appeared in journals so far. As far as an answer to our third question is concerned, most of the research (24%) is validation research, while 7% are opinion papers, 20% of publications focus on solution proposal, and 19% of papers are philosophical. 17% of the

papers are experience papers, and 13% are evaluation papers.

Finally, the result shows that MDWE is a wide, new and active area for publication. Also, some fields need to be improved, and this is a good area for publication. This paper helps the web engineering researcher to find weaknesses and strengths in this area, and to understand which point or which side of this area needs to be enhanced. With regards to future work based on resulting maps in systematic mapping, researchers can make systematic mapping one of the research focuses, for example modeling and notation in MDWE, web management, web applicability, requirements, designs and web services in MDWE. Furthermore, the researcher can utilize the subtopics, including Semantic Web in MDWE, CMS, social web, and SOA in MDWE. These can be potential areas for future work. Furthermore, some ideas in the web domain have not appeared, or there can be articles not yet published, such as those related to Crawling in MDWE. Also researchers can look for new web domains to be added to MDWE. There is also a need for better empirical research, like the use of application/validation methods used for evaluation and validation research. Solutions proposed within the solution proposal need to be empirically validated, in order to strengthen the empirical research. Furthermore, researchers can use another method for classification and evaluation in order to find the best result, for instance for heuristic evaluations.

## References

- [1] A. Kraus, A. Knapp, and N. Koch, "Model-driven generation of web applications in UWE," *3rd International Workshop on Model Driven Web Engineering (MDWE07, CEUR-WS)*, Vol. 261, 2007.
- [2] G. Aragón, M.J. Escalona, M. Lang, and J.R. Hilara, "An analysis of model-driven web engineering methodologies," *International Journal of Innovative Computing, Information and Control*, 2013.
- [3] G. Aragón, M. Escalona, J.R. Hilara, L. Fernandez-Sanz, and S. Misra, "Applying model-driven paradigm for the improvement

- of web requirement validation,” *Acta Polytechnica Hungarica*, Vol. 9, No. 6, 2012, pp. 211–232.
- [4] A. Kraus, “Model driven software engineering for web applications,” Ph.D. dissertation, München, Germany, 2007.
- [5] M.J. Escalona, J.J. Gutierrez, M. Perez-Perez, A. Molina, E. Dominguez-Mayo, and F. Dominguez-Mayo, “Measuring the quality of model-driven projects with NDT-Quality,” in *Information Systems Development*. Springer, 2011, pp. 307–317.
- [6] F. Garzotto, P. Paolini, and D. Schwabe, “HDM-a model-based approach to hypertext application design,” *ACM Transactions on Information Systems (TOIS)*, Vol. 11, No. 1, 1993, pp. 1–26.
- [7] G. Rossi, O. Pastor, D. Schwabe, and L. Olsina, *Web engineering: modelling and implementing web applications*. Springer Science & Business Media, 2007.
- [8] M. Lang and C. Barry, “A survey of multimedia and web development techniques and methodology usage,” *IEEE Multimedia, Special Issue on Web Engineering*, 2001, pp. 52–60.
- [9] M.J. Escalona and N. Koch, “Requirements engineering for web applications—a comparative study,” *J. Web Eng.*, Vol. 2, No. 3, 2004, pp. 193–212.
- [10] M. Escalona, J. Torres, M. Mejías, J. Gutiérrez, and D. Villadiego, “The treatment of navigation in web engineering,” *Advances in Engineering Software*, Vol. 38, No. 4, 2007, pp. 267–282.
- [11] W. Schwinger, W. Retschitzegger, A. Schauerhuber, G. Kappel, M. Wimmer, B. Proll, C.C. Castro, S. Casteleyn, O.D. Troyer, and P. Fraternali, “A survey on web modeling approaches for ubiquitous web applications,” *International Journal of Web Information Systems*, Vol. 4, No. 3, 2008, pp. 234–305.
- [12] S. Murugesan, Y. Deshpande, S. Hansen, and A. Ginige, “Web engineering: A new discipline for development of web-based systems,” in *Web Engineering*. Springer, 2001, pp. 3–13.
- [13] M. Lang, “Hypermedia systems development: Do we really need new methods?” in *Proceedings of the Informing Science+ IT Education Conference, Cork, Ireland*. Citeseer, 2002, pp. 883–891.
- [14] N. Koch, S. Meliá-Beigbeder, N. Moreno-Vergara, V. Pelechano-Ferragud, F. Sánchez-Figueroa, and J. Vara-Mesa, “Model-driven web engineering,” *Upgrade-Novática Journal (English and Spanish), Council of European Professional Informatics Societies (CEPIS) IX*, Vol. 2, 2008, pp. 40–45.
- [15] N. Moreno, J.R. Romero, and A. Vallecillo, “An overview of model-driven web engineering and the MDA,” in *Web Engineering: Modelling and Implementing Web Applications*. Springer, 2008, pp. 353–382.
- [16] B. Kitchenham and S. Charters, “Guidelines for performing systematic literature reviews in software engineering,” EBSE, Tech. Rep. EBSE-2007-01, 2007.
- [17] J. Biolchini, P.G. Mian, A.C.N. Cruz, and G.H. Travassos, “Systematic review in software engineering,” *System Engineering and Computer Science Department COPPE/UFRJ, Technical Report ES*, Vol. 679, No. 05, 2005, p. 45.
- [18] Journal of web engineering. Rinton Press. (2014, May). [Online]. <http://www.rintonpress.com/journals/jwe/index.html>
- [19] International journal of web engineering. Scientific & Academic Publishing. (2014, May). [Online]. <http://www.sapub.org/journal/aimsandscope.aspx?journalid=1095>
- [20] International conference on web engineering. Toulouse, France. (2014, May). [Online]. <http://icwe2014.webengineering.org/>
- [21] H. Londoño, J. Andrés, and J.F. Duitama, “Model-driven web engineering methods: a literature review,” *Revista Facultad de Ingeniería Universidad de Antioquia*, No. 63, 2012, pp. 69–81.
- [22] E. Mendes, “A systematic review of web engineering research,” in *International Symposium on Empirical Software Engineering*. IEEE, 2005, p. 10 pp.
- [23] B. Kitchenham, “Procedures for performing systematic reviews,” Keele, UK, Keele University, Joint Technical Report, 2004.
- [24] J.A. Aguilar, I. Garrigos, and J.N. Mazon, “Requirements in web engineering: A systematic literature review,” *Journal of Web Engineering*, 2003.
- [25] E. Insfran and A. Fernandez, “A systematic review of usability evaluation in web development,” in *Web Information Systems Engineering WISE 2008 Workshops*. Springer, 2008, pp. 81–91.
- [26] B.P. Lamanha, M.P. Usaola, and M.P. Velthuis, “Software product line testing,” *A Systematic Review. ICISOFT (1)*, 2009, pp. 23–30.
- [27] S. bin Abid, “Resolving traceability issues of product derivation for software product

