e-Informatica Software Engineering Journal, Volume 2, Issue 1, 2008

Integrating Human Judgment and Data Analysis to Identify Factors Influencing Software Development Productivity

Adam Trendowicz*, Michael Ochs*, Axel Wickenkamp*, Jürgen Münch*, Yasushi Ishigai**, Takashi Kawaguchi***

*Fraunhofer Institute for Experimental Software Engineering (Germany)
**Information-Technology Promotion Agency, Software Engineering Center (Japan)
***Toshiba Information Systems Corporation (Japan)

adam.trendowicz@iese.fraunhofer.de, michael.ochs@iese.fraunhofer.de, axel.wickenkamp@iese.fraunhofer.de, juergen.muench@iese.fraunhofer.de, ishigai@ipa.go.jp, kawa@tjsys.co.jp

Abstract

Managing software development productivity and effort are key issues in software organizations. Identifying the most relevant factors influencing project performance is essential for implementing business strategies by selecting and adjusting proper improvement activities. There is, however, a large number of potential influencing factors. This paper proposes a novel approach for identifying the most relevant factors influencing software development productivity. The method elicits relevant factors by integrating data analysis and expert judgment approaches by means of a multi-criteria decision support technique. Empirical evaluation of the method in an industrial context has indicated that it delivers a different set of factors compared to individual data- and expert-based factor selection methods. Moreover, application of the integrated method significantly improves the performance of effort estimation in terms of accuracy and precision. Finally, the study did not replicate the observation of similar investigations regarding improved estimation performance on the factor sets reduced by a data-based selection method.

1 Introduction

Many software organizations are still proposing unrealistic software costs, work within tight schedules, and finish their projects behind schedule and budget, or do not complete them at all [12]. This illustrates that reliable methods for managing software development effort and productivity are a key issue in software organizations.

At the same time, software cost estimation is considered to be more difficult than cost estimation in other industries. This is mainly due to the fact that software organizations typically develop new products as opposed to fabricating the same product over and over again. Moreover, software development is a human-based activity with extreme uncertainties from the outset. This leads to many difficulties in cost estimation, especially during early project phases. To address these difficulties, considerable research has been directed

at gaining a better understanding of the software development processes, and at building and evaluating software cost estimation techniques, methods, and tools [7, 34].

One essential aspect when managing development effort and productivity is the large number of associated and unknown influencing factors (so-called *productivity factors*) [33]. Identifying the right productivity factors increases the effectiveness of productivity improvement strategies by concentrating management activities directly on those development processes that have the greatest impact on productivity. On the other hand, focusing measurement activities on a limited number of the most relevant factors (goal-oriented measurement) reduces the cost of quantitative project management (collecting, analyzing, and maintaining the data). The computational complexity of numerous quantitative methods grows exponentially with the number of input factors [6], which significantly restricts their acceptance in industry.

In practice, two strategies for identifying relevant productivity factors, promoted in the related literature, are widely applied. In *expert-based* approaches, one or more software experts decide about a factor's relevancy [33]. In *data-based* approaches, existing measurement data covering a certain initial set of factors are analyzed in order to identify a subset of factors relevant with respect to a certain criterion [8, 15]. These factor selection strategies have, however, significant practical limitations when applied individually. Experts usually base their decisions on subjective preferences and experiences. In consequence, they tend to disagree by a wide margin and omit relevant factors while selecting irrelevant ones [33]. The effectiveness of data-based methods, on the other hand, largely depends on the quantity and quality of available data. They cannot, for instance, identify a relevant factor if it is not present in the initial set of factors contained by the underlying data set. Moreover, data analysis techniques are usually sensitive to messy (incomplete and inconsistent) data. Yet, assuring that all relevant factors are covered by a sufficient quantity of high-quality measurement data is simply not feasible in practice.