- lines,” *International Conference on Software and Data Technologies*, 2009.
- [28] T. Dybå, B. Kitchenham, M. Jorgensen *et al.*, “Evidence-based software engineering for practitioners,” *IEEE Software*, Vol. 22, No. 1, 2005, pp. 58–65.
- [29] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, “Systematic mapping studies in software engineering,” in *12th International Conference on Evaluation and Assessment in Software Engineering*, Vol. 17, 2008, p. 1.
- [30] P. Bourque and R.E. Fairley, *Guide to the software engineering body of knowledge-SWEBOK*. IEEE Press, 2014.
- [31] D. Pfahl. Software engineering. (2013, Dec). [Online]. [https://courses.cs.ut.ee/MTAT.03.047/2013\\_fall/uploads/Main/SoftwareEngineering.pdf](https://courses.cs.ut.ee/MTAT.03.047/2013_fall/uploads/Main/SoftwareEngineering.pdf)
- [32] F.P. Brooks, *The mythical man-month: Essays on Software Engineering*. Addison-Wesley Reading, MA, 1995.
- [33] K. Wakil, A. Safi, D.N. Jawawi *et al.*, “Enhancement of UWE navigation model: Homepage development case study,” *International Journal of Software Engineering & Its Applications*, Vol. 8, No. 4, 2014.
- [34] P. Freudenstein, M. Nussbaumer, F. Allarding, and M. Gaedke, “A domain-specific language for the model-driven construction of advanced web-based dialogs,” in *Proceedings of the 17th international conference on World Wide Web*. ACM, 2008, pp. 1069–1070.
- [35] V. Torres, V. Pelechano, M. Ruiz, and P. Valderas, “A model driven approach for the integration of external functionality in web applications. The travel agency system,” in *Workshop on Model-driven Web Engineering (MDWE)*, 2005, pp. 1–11.
- [36] N. Moreno and A. Vallecillo, “A model-based approach for integrating third party systems with web applications,” in *Web Engineering*. Springer, 2005, pp. 441–452.
- [37] R. Cheung, “A model-driven framework for dynamic web application development,” in *Advances in Software Engineering*. Springer, 2009, pp. 29–42.
- [38] R. de Souza, R. de Barros Souto Maior *et al.*, “A model-driven method for the development of web applications user interaction layer,” in *TASE’08. 2nd IFIP/IEEE International Symposium on Theoretical Aspects of Software Engineering*. IEEE, 2008, pp. 91–98.
- [39] P. Valdera and V. Pelechano, “A survey of requirements specification in model-driven development of web applications,” *ACM Transactions on the Web*, Vol. 5, No. 2, 2011.
- [40] D. Tian, J. Wen, Y. Liu, N. Ma, and H. Wei, “A test-driven web application model based on layered approach,” in *IEEE International Conference on Information Theory and Information Security (ICITIS)*. IEEE, 2010, pp. 160–163.
- [41] A. Fernandez, S. Abrahão, and E. Insfran, “A web usability evaluation process for model-driven web development,” in *Advanced Information Systems Engineering*. Springer, 2011, pp. 108–122.
- [42] V. Torres, J. Muñoz, and V. Pelechano, “A model driven method for the integration of web applications,” in *Third Latin American Web Congress, LA-WEB*. IEEE, 2005, p. 10.
- [43] M. Bernardi, G. Di Lucca, and D. Distanto, “A model-driven approach for the fast prototyping of web applications,” in *13th IEEE International Symposium on Web Systems Evolution (WSE)*. IEEE, 2011, pp. 65–74.
- [44] A. Martin and A. Cechich, “A model-driven reengineering approach to web site personalization,” in *LA-WEB Third Latin American Web Congress*. IEEE, 2005, p. 9 pp.
- [45] S.M. Beigbeder and C.C. Castro, “An MDA approach for the development of web applications,” in *Web Engineering*. Springer, 2004, pp. 300–305.
- [46] J. Fons, V. Pelechano, O. Pastor, P. Valderas, and V. Torres, “Applying the OOWS model-driven approach for developing web applications. The internet movie database case study,” in *Web Engineering: Modelling and Implementing Web Applications*. Springer, 2008, pp. 65–108.
- [47] K. Nguyen and T. Dillon, “Atomic use case as a concept to support the MDE approach to web application development,” in *Workshop on Model-driven Web Engineering*, 2005, p. 89.
- [48] A. Cicchetti, D.D. Ruscio, R. Eramo, F. MacCarrone, and A. Pierantonio, *beContent: A model-driven platform for designing and maintaining web applications*. Springer, 2009.
- [49] M. Vasko, E. Oberortner, and S. Dustdar, “Collaborative modeling of web applications for various stakeholders,” in *Proceedings of the 9th International Conference on Web Engineering (ICWE)*, San Sebastian, Spain, 2009.
- [50] A. Langegger, J. Palkoska, and R. Wagner, “Davinci – A model-driven web engineering framework,” *International Journal of Web In-*

- formation Systems, Vol. 2, No. 2, 2006, pp. 119–134.
- [51] V. Okanović, D. Donko, and T. Mateljan, “Frameworks for model-driven development of web applications,” in *Proceedings of the 9th WSEAS international conference on Data networks, communications, computers*. World Scientific and Engineering Academy and Society (WSEAS), 2010, pp. 67–72.
- [52] L.D. Marco, F. Ferrucci, C. Gravino, F. Sarro, S. Abrahao, and J. Gomez, “Functional versus design measures for model-driven web applications: A case study in the context of web effort estimation,” in *3rd International Workshop on Emerging Trends in Software Metrics (WET-SoM)*. IEEE, 2012, pp. 21–27.
- [53] Y. Martínez, C. Cachero, M. Matera, S. Abrahao, and S. Luján, “Impact of MDE approaches on the maintainability of web applications: an experimental evaluation,” in *Conceptual Modeling – ER 2011*. Springer, 2011, pp. 233–246.
- [54] M.L. Bernardi, M. Cimitile, G.A.D. Lucca, and F.M. Maggi, “M3d: a tool for the model driven development of web applications,” in *Proceedings of the twelfth international workshop on Web information and data management*. ACM, 2012, pp. 73–80.
- [55] M. Taleb, A. Seffah, and A. Abran, “Model-driven architecture for web applications,” in *Human-Computer Interaction. Interaction Design and Usability*. Springer, 2007, pp. 1198–1205.
- [56] I. Manolescu, M. Brambilla, S. Ceri, S. Comai, and P. Fraternali, “Model-driven design and deployment of service-enabled web applications,” *ACM Transactions on Internet Technology (TOIT)*, Vol. 5, No. 3, 2005, pp. 439–479.
- [57] M. Matera, A. Maurino, S. Ceri, and P. Fraternali, “Model-driven design of collaborative web applications,” *Software: Practice and Experience*, Vol. 33, No. 8, 2003, pp. 701–732.
- [58] M. Brambilla, S. Ceri, P. Fraternali, R. Acerbis, and A. Bongio, “Model-driven design of service-enabled web applications,” in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. ACM, 2005, pp. 851–856.
- [59] A. Bozzon, M. Brambilla, and P. Fraternali, *Model-Driven Development of Audio-Visual Web Search Applications: The PHAROS Demonstration*. Springer, 2009.
- [60] G.M. Kapitsaki, D.A. Kateros, C.A. Pappas, N.D. Tselikas, and I.S. Venieris, “Model-driven development of composite web applications,” in *Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*. ACM, 2008, pp. 399–402.
- [61] H. Tai, K. Mitsui, T. Nerome, M. Abe, K. Ono, and M. Hori, “Model-driven development of large-scale web applications,” *IBM Journal of Research and Development*, Vol. 48, No. 5.6, 2004, pp. 797–809.
- [62] D. Distanto, P. Pedone, G. Rossi, and G. Canfora, “Model-driven development of web applications with UWA, MVC and JavaServer faces,” in *Web Engineering*. Springer, 2007, pp. 457–472.
- [63] P. Fraternali and P. Paolini, “Model-driven development of web applications: the AutoWeb system,” *ACM Transactions on Information Systems (TOIS)*, Vol. 18, No. 4, 2000, pp. 323–382.
- [64] R. Quintero, L. Zepeda, and L. Vega, “Model-driven software development of applications based on web services,” *International Journal of Web and Grid Services*, Vol. 6, No. 3, 2010, pp. 313–330.
- [65] F. Bolis, A. Gargantini, M. Guarnieri, E. Magri, and L. Musto, “Model-driven testing for web applications using abstract state machines,” in *Current Trends in Web Engineering*. Springer, 2012, pp. 71–78.
- [66] J. Gómez, “Model-driven web development with visualwade,” in *Web Engineering*. Springer, 2004, pp. 611–612.
- [67] D.A. Nunes and D. Schwabe, “Rapid prototyping of web applications combining domain specific languages and model driven design,” in *Proceedings of the 6th international conference on Web engineering*. ACM, 2006, pp. 153–160.
- [68] A. Cicchetti, D.D. Ruscio, and A.D. Salle, “Software customization in model driven development of web applications,” in *Proceedings of the 2007 ACM symposium on Applied computing*. ACM, 2007, pp. 1025–1030.
- [69] P. Barna, G.J. Houben, and F. Frasinca, “Specification of adaptive behavior using a general-purpose design methodology for dynamic web applications,” in *Adaptive Hypermedia and Adaptive Web-Based Systems*. Springer, 2004, pp. 283–286.
- [70] Y. Cho, W. Lee, and K. Chong, “The technique of business model driven analysis and test design for development of web applications,” *International Journal of Software En-*

- gineering and Knowledge Engineering, Vol. 15, No. 4, 2005, pp. 587–605.
- [71] J. Gómez, A. Bia, and A. Parraga, “Tool support for model-driven development of web applications,” in *Web Information Systems Engineering–WISE*. Springer, 2005, pp. 721–730.
- [72] R. Acerbis, A. Bongio, M. Brambilla, and S. Butti, “Webratio 5: An eclipse-based case tool for engineering web applications,” in *Web Engineering*. Springer, 2007, pp. 501–505.
- [73] J.L. Herrero, P. Carmona, and F. Lucio, “Towards a model-driven development of web applications,” in *WEBIST 2013 – Proceedings of the 9th International Conference on Web Information Systems and Technologies*, 2013, pp. 71–76.
- [74] R. Luo, X. Peng, Q. Lv, M. Wu, B. Peng, S. Wang, and M. Guo, “An MDA based modeling and implementation for web app,” *Journal of Software*, Vol. 8, No. 8, 2013, pp. 1881–1888.
- [75] M.L. Bernardi, G.A. Di Lucca, D. Distanto, and M. Cimitile, “Model driven evolution of web applications,” in *15th IEEE International Symposium on Web Systems Evolution (WSE)*. IEEE, 2013, pp. 45–50.
- [76] R. Rodríguez-Echeverría, F. Macías, V.M. Pavón, J.M. Conejero, and F. Sánchez-Figueroa, “Model-driven generation of a REST API from a legacy web application,” in *Current Trends in Web Engineering*. Springer, 2013, pp. 133–147.
- [77] F.J. Martínez-Ruiz, J.M. Arteaga, J. Vanderdonckt, J.M. Gonzalez-Calleros, and R. Mendoza, “A first draft of a model-driven method for designing graphical user interfaces of rich internet applications,” in *LA-Web’06 Fourth Latin American Web Congress*. IEEE, 2006, pp. 32–38.
- [78] M. Linaje, J.C. Preciado, and F. Sánchez-Figueroa, “A method for model based design of rich internet application interactive user interfaces,” in *Web Engineering*. Springer, 2007, pp. 226–241.
- [79] S. Meliá, J. Gomez, S. Perez, and O. Diaz, “A model-driven development for GWT-based rich internet applications with OOH4RIA,” in *ICWE’08 Eighth International Conference on Web Engineering*. IEEE, 2008, pp. 13–23.
- [80] Y.C. Huang, C.C. Wu, C.P. Chu *et al.*, “A new approach for web engineering based on model driven architecture,” in *International Conference on Management Learning and Business Technology Education*, 2011.
- [81] F. Valverde, O. Pastor, P. Valderas, and V. Pelechano, “A model-driven engineering approach for defining rich internet applications: a web 2.0 case study,” *Handbook of research on web*, Vol. 2, No. 3.0, 2009, pp. 40–58.
- [82] S. Meliá, J.J. Martínez, S. Mira, J.A. Osuna, and J. Gómez, *An Eclipse plug-in for model-driven development of rich internet applications*. Springer, 2010.
- [83] J.C. Preciado, M. Linaje, R. Morales-Chaparro, F. Sanchez-Figueroa, G. Zhang, C. Kroiś, and N. Koch, “Designing rich internet applications combining UWE and RUX-method,” in *Eighth International Conference on Web Engineering, ICWE’08*. IEEE, 2008, pp. 148–154.
- [84] J.M. Hermida, S. Meliá, J.J. Martínez, A. Montoyo, and J. Gómez, “Developing semantic rich internet applications with the Sm4RIA extension for OIDE,” in *Current Trends in Web Engineering*. Springer, 2012, pp. 20–25.
- [85] P. Fraternali, S. Comai, A. Bozzon, and G.T. Carughi, “Engineering rich internet applications with a model-driven approach,” *ACM Transactions on the Web (TWEB)*, Vol. 4, No. 2, 2010, p. 7.
- [86] F. Valverde and O. Pastor, *Facing the technological challenges of web 2.0: A RIA model-driven engineering approach*. Springer, 2009.
- [87] R. Rodríguez-Echeverría, J.M. Conejero, P.J. Clemente, V.M. Pavón, and F. Sánchez-Figueroa, “Model driven extraction of the navigational concern of legacy web applications,” in *Current Trends in Web Engineering*. Springer, 2012, pp. 56–70.
- [88] F.J. Martínez-Ruiz, J. Vanderdonckt, J.M. Gonzalez-Calleros, and J.M. Arteaga, “Model driven engineering of rich internet applications equipped with zoomable user interfaces,” in *LA-WEB’09 Latin American Web Congress*. IEEE, 2009, pp. 44–51.
- [89] R. Paiano, L. Mainetti, and A. Pandurino, “Model-driven and metrics-driven user experience re-modeling for rich internet applications,” in *14th IEEE International Symposium on Web Systems Evolution (WSE)*. IEEE, 2012, pp. 61–65.
- [90] G. Toffetti Carughi, “Modeling data-intensive rich internet applications with server push support,” in *Int. workshop Model-Driven Web Engineering in conjunction with ICWE*, Como (Italy), 2007.

- [91] N. Koch, M. Pigerl, G. Zhang, and T. Morozova, *Patterns for the Model-based Development of RIAs*. Springer, 2009.
- [92] G. Rossi, M. Urbieta, J. Ginzburg, D. Distanto, and A. Garrido, "Refactoring to rich internet applications. a model-driven approach," in *Eighth International Conference on Web Engineering, ICWE'08*. IEEE, 2008, pp. 1–12.
- [93] J.L.H. Agustin and P.C. Del Barco, "A model-driven approach to develop high performance web applications," *Journal of Systems and Software*, Vol. 86, No. 12, 2013, pp. 3013–3023.
- [94] J.M. Hermida, S. Meliá, A. Montoyo, and J. Gómez, "Applying model-driven engineering to the development of rich internet applications for business intelligence," *Information Systems Frontiers*, Vol. 15, No. 3, 2013, pp. 411–431.
- [95] R. Rodríguez-Echeverría, J.M. Conejero, P.J. Clemente, J.C. Preciado, and F. Sánchez-Figueroa, "Modernization of legacy web applications into rich internet applications," in *Current Trends in Web Engineering*. Springer, 2012, pp. 236–250.
- [96] R. Vdovjak and G.J. Houben, "A model-driven approach for designing distributed web information systems," in *Web Engineering*. Springer, 2005, pp. 453–464.
- [97] H.B. Zghal, M.A. Aufaure, and N.B. Mustapha, "A model-driven approach of ontological components for on-line semantic web information retrieval," *Journal of Web Engineering*, Vol. 6, No. 4, 2007, p. 309.
- [98] E. Chavarriaga and J.A. Macías, "A model-driven approach to building modern semantic web-based user interfaces," *Advances in Engineering Software*, Vol. 40, No. 12, 2009, pp. 1329–1334.
- [99] W. Sun, S. Li, D. Zhang, and Y. Yan, "A model-driven reverse engineering approach for semantic web services composition," in *WRI World Congress on Software Engineering, WCSE'09*, Vol. 3. IEEE, 2009, pp. 101–105.
- [100] M. Álvarez Álvarez, B.C.P. G-Bustelo, O. Sanjuán-Martínez, and J.M.C. Lovelle, "Bridging together semantic web and model-driven engineering," in *Distributed Computing and Artificial Intelligence*. Springer, 2010, pp. 601–604.
- [101] V. Torres, V. Pelechano, and Ó. Pastor, "Building semantic web services based on a model driven web engineering method," in *Advances in Conceptual Modeling-Theory and Practice*. Springer, 2006, pp. 173–182.
- [102] K. Musumbu, M. Diouf, and S. Maabout, "Business rules generation methods by merging model driven architecture and web semantics," in *IEEE International Conference on Software Engineering and Service Sciences (ICSESS)*. IEEE, 2010, pp. 33–36.
- [103] J. Cañadas, J. Palma, and S. Túnez, "Defining the semantics of rule-based web applications through model-driven development," *International Journal of Applied Mathematics and Computer Science*, Vol. 21, No. 1, 2011, pp. 41–55.
- [104] D. Amar Bensaber and M. Malki, "Development of semantic web services: model driven approach," in *Proceedings of the 8th International Conference on New Technologies in Distributed Systems*. ACM, 2008, p. 40.
- [105] J. Cañadas, J. Palma, and S. Túnez, *InSCo-Gen: A MDD tool for Web rule-based applications*. Springer, 2009.
- [106] J. Lee, "Model-driven business transformation and the semantic web," *Communications of the ACM*, Vol. 48, No. 12, 2005, pp. 75–77.
- [107] M. Brambilla, S. Ceri, F.M. Facca, I. Celino, D. Cerizza, and E.D. Valle, "Model-driven design and development of semantic web service applications," *ACM Transactions on Internet Technology (TOIT)*, Vol. 8, No. 1, 2007, p. 3.
- [108] C. Hahn, S. Nesbigall, S. Warwas, I. Zinnikus, M. Klusch, and K. Fischer, "Model-driven approach to the integration of multiagent systems and semantic web services," in *12th Enterprise Distributed Object Computing Conference Workshops*. IEEE, 2008, pp. 317–324.
- [109] R. Grønmo and M.C. Jaeger, "Model-driven semantic web service composition," in *12th Asia-Pacific Software Engineering Conference, APSEC'05*. IEEE, 2005, p. 8.
- [110] M. Belchior, D. Schwabe, and F.S. Parreiras, "Role-based access control for model-driven web applications," in *Web Engineering*. Springer, 2012, pp. 106–120.
- [111] A. Staikopoulos, O. Cliffe, R. Popescu, J. Padget, and S. Clarke, "Template-based adaptation of semantic web services with model-driven engineering," *IEEE Transactions on Services Computing*, Vol. 3, No. 2, 2010, pp. 116–130.
- [112] N.A. Tavares and S. Vale, "A model driven approach for the development of semantic RESTful web services," in *Proceedings of International Conference on Information Integration and Web-based Applications & Services*. ACM, 2013, p. 290.