In this paper, we propose an integrated approach to selecting relevant productivity factors for the purpose of software effort estimation. We combine expert- with data-based factor selection methods, using a novel multi-criteria decision aid method called *AvalOn*. The presented approach is then evaluated in the context of a large software organization.

The remainder of the paper is organized as follows. Section 2 provides an overview of factor selection methods. Next, in Section 3, we present the integrated factor selection method, followed by the design of its empirical evaluation (Section 4) and an analysis of the results (Section 5). The paper ends with conclusions (Section 6) and further work perspectives (Section 7).

2 Related Work

2.1 Introduction to Factor Selection

Factor selection can be defined as a process that chooses a subset of M factors from the original space of N factors ($M \leq N$), so that the factor space is optimally reduced according to a certain criterion. In principle, selection process may be based on data analysis, expert' assessments, or both experts and data.

In *expert-based factor selection*, the factor space is practically infinite and not know a priori. There are, in principle, three major types of expert-based factor selection. In the most basic case, experts simply identify a set of most relevant factors without distinguishing the relevance level (factor selection). In addition to selecting the most relevant factors, experts may rank factors (provide order) with respect to their relevancy (factor ranking). Such a ranking does not, however, provide information about the relative distance between certain factors with respect to their relevancy. The most informative way of selecting factors is to quantify their relevancy on the ratio or interval scale. The most common approach is to define factor relevancy on the Likert scale [31] and ask experts to quantify these factors accordingly. In general, quantification could also be done on more than one criterion. In order to select the most relevant factors, quantifications on various criteria have to be aggregated.

In data-based factor selection, the factor space is given a priori and limited by the available measurement data. The data are analyzed in order to identify a limited set of the most relevant factors. Most of the dedicated data-based factor selection methods belong to one of the major areas of the data mining domain, where they are a subclass of the general problem of dimensionality reduction [21] (Figure 1).

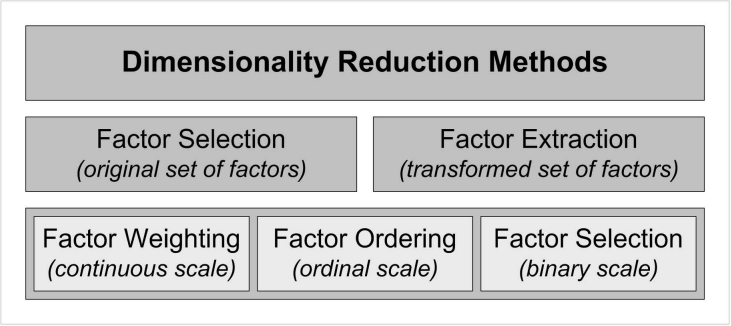

Figure 1: Dimensionality reduction methods

The purpose of dimensionality reduction methods is to reduce a potentially large number of cases and attributes (dimensions) in data in order to limit data noise and the computational time of the analysis. Since the computational complexity of numerous data mining algorithms grows exponentially with the number of dimensions (so-called NP-complete problems), dimensionality reduction allows applying them in practice (they finish within a reasonable amount of time). In this paper, we focus on reducing the number of attributes.

Attribute-oriented dimensionality reduction methods either extract or select factors based on an initial set. *Factor extraction* is a process that extracts a set of M new factors from the original N factors ($M < N$) through some functional mapping (e.g., sum or product). An example factor extraction method is Principal Component Analysis (PCA), which creates new factors as linear transformation of the initial factors. *Factor selection* is a process that extracts a set of M original factors from the original N factors ($M \leq N$).

Moreover, besides a simple subset of factors (factor selection), dimensionality reduction methods may provide an ordered (factor ranking) or weighted (factor weighting) set of factors. *Factor ranking* orders factors with respect to their relevancy, however, no information regarding the distance in rank between subsequent factors is provided (ordinal instead of interval scale).