- [113] H. Jinkui, W. Jiancheng, and Y. Yongtang, "A semantics-reconstruction based model-driven development approach for web information systems," in *Chinese Control Conference*. IEEE, 2007, pp. 344–348.
- [114] W. El Kaim, P. Studer, and P.A. Muller, "Model driven architecture for agile web information system engineering," in *Object-Oriented Information Systems*. Springer, 2003, pp. 299–303.
- [115] C. Batini, D. Bolchini, S. Ceri, M. Matera, A. Maurino, and P. Paolini, "The UM-MAIS methodology for multi-channel adaptive web information systems," *World Wide Web*, Vol. 10, No. 4, 2007, pp. 349–385.
- [116] A. Bozzon, T. Iofciu, W. Nejdl, and S. Tönnies, "Integrating databases, search engines and web applications: a model-driven approach," in *Web Engineering*. Springer, 2007, pp. 210–225.
- [117] I. Celino, E. Della Valle, D. Cerizza, and A. Turati, "Squiggle: an experience in model-driven development of real-world semantic search engines," in *Web Engineering*. Springer, 2007, pp. 485–490.
- [118] R. Kumar, J. Bopaiah, P. Jain, N. Nalini, and K.C. Sekaran, "Model driven approach for developing cloud application," *International Journal of Scientific & Technology Research*, Vol. 2, No. 10, 2013.
- [119] P. Díaz, I. Aedo, D. Sanz, and A. Malizia, "A model-driven approach for the visual specification of role-based access control policies in web systems," in *VL/HCC 2008, IEEE Symposium on Visual Languages and Human-Centric Computing*. IEEE, 2008, pp. 203–210.
- [120] P. Xiong and L. Peyton, "A model-driven penetration test framework for web applications," in *Eighth Annual International Conference on Privacy Security and Trust (PST)*. IEEE, 2010, pp. 173–180.
- [121] M. Jensen and S. Feja, "A security modeling approach for web-service-based business processes," in *16th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems*. IEEE, 2009, pp. 340–347.
- [122] B. Hoisl and S. Sobernig, "Integrity and confidentiality annotations for service interfaces in SoaML models," in *Sixth International Conference on Availability, Reliability and Security (ARES)*. IEEE, 2011, pp. 673–679.
- [123] Y. Nakamura, M. Tatsubori, T. Imamura, and K. Ono, "Model-driven security based on a web services security architecture," in *IEEE International Conference on Services Computing*, Vol. 1. IEEE, 2005, pp. 7–15.
- [124] Z. Ma, C. Wagner, and T. Bleier, "Model-driven security for web services in e-government system: Ideal and real," in *2011 7th International Conference on Next Generation Web Services Practices (NWeSP)*. IEEE, 2011, pp. 221–226.
- [125] P. Patil and S. Pawar, "Remote agent based automated framework for threat modelling, vulnerability testing of SOA solutions and web services," in *2012 World Congress on Internet Security (WorldCIS)*. IEEE, 2012, pp. 127–131.
- [126] M. Busch, N. Koch, M. Masi, R. Pugliese, and F. Tiezzi, "Towards model-driven development of access control policies for web applications," in *Proceedings of the Workshop on Model-Driven Security*. ACM, 2012, p. 4.
- [127] E. Oberortner, M. Vasko, and S. Dustdar, "Towards modeling role-based pageflow definitions within web applications," in *Proc. of the 4th International Workshop on Model-Driven Web Engineering (MDWE 2008)*, Vol. 389, 2008, pp. 1–15.
- [128] S. Kent, "Model driven engineering," in *Integrated formal methods*. Springer, 2002, pp. 286–298.
- [129] Z. Ma, C. Wagner, R. Woitsch, F. Skopik, and T. Bleier, "Model-driven security: from theory to application," *International Journal of Computer Information Systems and Industrial Management Applications*, Vol. 5, 2013, pp. 151–158.
- [130] F. Domínguez-Mayo, M. Escalona, M. Mejias, and A. Torres, "A quality model in a quality evaluation framework for mdwe methodologies," in *2010 Fourth International Conference on Research Challenges in Information Science (RCIS)*. IEEE, 2010, pp. 495–506.
- [131] R. Grønmo and M.C. Jaeger, "Model-driven methodology for building QoS-optimised web service compositions," in *Distributed Applications and Interoperable Systems*. Springer, 2005, pp. 68–82.
- [132] P. Fraternali, P.L. Lanzi, M. Matera, and A. Maurino, "Model-driven web usage analysis for the evaluation of web application quality," *J. Web Eng.*, Vol. 3, No. 2, 2004, pp. 124–152.
- [133] G. Ortiz and B. Bordbar, "Model-driven quality of service for web services: an aspect-oriented approach," in *ICWS'08. IEEE International Conference on Web Services*. IEEE, 2008, pp. 748–751.

- [134] F. Domínguez-Mayo, M.J. Escalona, M. Mejías, M. Ross, and G. Staples, "Quality evaluation for model-driven web engineering methodologies," *Information and Software Technology*, Vol. 54, No. 11, 2012, pp. 1265–1282.
- [135] F. Domínguez-Mayo, M. Escalona, and M. Mejías, "Quality issues on model-driven web engineering methodologies," in *Information Systems Development*. Springer, 2011, pp. 295–306.
- [136] F. Domínguez-Mayo, M.J. Escalona, and M. Mejías, *QuEF (quality evaluation framework) for model-driven web methodologies*. Springer, 2010.
- [137] N. Li, Q.q. Ma, J. Wu, M.z. Jin, and C. Liu, "A framework of model-driven web application testing," in *COMPSAC '06. 30th Annual International Computer Software and Applications Conference*, Vol. 2. IEEE, 2006, pp. 157–162.
- [138] S. Haustein and J. Pleumann, "A model-driven runtime environment for web applications," *Software & Systems Modeling*, Vol. 4, No. 4, 2005, pp. 443–458.
- [139] E.R. Luna, J. Grigera, and G. Rossi, *Bridging test and model-driven approaches in web engineering*. Springer, 2009.
- [140] E. Escott, P. Strooper, J. Steel, and P. King, "Integrating model-based testing in model-driven web engineering," in *18th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2011, pp. 187–194.
- [141] P. Strooper, "A model-driven approach to developing and testing web applications," in *2014 International Conference on Information, Communication Technology and System (ICTS)*. IEEE, 2014, pp. 3–4.
- [142] A. Safi, D.N. Jawawi, K. Wakil *et al.*, "Web services composition with redundancy consideration," in *IEEE Conference on Open Systems (ICOS)*. IEEE, 2013, pp. 112–117.
- [143] G. Ortiz and J. Hernandez, "A case study on integrating extra-functional properties in web service model-driven development," in *ICIW'07, Second International Conference on Internet and Web Applications and Services*. IEEE, 2007, pp. 35–35.
- [144] M.B. Blake, "A lightweight software design process for web services workflows," in *ICWS'06, International Conference on Web Services*. IEEE, 2006, pp. 411–418.
- [145] D. Kateros, G.M. Kapitsaki, N.D. Tselikas, I.S. Venieris *et al.*, "A methodology for model-driven web application composition," in *SCC'08, IEEE International Conference on Services Computing*, Vol. 2. IEEE, 2008, pp. 489–492.
- [146] V. Torres, J. Muñoz, and V. Pelechano, "A model driven method for the integration of web applications," in *Third Latin American Web Congress (LA-WEB)*. IEEE, 2005, pp. 10–pp.
- [147] R. Kulesza, S.R. Meira, T.P. Ferreira, E.S. Alexandre, L. Guido Filho, M.C.M. Neto, and C.A. San, "A model-driven approach for integration of interactive applications and web services: A case study in interactive digital TV platform," in *IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*. IEEE, 2012, pp. 266–271.
- [148] A. Charfi, S.H. Turki, A. Cha? bane, H. Witteborg, and R. Bouaziz, "A model-driven approach to developing web service compositions based on BPMN4SOA," *International Journal of Reasoning-Based Intelligent Systems*, Vol. 3, No. 3-4, 2011, pp. 194–204.
- [149] G. Botterweck, "A model-driven approach to the engineering of multiple user interfaces," in *Models in Software Engineering*. Springer, 2007, pp. 106–115.
- [150] X. Yu, Y. Zhang, T. Zhang, L. Wang, J. Hu, J. Zhao, and X. Li, "A model-driven development framework for enterprise web services," *Information Systems Frontiers*, Vol. 9, No. 4, 2007, pp. 391–409.
- [151] A. Achilleos, G.M. Kapitsaki, and G.A. Papadopoulos, "A model-driven framework for developing web service oriented applications," in *Current Trends in Web Engineering*. Springer, 2012, pp. 181–195.
- [152] S. Comai and D. Mazza, "A model-driven methodology to the content layout problem in web applications," *ACM Transactions on the Web (TWEB)*, Vol. 6, No. 3, 2012, p. 10.
- [153] A. Achilleos, N. Paspallis, G. Papadopoulos *et al.*, "Automating the development of device-aware web services: A model-driven approach," in *IEEE 35th Annual Computer Software and Applications Conference (COMPSAC)*. IEEE, 2011, pp. 535–540.
- [154] X. Qafmolla and V.C. Nguyen, "Automation of web services development using model driven techniques," in *The 2nd International Conference on Computer and Automation Engineering (ICCAE)*, Vol. 3. IEEE, 2010, pp. 190–194.
- [155] K. Pfadenhauer, S. Dustdar, and B. Kittl, "Challenges and solutions for model driven web service composition," in *14th IEEE International Workshops on Enabling Technolo-*



- gies: Infrastructure for Collaborative Enterprise*. IEEE, 2005, pp. 126–131.
- [156] K. Pfadenhauer, S. Dustdar, and B. Kittl, “Comparison of two distinctive model driven web service orchestration proposals,” in *Seventh IEEE International Conference on E-Commerce Technology Workshops*. IEEE, 2005, pp. 29–36.
- [157] F. Valverde and O. Pastor, “Dealing with REST services in model-driven web engineering methods,” *V Jornadas Científico-Técnicas en Servicios Web y SOA, JSWEB*, 2009.
- [158] M. Ruiz, V. Pelechano, and Ó. Pastor, “Designing web services for supporting user tasks: A model driven approach,” in *Advances in Conceptual Modeling-Theory and Practice*. Springer, 2006, pp. 193–202.
- [159] N. Blum, T. Magedanz, J. Kleessen, and T. Margaria, “Enabling extreme model driven design of Parlay X-based communications services for end-to-end multiplatform service orchestrations,” in *14th IEEE International Conference on Engineering of Complex Computer Systems*. IEEE, 2009, pp. 240–247.
- [160] B. Bauer and J.P. Müller, “MDA applied: From sequence diagrams to web service choreography,” in *Web Engineering*. Springer, 2004, pp. 132–136.
- [161] L. Zhu, I. Gorton, Y. Liu, and N.B. Bui, “Model driven benchmark generation for web services,” in *Proceedings of the 2006 international workshop on Service-oriented software engineering*. ACM, 2006, pp. 33–39.
- [162] R. Barrett and C. Pahl, “Model driven design of distribution patterns for web service compositions,” in *ICWS’06. International Conference on Web Services*. IEEE, 2006, pp. 887–888.
- [163] R. Barrett, L.M. Patcas, C. Pahl, and J. Murphy, “Model driven distribution pattern design for dynamic web service compositions,” in *Proceedings of the 6th international conference on Web engineering*. ACM, 2006, pp. 129–136.
- [164] M. Ruiz and V. Pelechano, “Model driven design of web service operations using web engineering practices,” in *Emerging Web Services Technology*. Springer, 2007, pp. 83–100.
- [165] V. De Castro, J.M. Vara, and E. Marcos, “Model transformation for service-oriented web applications development,” in *MDWE*, 2007.
- [166] G. Ortiz, J. Hernández, and F. Sánchez, “Model driven extra-functional properties for web services,” in *IEEE Services Computing Workshops, SCW’06*. IEEE, 2006, pp. 113–120.
- [167] S. Lohmann, J.W. Kaltz, and J. Ziegler, “Model-driven dynamic generation of context-adaptive web user interfaces,” in *Models in Software Engineering*. Springer, 2007, pp. 116–125.
- [168] C. Dumez, J. Gaber, and M. Wack, “Model-driven engineering of composite web services using UML-S,” in *Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*. ACM, 2008, pp. 395–398.
- [169] M. Ribarić, D. Gašević, M. Milanović, A. Giurca, S. Lukichev, and G. Wagner, “Model-driven engineering of rules for web services,” in *Generative and Transformational Techniques in Software Engineering II*. Springer, 2008, pp. 377–395.
- [170] T. Dirgahayu, “Model-driven engineering of web service compositions: A transformation from ISDL to BPEL,” Master’s thesis, University of Twente, 2005.
- [171] R. Grønmo, D. Skogan, I. Solheim, and J. Oldevik, “Model-driven web services development,” in *IEEE International Conference on e-Technology, e-Commerce and e-Service, EEE’04*. IEEE, 2004, pp. 42–45.
- [172] B. Li, Y. Zhou, and J. Pang, “Model-driven automatic generation of verified bpm code for web service composition,” in *Asia-Pacific Software Engineering Conference, APSEC’09*. IEEE, 2009, pp. 355–362.
- [173] H. Zhang, J. Liu, L. Zheng, and J. Wang, “Modeling of web service development process based on MDA and procedure blueprint,” in *IEEE/ACIS 11th International Conference on Computer and Information Science (ICIS)*. IEEE, 2012, pp. 422–427.
- [174] K. Nguyen, T.S. Dillon, and E. Danielsen, “The concept of web event and a practical model-driven approach to web information system development,” *International Journal of Web Information Systems*, Vol. 2, No. 1, 2006, pp. 19–36.
- [175] P. Hrastnik and W. Winiwarter, “Using advanced transaction meta-models for creating transaction-aware web service environments,” *International Journal of Web Information Systems*, Vol. 1, No. 2, 2005, pp. 89–100.
- [176] N. Glombitza, D. Pfisterer, and S. Fischer, “Using state machines for a model driven development of web service-based sensor network applications,” in *Proceedings of the 2010 ICSE Workshop on Software Engineering for Sensor Network Applications*. ACM, 2010, pp. 2–7.
- [177] Y. Taher, J. Boubeta-Puig, W.J. van den Heuvel, G. Ortiz, and I. Medina-Bulo, “A model-driven approach for web service adap-

- tation using complex event processing,” in *Advances in Service-Oriented and Cloud Computing*. Springer, 2013, pp. 346–359.
- [178] R. Maraoui, E. Cariou, and B. Ayeb, “A model-driven engineering approach for the formal verification of composite web services,” in *IEEE 22nd International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*. IEEE, 2013, pp. 266–271.
- [179] W. Li, Y. Badr, and F. Biennier, “Improving web service composition with user requirement transformation and capability model,” in *On the Move to Meaningful Internet Systems: OTM 2013 Conferences*. Springer, 2013, pp. 300–307.
- [180] C. Dumez, M. Bakhouya, J. Gaber, M. Wack, and P. Lorenz, “Model-driven approach supporting formal verification for web service composition protocols,” *Journal of network and computer applications*, Vol. 36, No. 4, 2013, pp. 1102–1115.
- [181] A. Kalantari, S. Ibrahim, S.G.H. Tabatabaei, and H. Taherdoot, “A categorization of model-driven approaches for developing semantic web service,” in *3rd International Conference on Data Mining and Intelligent Information Technology Applications (ICMiA)*. IEEE, 2011, pp. 92–97.
- [182] S. Chung, S. Davalos, C. Niiyama, D. Won, S.H. Baeg, and S. Park, “A uml model-driven business process development methodology for a virtual enterprise using SOA & ESB,” in *IEEE Asia-Pacific Services Computing Conference, APSCC*. IEEE, 2009, pp. 246–253.
- [183] M. Ruiz, P. Valderas, V. Torres, and V. Pelechano, “A model driven approach to design web services in a web engineering method,” in *CAiSE Short Paper Proceedings*, 2005.
- [184] C. Zhao, Z. Duan, and M. Zhang, “A model-driven approach for dynamic web service composition,” in *WRI World Congress on Software Engineering, WCSE’09*, Vol. 4. IEEE, 2009, pp. 273–277.
- [185] C. Momm, M. Gebhart, and S. Abeck, “A model-driven approach for monitoring business performance in web service compositions,” in *Fourth International Conference on Internet and Web Applications and Services ICIW’09*. IEEE, 2009, pp. 343–350.
- [186] Y.H. Liu, J.S. Yih, F. Pinel, and T. Chieu, “A model-driven SOA implementation of multi-channel websphere commerce gift center,” in *IEEE International Conference on e-Business Engineering, ICEBE’08*. IEEE, 2008, pp. 29–34.
- [187] K. Dahman, F. Charoy, and C. Godart, “Generation of component based architecture from business processes: model driven engineering for SOA,” in *8th European Conference on Web Services (ECOWS)*. IEEE, 2010, pp. 155–162.
- [188] N. Zhou, L.J. Zhang, Y.M. Chee, and L. Chen, “Legacy asset analysis and integration in model-driven SOA solution,” in *International Conference on Services Computing (SCC)*. IEEE, 2010, pp. 554–561.
- [189] M. Bajohr and T. Margaria, “Model-driven self-reconfiguration for highly available SOAs,” in *Sixth IEEE Conference and Workshops on Engineering of Autonomic and Autonomous Systems, EASE*. IEEE, 2009, pp. 13–22.
- [190] M. Leotta, G. Reggio, F. Ricca, and E. Asteiano, “Towards a lightweight model driven method for developing SOA systems using existing assets,” in *14th IEEE International Symposium on Web Systems Evolution (WSE)*. IEEE, 2012, pp. 51–60.
- [191] M. Fritzsche, W. Gilani, I. Spence, T.J. Brown, P. Kilpatrick, and R. Bashroush, “Towards performance related decision support for model driven engineering of enterprise SOA applications,” in *15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems, ECBS*. IEEE, 2008, pp. 57–65.
- [192] E. Sosa, P.J. Clemente, J.M. Conejero, and R. Rodriguez-Echeverria, “A model-driven process to modernize legacy web applications based on service oriented architectures,” in *15th IEEE International Symposium on Web Systems Evolution (WSE)*. IEEE, 2013, pp. 61–70.
- [193] M. Wagner, D. Zöbel, and A. Meroth, “Model-driven development of SOA-based driver assistance systems,” *ACM SIGBED Review*, Vol. 10, No. 1, 2013, pp. 37–42.
- [194] M.J. Escalona, C. Parra, F. Martín, J. Nieto, A. Llergo, and F. Pérez, “A practical example for model-driven web requirements,” in *Information Systems Development*. Springer, 2009, pp. 157–168.
- [195] H. Wada, J. Suzuki, and K. Oba, “A model-driven development framework for non-functional aspects in service oriented architecture,” *Web Services Research for Emerging Applications: Discoveries and Trends: Discoveries and Trends*, 2010, p. 358.