Factor weighting methods represent the most robust dimensionality reduction approach. They quantify the relative relevance of each i-th factor f_i by providing a ratio-scale weight $w(f_i)$, where usually $w \in [0, 1]$. In this context, factor ranking is weighting on an integer scale (a factor's weight represents its rank) and factor selection is weighting on a dichotomous 0-1 scale (a factor's weight represent its selection or exclusion).

In general, a dimensionality reduction algorithm consists of four basic steps (Figure 2): subset generation, subset evaluation, stopping criterion, and result validation [10].

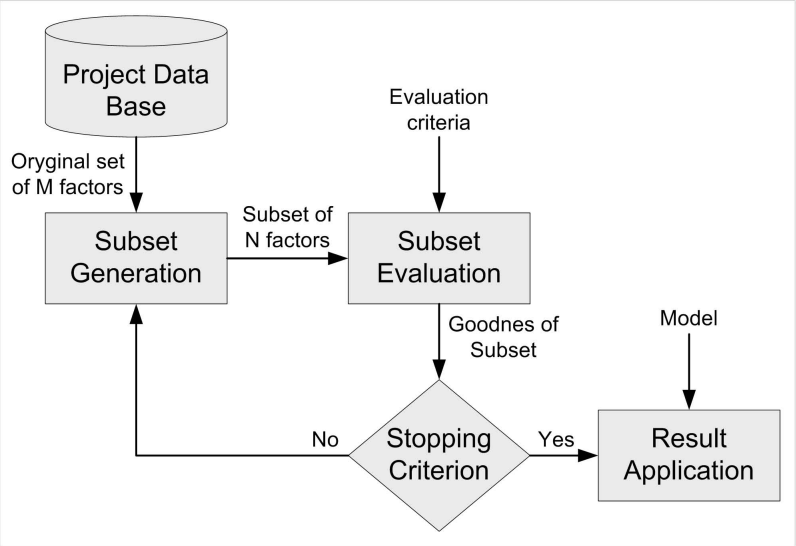

Figure 2: General dimensionality reduction process

Subset generation refers to a search (factor selection) or construction (factor extraction) procedure. Basically, it generates subsets of features for evaluation. In case of search procedures, there are various *directions* and *strategies* for searching through the factor space. The most popular search direction, greedy search, comes in three flavors (dependent on the starting point in the search space). Search can start with an empty set and add factors iteratively (*forward selection*) or it can start with a full set (N factors) and be reduced iteratively (*backward elimination*) until the factor subset meets a certain criterion. Hybrid, *bidirectional search* is also possible. Factors can also be searched randomly (*random search*).

The search strategy decides about the scope of the search. *Exhaustive* (*complete*) search performs a complete search through the factor space. Although it guarantees finding the optimal subset of factors, it is usually impractical due to the high computational costs (the problem is known to be NP-hard [11]). *Heuristic search*, as the name suggests, employs

heuristics in conducting the search. It avoids being complete, but at the same time risks losing optimal subsets. *Non-deterministic search*, unlike the first two types of strategies, searches for the next set at random (i.e., a current set does not directly grow or shrink from any previous set following a deterministic rule).

The selected subset of factors is always relative to a certain evaluation criterion. Evaluation criteria can be broadly categorized into two groups based on their dependency on the method applied on the selected factors.