- [196] J.A. Aguilar, I. Garrigós, J.N. Mazón, and J. Trujillo, "An MDA approach for goal-oriented requirement analysis in web engineering," *Journal of Universal Computer Science*, Vol. 16, No. 17, 2010, pp. 2475–2494.
- [197] J.A. Aguilar, I. Garrigós, J.N. Mazón, and A. Zaldívar, "Dealing with dependencies among functional and non-functional requirements for impact analysis in web engineering," in *Computational Science and Its Applications–ICCSA 2012*. Springer, 2012, pp. 116–131.
- [198] A. Fernandez, S. Abrahão, E. Insfran, and M. Matera, "Further analysis on the validation of a usability inspection method for model-driven web development," in *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement*. ACM, 2012, pp. 153–156.
- [199] E.R. Luna, J.I. Panach, J.I. Grigera, G. Rossi, and O. Pastor, "Incorporating usability requirements in a test/model-driven web engineering approach," *J. Web Eng.*, Vol. 9, No. 2, 2010, pp. 132–156.
- [200] A. Fernandez, E. Insfran, and S.M. Abrahão, "Integrating a usability model into model-driven web development processes," in *WISE*. Springer, 2009, pp. 497–510.
- [201] F. Molina and A. Toval, "Integrating usability requirements that can be evaluated in design time into model driven engineering of web information systems," *Advances in Engineering Software*, Vol. 40, No. 12, 2009, pp. 1306–1317.
- [202] P. Valderas and V. Pelechano, "Introducing requirements traceability support in model-driven development of web applications," *Information and Software Technology*, Vol. 51, No. 4, 2009, pp. 749–768.
- [203] J.C. Castrejón, R. López-Landa, and R. Lozano, "Model2Roo: a model driven approach for web application development based on the eclipse modeling framework and spring roo," in *Electrical Communications and Computers (CONIELECOMP), 2011 21st International Conference on*. IEEE, 2011, pp. 82–87.
- [204] M.J. Escalona and G. Aragón, "NDT. A model-driven approach for web requirements," *IEEE Transactions on Software Engineering*, Vol. 34, No. 3, 2008, pp. 377–390.
- [205] N. Koch and S. Kozuruba, "Requirements models as first class entities in model-driven web engineering," in *Current Trends in Web Engineering*. Springer, 2012, pp. 158–169.
- [206] A. Fernandez, S. Abrahão, and E. Insfran, "Towards to the validation of a usability evaluation method for model-driven web development," in *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*. ACM, 2010, p. 54.
- [207] T.J. Bittar, R.P. Fortes, L.L. Lobato, and W.M. Watanabe, "Web communication and interaction modeling using model-driven development," in *Proceedings of the 27th ACM international conference on Design of communication*. ACM, 2009, pp. 193–198.
- [208] Y. Martínez, C. Cachero, and S. Meliá, "Empirical study on the maintainability of web applications: Model-driven engineering vs code-centric," *Empirical Software Engineering*, Vol. 19, No. 6, 2014, pp. 1887–1920.
- [209] A. Fernandez, S. Abrahão, and E. Insfran, "Empirical validation of a usability inspection method for model-driven web development," *Journal of Systems and Software*, Vol. 86, No. 1, 2013, pp. 161–186.
- [210] A. Fernandez, S. Abrahão, E. Insfran, and M. Matera, "Usability inspection in model-driven web development: Empirical validation in webml," in *Model-Driven Engineering Languages and Systems*. Springer, 2013, pp. 740–756.
- [211] F.J. Domínguez-Mayo, M.J. Escalona, M. Mejías, and J. Torres, "Studying maintainability on model-driven web methodologies," in *Information Systems Development*. Springer, 2011, pp. 195–206.
- [212] S. Abrahão, J. Gomez, and E.M.E. Insfran, "A model-driven measurement procedure for sizing web applications," in *Conference on Model-Driven Engineering Languages and Systems (MODELS 2007), Nashville, TN, USA, September, 2007*.
- [213] S. Meliá and J. Gómez, "Applying WebSA to a case study: A travel agency system," in *Workshop on Model-driven Web Engineering*. Cite-seer, 2005, p. 30.
- [214] E. Escott, P. Strooper, J.G. Suß, and P. King, "Architecture-centric model-driven web engineering," in *18th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2011, pp. 106–113.
- [215] J. Pu, H. Yang, B. Xu, L. Xu, and W.C.C. Chu, "Combining MDE and UML to reverse engineer web-based legacy systems," in *Annual IEEE International Computer Software*

- and *Applications Conference*. IEEE, 2008, pp. 718–725.
- [216] L. Tambour, V. Houles, L. Cohen-Jonathan, V. Auffray, P. Escande, and E. Jallas, “Design of a model-driven web decision support system in agriculture: Scientific models to the final software,” *Advances in Modeling Agricultural Systems*, Vol. 25, 2009, p. 67.
- [217] M.M. Urbietta, G. Rossi, S. Gordillo, W. Schwinger, W. Retschitzegger, and M.J. Escalona, “Identifying and modelling complex workflow requirements in web applications,” in *Current Trends in Web Engineering*. Springer, 2012, pp. 146–157.
- [218] P. Freudenstein, J. Buck, M. Nussbaumer, and M. Gaedke, “Model-driven construction of workflow-based web applications with domain-specific languages,” in *MDWE*, 2007.
- [219] P. Dolog, “Model-driven navigation design for semantic web applications with the UML-Guide,” in *ICWE Workshops*, 2004, pp. 75–86.
- [220] E. Escott, P. Strooper, P. King, and I.J. Hayes, “Model-driven web form validation with UML and OCL,” in *Current Trends in Web Engineering*. Springer, 2012, pp. 223–235.
- [221] S. Jeschke, O. Pfeiffer, and H. Vieritz, “Using web accessibility patterns for web application development,” in *Proceedings of the 2009 ACM symposium on Applied Computing*. ACM, 2009, pp. 129–135.
- [222] A.R. Guzmán, V. López, F. Valverde, and J.I. Panach, “Web 2.0 patterns: A model-driven engineering approach,” in *Sixth International Conference on Research Challenges in Information Science (RCIS)*. IEEE, 2012, pp. 1–2.
- [223] C. Dumez, J. Gaber, and M. Wack, “Web services composition using UML-S: a case study,” in *IEEE GLOBECOM Workshops*. IEEE, 2008, pp. 1–6.
- [224] R. Popp, H. Kaindl, and D. Raneburger, “Connecting interaction models and application logic for model-driven generation of web-based graphical user interfaces,” in *20th Asia-Pacific Software Engineering Conference (APSEC)*, Vol. 1. IEEE, 2013, pp. 215–222.
- [225] I.C. Hsu, “Visual modeling for web 2.0 applications using model driven architecture approach,” *Simulation Modelling Practice and Theory*, Vol. 31, 2013, pp. 63–76.
- [226] R. Acerbis, A. Bongio, M. Brambilla, M. Tisi, S. Ceri, and E. Tosetti, “Developing ebusiness solutions with a model driven approach: the case of acer EMEA,” in *Web Engineering*. Springer, 2007, pp. 539–544.
- [227] G. Zhuang and J. Du, “Mda-based modeling and implementation of e-commerce web applications in WebML,” in *Second International Workshop on Computer Science and Engineering, WCSE’09*, Vol. 2. IEEE, 2009, pp. 507–510.
- [228] Y. Li, J. Shen, J. Shi, W. Shen, Y. Huang, and Y. Xu, “Multi-model driven collaborative development platform for service-oriented e-business systems,” *Advanced Engineering Informatics*, Vol. 22, No. 3, 2008, pp. 328–339.
- [229] P. Hernández, O. Glorio, I. Garrigós, and J.N. Mazón, “Towards a model-driven framework for web usage warehouse development,” in *Advances in Conceptual Modeling. Recent Developments and New Directions*. Springer, 2011, pp. 336–337.
- [230] J. Martinez, C. Lopez, E. Ulacia, and M. del Hierro, “Towards a model-driven product line for web systems,” in *5th Model-Driven Web Engineering Workshop, MDWE*, 2009, pp. 1–15.
- [231] P. Lachenmaier, F. Ott, A. Immerz, and A. Richter, “Community mashup a flexible social mashup based on a model-driven-approach,” in *IEEE International Conference on Information Reuse and Integration (IRI)*. IEEE, 2011, pp. 48–51.
- [232] M. Brambilla and A. Mauri, “Model-driven development of social network enabled applications with WebML and social primitives,” in *Current Trends in Web Engineering*. Springer, 2012, pp. 41–55.
- [233] F. Chen, H. Yang, H. Zhou, B. Qiao, and H. Deng, “Web-based system evolution in model driven architecture,” in *10th International Symposium on Web Site Evolution, WSE*. IEEE, 2008, pp. 69–72.
- [234] N.V. Cuong and X. Qafmolla, “Model transformation in web engineering and automated model driven development,” *International Journal of Modeling and Optimization*, Vol. 1, No. 1, 2011, pp. 7–12.
- [235] M. Brambilla, P. Fraternali, and M. Tisi, “A metamodel transformation framework for the migration of WebML models to MDA,” in *MDWE, CEUR Workshop Proceedings*, Vol. 389, 2008, pp. 91–105.
- [236] N. Moreno, S. Meliá, N. Koch, and A. Vallecillo, “Addressing new concerns in model-driven web engineering approaches,” in *Web Information Systems Engineering-WISE 2008*. Springer, 2008, pp. 426–442.
- [237] S. Meliá and J. Gómez, *Applying transformations to model driven development of web applications*. Springer, 2005.

- [238] N. Koch, A. Knapp, and S. Kozuruba, "Assessment of effort reduction due to model-to-model transformations in the web domain," in *Web Engineering*. Springer, 2012, pp. 215–222.
- [239] P. Giner, V. Torres, and V. Pelechano, "Bridging the gap between BPMN and WS-BPEL. M2M transformations in practice," *MDWE*, Vol. 261, 2007.
- [240] P. Valderas, J. Fons, and V. Pelechano, "From web requirements to navigational design – A transformational approach," in *Web Engineering*. Springer, 2005, pp. 506–511.
- [241] G.J. Houben, N. Koch, G. Rossi, and A. Vallecillo, "Guest editorial to the theme section on model-driven web engineering," *Software and Systems Modeling*, 2013, pp. 1–3.
- [242] M. Brambilla and P. Fraternali, "Implementing the semantics of BPMN through model-driven web application generation," in *Business Process Model and Notation*. Springer, 2011, pp. 124–129.
- [243] M. Brambilla and P. Fraternali, "Large-scale model-driven engineering of web user interaction: The WebML and webratio experience," *Science of Computer Programming*, Vol. 89, 2014, pp. 71–87.
- [244] S. Meliá, A. Kraus, and N. Koch, "MDA transformations applied to web application development," in *Web Engineering*. Springer, 2005, pp. 465–471.
- [245] H.A. Schmid, "Model driven architecture with OOHDm," in *International Conference on Web Engineering (ICWE) Workshops*, 2004, pp. 12–25.
- [246] D. Di Ruscio and A. Pierantonio, "Model transformations in the development of data-intensive web applications," in *Advanced Information Systems Engineering*. Springer, 2005, pp. 475–490.
- [247] P. Hernández, I. Garrigós, and J.N. Mazón, "Model-driven development of multidimensional models from web log files," in *Advances in Conceptual Modeling—Applications and Challenges*. Springer, 2010, pp. 170–179.
- [248] P. Fraternali and M. Tisi, *Multi-level tests for model driven web applications*. International Conference on Web Engineering, 2010.
- [249] H.A. Schmid and O. Donnerhak, "OOHDMDA – an MDA approach for OOHDm," in *Web Engineering*. Springer, 2005, pp. 569–574.
- [250] M.A.O. Mukhtar, M.F.B. Hassan, and J.B. Jaafar, "Optimizing method to provide model transformation of model-driven architecture as web-based services," in *International Conference on Computer & Information Science (IC-CIS)*, Vol. 2. IEEE, 2012, pp. 874–879.
- [251] M. Wimmer, N. Moreno, and A. Vallecillo, "Systematic evolution of WebML models by coupled transformations," in *Web Engineering*. Springer, 2012, pp. 185–199.
- [252] R. Akkiraju, T. Mitra, N. Ghosh, D. Saha, U. Thulasiram, and S. Chakraborty, "Toward the development of cross-platform business applications via model-driven transformations," in *World Conference on Services*. IEEE, 2009, pp. 585–592.
- [253] A. Fatolahi, S.S. Somé, and T.C. Lethbridge, "Towards a semi-automated model-driven method for the generation of web-based applications from use cases," in *4th Model Driven Web Engineering Workshop*, 2008, p. 31.
- [254] H. Heitkoetter, "Transforming PICTURE to BPMN 2.0 as part of the model-driven development of electronic government systems," in *44th Hawaii International Conference on System Sciences (HICSS)*. IEEE, 2011, pp. 1–10.
- [255] P. Valderas, J. Fons, and V. Pelechano, "Transforming web requirements into navigational models: AN MDA based approach," in *Conceptual Modeling—ER 2005*. Springer, 2005, pp. 320–336.
- [256] S. Meliá, J. Gómez, and J.L. Serrano, "WebTE: MDA transformation engine for web applications," in *Web Engineering*. Springer, 2007, pp. 491–495.
- [257] M.A.O. Mukhtar, M.F.B. Hassan, J. Bin Jaafar, and L.A. Rahim, "Enhanced approach for developing web applications using model driven architecture," in *International Conference on Research and Innovation in Information Systems (ICRIIS)*. IEEE, 2013, pp. 145–150.
- [258] A. Fatolahi, S.S. Somé, and T.C. Lethbridge, "A meta-model for model-driven web development," *International Journal of Software and Informatics*, Vol. 6, No. 2, 2012, pp. 125–162.
- [259] M.J. Escalona, J.J. Gutiérrez, F. Morero, C. Parra, J. Nieto, F. Pérez, F. Martín, and A. Llergo, "A practical environment to apply model-driven web engineering," in *Information Systems Development*. Springer, 2010, pp. 249–258.
- [260] L.A. Ricci and D. Schwabe, "An authoring environment for model-driven web applications," in *Proceedings of the 12th Brazilian Symposium on Multimedia and the web*. ACM, 2006, pp. 11–19.

- [261] A. Schauerhuber, M. Wimmer, and E. Kapsammer, "Bridging existing web modeling languages to model-driven engineering: a meta-model for WebML," in *Workshop proceedings of the sixth international conference on Web engineering*. ACM, 2006, p. 5.
- [262] A. Schauerhuber, M. Wimmer, E. Kapsammer, W. Schwinger, and W. Retschitzegger, "Bridging webml to model-driven engineering: from document type definitions to meta object facility," *Software, IET*, Vol. 1, No. 3, 2007, pp. 81–97.
- [263] J.M. Rivero, G. Rossi, J. Grigera, J. Burella, E.R. Luna, and S. Gordillo, *From mockups to user interface models: an extensible model driven approach*. Springer, 2010.
- [264] B. De Silva and A. Ginige, "Meta-model to support end-user development of web based business information systems," in *Web Engineering*. Springer, 2007, pp. 248–253.
- [265] D. Karagiannis, V. Hrgovic, and R. Woitsch, "Model driven design for e-applications: The meta model approach," in *Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services*. ACM, 2011, pp. 451–454.
- [266] S. Ceri, M. Brambilla, and P. Fraternali, "The history of webml lessons learned from 10 years of model-driven development of web applications," in *Conceptual Modeling: Foundations and Applications*. Springer, 2009, pp. 273–292.
- [267] G. Jomier, G. Dodinet, and M. Zam, "The United States of a meta-model build with MyDraft an agile model-driven cloud-based platform for data-oriented rich web applications," 2012.
- [268] N. Koch and A. Kraus, "Towards a common metamodel for the development of web applications," in *Web Engineering*. Springer, 2003, pp. 497–506.
- [269] C.C. Castro, S. Meliá, M. Genero, G. Poels, and C. Calero, "Towards improving the navigability of web applications: a model-driven approach," *European Journal of Information Systems*, Vol. 16, No. 4, 2007, pp. 420–447.
- [270] D. Ruiz-González, N. Koch, C. Kroiss, J.R. Romero, and A. Vallecillo, "Viewpoint synchronization of UWE models," in *Proc. 5th International Workshop on Model-Driven Web Engineering*, 2009, pp. 46–60.
- [271] R. Cheung, "XFlash—a web application design framework with model-driven methodology," *International Journal of U- and E-service, Science and Technology*, Vol. 1, No. 1, 2008, pp. 47–54.
- [272] X. Qafmolla and N.V. Cuong, "A two-way meta-modeling approach in web engineering," *Global Journal on Technology*, Vol. 3, 2013.
- [273] T. Jiang, J. Ying, M. Wu, and C. Jin, "A method for model-driven development of adaptive web applications," in *12th International Conference on Computer Supported Cooperative Work in Design, CSCWD*. IEEE, 2008, pp. 386–391.
- [274] G. Grossmann, M. Schrefl, and M. Stumptner, "A model-driven framework for runtime adaptation of web service compositions," in *Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. ACM, 2011, pp. 184–189.
- [275] C. Dorn and R.N. Taylor, "Architecture-driven modeling of adaptive collaboration structures in large-scale social web applications," in *Web Information Systems Engineering-WISE 2012*. Springer, 2012, pp. 143–156.
- [276] I. Kurtev and K. van den Berg, "Building adaptable and reusable XML applications with model transformations," in *Proceedings of the 14th international conference on World Wide Web*. ACM, 2005, pp. 160–169.
- [277] S. Ceri, P. Dolog, M. Matera, and W. Nejdl, "Model-driven design of web applications with client-side adaptation," in *Web Engineering*. Springer, 2004, pp. 201–214.
- [278] G.M. Kapitsaki, D.A. Kateros, G.N. Prezerakos, and I.S. Venieris, "Model-driven development of composite context-aware web applications," *Information and Software technology*, Vol. 51, No. 8, 2009, pp. 1244–1260.
- [279] S. Ceri, F. Daniel, M. Matera, and F.M. Facca, "Model-driven development of context-aware web applications," *ACM Transactions on Internet Technology (TOIT)*, Vol. 7, No. 1, 2007, p. 2.
- [280] G.M. Kapitsaki and I.S. Venieris, "Model-driven development of context-aware web applications based on a web service context management architecture," in *Models in Software Engineering*. Springer, 2009, pp. 343–355.
- [281] N.M. Vergara, J.M.T. Linero, and A.V. Moreno, "Model-driven component adaptation in the context of web engineering," *European Journal of Information Systems*, Vol. 16, No. 4, 2007, pp. 448–459.
- [282] N. Koch, "Classification of model transformation techniques used in UML-based web engineering," *IET software*, Vol. 1, No. 3, 2007, pp. 98–111.