In a *filter approach*, the goodness of a factor subset is evaluated according to criteria independent of the method that will later be applied on those factors (i.e., without the involvement of this method). The most common independent criteria are distance measure, information measure, dependency measure, consistency measure, and similarity measure. *Distance measures* are also known as separability, divergence, or discrimination measures. For a given objective factor Z, a factor X is preferred to another factor Y if X induces a greater difference between the conditional probabilities of Z's values than Y; if the difference is zero, then X and Y are indistinguishable. An example is the Euclidean distance measure. Information measures typically determine the information gain related to a certain factor. The information gain from a factor X is defined as the difference between the prior uncertainty and expected posterior uncertainty using X. Factor X is preferred to factor Y if the information gain from factor X is greater than that from factor Y. An example is the entropy measure [14]. *Dependency measures* or correlation measures qualify the ability to predict the value of one variable from the value of another. The coefficient is a classical dependency measure and can be used to find the correlation between a factor and an objective factor Z. If the correlation of factor X with Z is higher than the correlation of factor Y with Z, then factor X is preferred to Y. A slight variation of this is to determine the dependence of a factor on other factors; this value indicates the degree of redundancy of the factor. All evaluation functions based on dependency measures can be theoretically divided into distance and information measures, but they are usually kept as a separate category, because conceptually, they represent a different viewpoint. Finally, *consistency measures* are relatively new and have been in much focus recently. They rely heavily on the training dataset and the use of the Min-Factors bias in selecting a subset of factors. Min-Factors bias prefers consistent hypotheses definable over as few factors as possible. These measures find out the minimally sized subset that satisfies the acceptable inconsistency rate that is usually set by an expert.

In the *wrapper approach*, the goodness of the proposed factor subset is assessed by applying on it and evaluating the performance of the same method that will by applied on the subset selected eventually. In case of estimation, the well-known Mean Magnitude of Relative Error (MMRE) or the Prediction Level (Pred.25) [9] can be applied.

In an *embedded approach*, factor selection is a part of the learning (model building) process. One classical example is selecting factors along the paths of the Classification and Regression Tree (CART) model [5].

2.2 Factor Selection for Effort Estimation

The purpose of factor selection methods in software effort estimation is to reduce a large number of potential productivity factors (cost drivers) in order to improve estimation performance while maintaining (or reducing) estimation costs. Moreover, information on the most relevant influence factors may be used to guide measurement and improvement initiatives. In practice (authors' observation), relevant cost drivers are usually selected by experts and the selection process is often limited to uncritically adopting factors published in the related literature. Software practitioners adopt the complete effort model along with the integrated factors set (e.g., COCOMO [3]) or build their own model on factors adapted from an existing model. In both situations, they risk collecting a significant amount of potentially irrelevant data and getting limited performance of the resulting model.

Factors uncritically adopted from other contexts most often do not work well leading to much disappointment. They usually contain many irrelevant factors that do not contribute to explaining development productivity variance and increase the cost of data collection, analysis, and maintenance. On the other hand, even though context-specific factors are selected by experts, they tend to disagree largely with respect to the selected factors and their impact on productivity (relevancy).

During the last two decades, several data-based approaches have been proposed to support software organizations that are already collecting data on arbitrarily selected factors in selecting relevant factors. Various data analysis techniques were proposed to simply reduce the factor space by excluding potentially irrelevant factors (factor selection). The original version of the ANGEL tool [28] addressed the problem of optimal factor selection by exhaustive search. However, for larger factor spaces (>15-20), analysis becomes computationally intractable due to its exponential complexity. Alternative, less computationally expensive factor selection methods proposed in the literature include Principal Component Analysis (PCA) [32], Monte Carlo simulation (MC) [17], general linear models (GLM) [19], and wrapper factor selection [15, 8]. The latter approach was investigated using various evaluation models (e.g., regression [8], case-based reasoning [15]), and different search strategies (forward selection [8, 15], as well as random selection and sequential hill climbing [15]). In all studies, a significant reduction (by 50%-75%) of an initial factor set and improved estimation accuracy (by 15%-379%) were reported. Chen et al. [8] conclude, however, that despite substantial improvements in estimation accuracy, removing more than half of the factors might not be wise in practice, because it is not the only decision criterion.

An alternative strategy for removing irrelevant factors would be assigning weights according to a factor's relevancy (*factor weighting*). The advantage of such an approach is that factors are not automatically discarded and software practitioners obtain information on the relative importance of each factor, which they may use to decide about the selection/exclusion of certain factors. Auer et al. [2] propose an optimal weighting method in the context of the k-Nearest Neighbor (k-NN) effort estimator; however, exponential computational complexity limits its practical applicability for large factor spaces. Weighting in higher-dimensionality environments can be, for instance, performed using one of the

heuristics based on rough set analysis, proposed recently in [16]. Yet, their application requires additional overhead to discretize continuous variables.

A first attempt towards an integrated factor selection approach was presented in [4], where Bayesian analysis was used to combine the weights of COCOMO II factors based on human judgment and regression analysis. Yet, both methods were applied on sets of factors previously limited (arbitrarily) by an expert. Moreover, experts weighted factor relevancy on a continuous scale, which proved to be difficult in practice and may lead to unreliable results [35]. Most recently, Trendowicz et al. proposed an informal, integrated approach to selecting relevant productivity factors [35]. They used an analysis of existing project data in an interactive manner to support experts in identifying relevant factors for the purpose of effort estimation. Besides increased estimation performance, the factor selection contributed to increased understanding and improvement of software processes related to development productivity and cost.

3 An Integrated Factor Selection Method

In this paper, we propose an integrated method for selecting relevant productivity factors. The method employs a novel multi-criteria decision aid (MCDA) technique called *AvalOn* to combine the results of data- and expert-based factor selection.

3.1 Expert-based Factor Selection

Expert-based selection of relevant productivity factors is a two-stage process [36]. First, a set of candidate factors is proposed during a group meeting (brainstorming session). Next, factor relevancy criteria are identified and quantified on a Likert scale. Example criteria may include a factor's impact, difficulty, or controllability. *Impact* reflects the strength of a given factor's influence on productivity. *Difficulty* represents the cost of collecting factor-related project data. Finally, *controllability* represents the extent to which a software organization has an impact on the factor's value (e.g., a customer's characteristics are hardly controllable). Experts are then asked to individually evaluate the identified factors according to specified criteria and corresponding measurement scales. In a simple case (when expert involvement has to be limited due to related manpower costs), a simple factor's ranking with respect to its impact on cost (significance) might be performed.

3.2 Data-based Factor Selection

Data-based selection of relevant productivity factors employs one of the available factor weighting techniques. As compared to simple factor selection or ranking techniques, weighting provides experts with the relative distance between subsequent factors regarding their relevance. Selected weighting should be applicable to regression problems, i.e., to the continuous dependent variable (here: development productivity). We recommend excluding factor extraction as well as embedded and wrapper approaches. Factor extraction methods create new set of abstract factors that are not understandable by experts and require additional analysis to gain insight into the relationship between the original

factors. There are several arguments for preferring the filter strategy over the wrapper and embedded approaches. Filters (e.g., those based on mutual information criteria) provide a generic selection of variables, not tuned (biased) for/by a given learning machine. Moreover, filtering operates independently of the prediction method, reducing the number of features prior to estimation. Therefore, they can be used as a preprocessing step for reducing space dimensionality and overcoming over-fitting. Finally, filters tend to be far less computationally intensive than wrappers, which run the estimation method in each factor selection cycle.

As far as the search strategy is concerned, although forward selection (FSS) is computationally more efficient than backward selection (BSS), weaker subsets are found by FSS because the importance of variables is not assessed in the context of other variables not included yet [21]. A BSS method may outsmart FSS by eliminating, in the first step, the factor that by itself provides the best performance (explanation of productivity variance, effort estimation accuracy, etc.) in order to retain these two factors that together perform best. Still, if a very small set of optimal factors (e.g., single best factor) is preferred, or a huge set of initial factors has to be analyzed, BSS would probably be a better alternative [1]. Since software engineering data do not usually cover a large number of factors [35] and usually contains numerous interactions, the BSS strategy should be preferred. One may consider applying the optimal weighting approach as presented in [2]. Due to its significant computational complexity and optimal factor weighting, it might, however, not always be feasible (large factor spaces).