- [283] M.L. Bernardi, M. Cimitile, G.D. Lucca, and F.M. Maggi, "Development of flexible process-centric web applications: An integrated model driven approach," in *14th IEEE International Symposium on Web Systems Evolution (WSE)*. IEEE, 2012, pp. 67–71.
- [284] M. Lenk, A. Vitzthum, and B. Jung, "Model-driven iterative development of 3D web-applications using SSIML, X3D and JavaScript," in *Proceedings of the 17th International Conference on 3D Web Technology*. ACM, 2012, pp. 161–169.
- [285] L. Baresi, P. Fraternali, M. Tisi, and S. Morasca, "Towards model-driven testing of a web application generator," in *Web Engineering*. Springer, 2005, pp. 75–86.
- [286] R. Gitzel, A. Korthaus, and M. Schader, "Using established web engineering knowledge in model-driven approaches," *Science of Computer Programming*, Vol. 66, No. 2, 2007, pp. 105–124.
- [287] C. Kroiss, N. Koch, and A. Knapp, "UWE4JSF: A model-driven generation approach for web applications," in *ICWE*, Vol. 5648. Springer, 2009, pp. 493–496.
- [288] D. Clowes, D. Kolovos, C. Holmes, L. Rose, R. Paige, J. Johnson, R. Dawson, and S. Proberts, "A reflective approach to model-driven web engineering," in *Modelling Foundations and Applications*. Springer, 2010, pp. 62–73.
- [289] Y. Martínez, C. Cachero, and S. Meliá, "Evaluating the impact of a model-driven web engineering approach on the productivity and the satisfaction of software development teams," in *Web Engineering*. Springer, 2012, pp. 223–237.
- [290] A. Vallecillo, N. Koch, C. Cachero Castro, S. Comai, P. Fraternali, I. Garrigós Fernández, J. Gómez Ortega, G. Kappel, A. Knapp, M. Matera *et al.*, "MDWEnet: A practical approach to achieving interoperability of model-driven web engineering methods," 2007.
- [291] S. Meliá and J. Gomez, "The webSA approach: Applying model driven engineering to web applications," *Journal of Web Engineering*, Vol. 5, No. 2, 2006, pp. 121–149.
- [292] A.G. Cuesta and R.V. Granja, Juan Carlos and OConnor, "A model driven architecture approach to web development," in *Software and Data Technologies*. Springer, 2009, pp. 101–113.
- [293] A.J. Berre, "An agile model-based framework for service innovation for the future internet," in *Current Trends in Web Engineering*. Springer, 2012, pp. 1–4.
- [294] X. Liang, I. Marmaridis, and A. Ginige, "Facilitating agile model driven development and end-user development for EvolvingWeb-based workflow applications," in *IEEE International Conference on e-Business Engineering, ICEBE*. IEEE, 2007, pp. 231–238.
- [295] J. Grigera, J.M. Rivero, E.R. Luna, F. Giacosa, and G. Rossi, "From requirements to web applications in an agile model-driven approach," in *Web Engineering*. Springer, 2012, pp. 200–214.
- [296] J.M. Rivero, J. Grigera, G. Rossi, E.R. Luna, and N. Koch, "Improving agility in model-driven web engineering," in *CAiSE Forum*, Vol. 734, 2011, pp. 163–170.
- [297] J.M. Rivero, J. Grigera, G. Rossi, E.R. Luna, and N. Koch, "Towards agile model-driven web engineering," in *IS Olympics: Information Systems in a Diverse World*. Springer, 2012, pp. 142–155.
- [298] J.M. Rivero and G. Rossi, "MockupDD: Facilitating agile support for model-driven web engineering," in *Current Trends in Web Engineering*. Springer, 2013, pp. 325–329.
- [299] M.A. Bochicchio and E.A. Longo, "Integrating web systems design and business process modeling," in *Workshop on Model-driven Web Engineering*, 2005, p. 60.
- [300] M.D. Jacyntho and D. Schwabe, *Models and meta models for transactions in web applications*. Springer, 2010.
- [301] A. Ruokonen, L. Pajunen, and T. Systa, "On model-driven development of mobile business processes," in *Sixth International Conference on Software Engineering Research, Management and Applications, SERA'08*. IEEE, 2008, pp. 59–66.
- [302] M. Brambilla, S. Butti, and P. Fraternali, *Webratio bpm: a tool for designing and deploying business processes on the web*. Springer, 2010.
- [303] F. Trias, "Building CMS-based web applications using a model-driven approach," in *Sixth International Conference on Research Challenges in Information Science (RCIS)*. IEEE, 2012, pp. 1–6.
- [304] J. de Sousa Saraiva and A.R. da Silva, "CMS-based web-application development using model-driven languages," in *Fourth International Conference on Software Engineering Advances, ICSEA'09*. IEEE, 2009, pp. 21–26.
- [305] L. Luinenburg, S. Jansen, J. Souer, I. Van De Weerd, and S. Brinkkemper, "Designing web content management systems using the method association approach," in *Proceedings of the 4th International Workshop*

- on *Model-Driven Web Engineering (MDWE 2008)*, 2008, pp. 106–120.
- [306] J.d.S. Saraiva and A.R.d. Silva, “Development of CMS-based web-applications using a model-driven approach,” in *Proceedings of the 2009 Fourth International Conference on Software Engineering Advances*. IEEE Computer Society, 2009, pp. 500–505.
- [307] K. Vlaanderen, F. Valverde, and O. Pastor, “Model-driven web engineering in the CMS domain: A preliminary research applying SME,” in *Enterprise Information Systems*. Springer, 2009, pp. 226–237.
- [308] J. Souer, T. Kupers, R. Helms, and S. Brinkkemper, *Model-driven web engineering for the automated configuration of web content management systems*. Springer, 2009.
- [309] J. Souer and T. Kupers, “Towards a pragmatic model driven engineering approach for the development of CMS-based web applications,” in *Proceedings of the 5th Model Driven Web Engineering Workshop (MDWE09)*, 2009, pp. 31–45.
- [310] S. Martínez, J. Garcia-Alfaro, F. Cuppens, N. Cuppens-Boulahia, and J. Cabot, “Towards an access-control metamodel for web content management systems,” in *Current Trends in Web Engineering*. Springer, 2013, pp. 148–155.
- [311] F. Trias, V. de Castro, M. López-Sanz, and E. Marcos, “Reverse engineering applied to CMS-based web applications coded in PHP: A proposal of migration,” in *Evaluation of Novel Approaches to Software Engineering*. Springer, 2013, pp. 241–256.
- [312] A. Adamkó and L. Kollár, “Interoperability of model-driven web engineering approaches,” *8th International Conference on Applied Informatics*, Vol. 2, January 2010, pp. 295–303.
- [313] J.M. Vara, M.V. De Castro, M. Didonet Del Fabro, and E. Marcos, “Using weaving models to automate model-driven web engineering proposals,” *International Journal of Computer Applications in Technology*, Vol. 39, No. 4, 2010, pp. 245–252.
- [314] A. Cichetti, D. Di Ruscio, L. Iovino, and A. Pierantonio, “Managing the evolution of data-intensive web applications by model-driven techniques,” *Software & Systems Modeling*, Vol. 12, No. 1, 2013, pp. 53–83.



**e-Informatica Software Engineering Journal** (EISEJ) is an international, open access, peer-reviewed journal that concerns theoretical and practical issues pertaining development of software systems. Our aim is to focus on experimentation and data mining in software engineering.

The purpose of **e-Informatica Software Engineering Journal** is to publish original and significant results in all areas of software engineering research.

The scope of **e-Informatica Software Engineering Journal** includes methodologies, practices, architectures, technologies and tools used in processes along the software development lifecycle, but particular stress is laid on empirical evaluation.

**e-Informatica Software Engineering Journal** is published online and in hard copy form. The online version (which is our primary version) is open access, which means it is available at no charge to the public.

Topics of interest include, but are not restricted to:

- 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
- Meta-analyses
- 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 mining in software engineering
- Prediction models in software engineering
- Tools for software researchers or practitioners
- Project management
- Software products and process improvement and measurement programs
- Process maturity models
- Search-based software engineering

Papers can be rejected administratively without undergoing review for a variety reasons, such as being out of scope, being badly presented to such an extent as to prevent review, missing some fundamental components of research such as the articulation of a research problem, a clear statement of the contribution and research methods via structured abstract or the evaluation of the proposed solution (empirical evaluation is strongly suggested).

The submissions will be accepted for publication on the base of positive reviews done by international Editorial Board and external reviewers.

English is the only accepted publication language. To submit an article please enter our online paper submission site.

Subsequent issues of the journal will appear continuously according to the reviewed and accepted submissions.

<http://www.e-informatyka.pl/wiki/e-Informatica>



**e-Informatica**

ISSN 1897-7979