Given the size of the factor space, optimal weighting [2] (small size) or weighting heuristics [16] (large size) should be considered. In this paper, we employ the *Regression ReliefF* (RRF) technique [25]. RRF is well suited for software engineering data due to its robustness against sparse and noisy data. The output of RRF (weighting) reflects the ratio of change to productivity explained by the input factors.

3.3 An Integrated Factor Selection Method

Integrated factor selection combines the results of data- and expert-based selections by means of the AvalOn MCDA method. It is the hierarchically (tree) structured model that was originally used in COTS (Commercial-of-the-shelf) software selection [22, 24, 23]. AvalOn incorporates the benefits of a tree-structured MCDA model such as the Analytic Hierarchy Process (AHP) [26] and leverages the drawbacks of pair-wise (subjective) comparisons. It comprises, at the same time, subjective and objective measurement as well as the incorporation of uncertainty under statistical and simulation aspects. In contrast to the AHP model, which only knows one node type, it distinguishes several node types representing different types of information and offering a variety of possibilities to process data. Furthermore, AvalOn offers a weight rebalancing algorithm mitigating typical hierarchy-based difficulties originating from the respective tree structure. Finally, it allows for any modification (add, delete) of the set of alternatives while maintaining consistency in the preference ranking of the alternatives.

3.3.1 Mathematical background of the AvalOn method

As in many MCDA settings [36], a preference among the alternatives is processed by summing up $weight \cdot preference$ of an alternative. In AvalOn (3), this is accomplished for the node types *root*, *directory*, and *criterion* by deploying the following abstract model in line with the meta-model structure presented in Equation 1.

$$\sum_{j \in subnodes(i)} w_j \cdot pref_j(a) \tag{1}$$

where i is a node in the hierarchy, a the alternative under analysis, $subnodes(i)$ the set of child/sub-nodes of node i, $pref_j(a) \in [0..1]$ the preference of a in subnode j, and $w_j \in [0..1]$ the weight of subnode j. Hence $pref_i(a) \in [0..1]$.

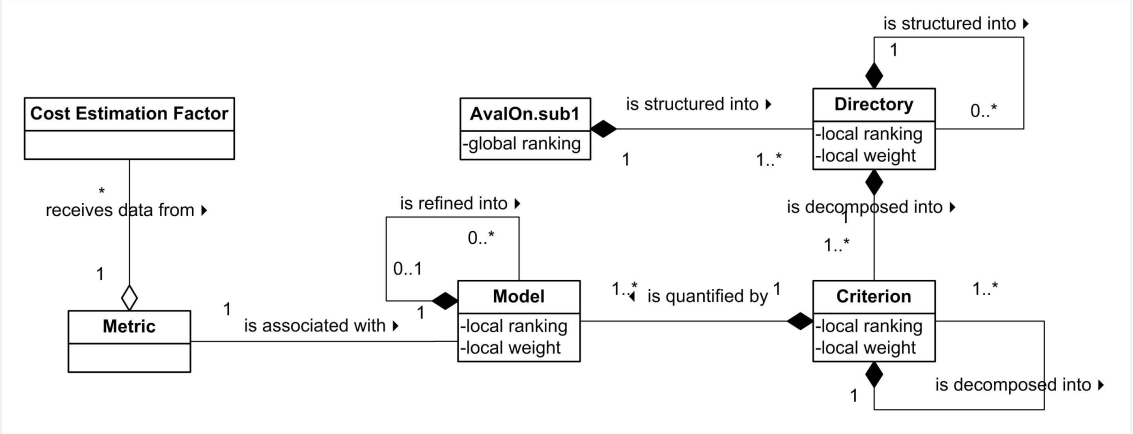

Figure 3: Meta-model for factor selection

In each model, a value function (*val*) is defined, building the relation between data from {*metrics x alternatives*} and the assigned preference values. *val* may be defined almost in an arbitrary way, i.e., it allows for preference mappings of metric scaled data as well as categorical data.

In this way, *val* can model the whole range of scales from semantic differential via Likert to agreement scales. Please note that when calculating $pref_i(a)$ on the lowest criterion level, the direct outputs of the function *val* in the subnodes, which are *models* in this case, are weighted and aggregated. The full details of the general model definition for *val* is described in [27]. In this context, two examples for *val*, one metric scaled (figure on the left) and one categorical (figure on the right), are given in Figure 4. On the x-axis, there are the input values of the respective metric, while the y-axis shows the individual preference output *val*. A full description of the AvalOn method can be found in [27, 24].

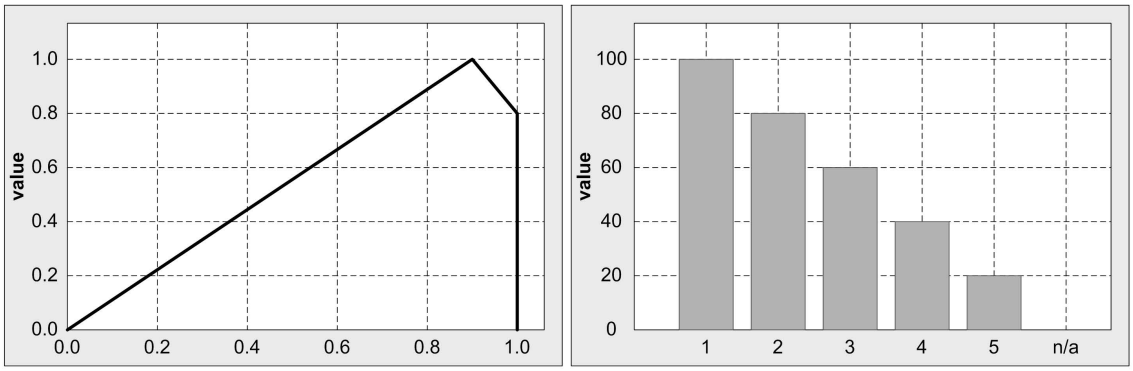

Figure 4: Example *val* models

3.3.2 Application of the AvalOn method

AvalOn allows for structuring complex information into groups (element *directory* in the meta-model) and criteria (element *criterion* in the meta-model). Each directory as well as each criterion may be refined into sub-directories and sub-criteria. Each (sub)-criterion may then be refined into individual model(s) and sub-models. The models transform the measurement data coming from each alternative into initial preference values. The models providing the preferences based on each measurement by alternative are associated with a set of previously defined metrics. Bottom-up, the data coming from each alternative for potential selection (here: *productivity factors*) are then processed through the models and aggregated from there into the criteria and directory level(s). Finally, in the root node (here: *AvalOn.sub1*), the overall preference of the productivity factors based on their data about individual metrics is aggregated using a weighting scheme that is also spread hierarchically across the tree of decision (selection) criteria. The hybrid character of the setting in this paper can be modeled by combining expert opinion and objective data from, e.g., preliminary data analyses, into criteria and models within different directories, and defining an adequate weighting scheme.

3.3.3 Result Views of the AvalOn Meta-Model

The AvalOn meta-model offers a tool-based variety of result views of the preferences of the alternatives [27]. Major views are available for every node of type *root*, *directory*, *criterion*, and *model* in the criteria hierarchy. In detail views are: (i) overall preference (Figure 5), (ii) node and subnode preference profile (Figure 6), (iii) overall preference range/uncertainty (Figure 7), (iv) preference weight sensitivity (Figure 8), and (v) pairwise preference comparability (Figure 9).

The *overall preference* view plainly shows the total preference of the alternatives in a selected node of the hierarchy. The *node and subnode preference profile* visualizes the preferences of the alternatives in the selected node on the left line and the individual preferences of the alternatives in the subnodes of the selected node on the lines to the right hand side of it.

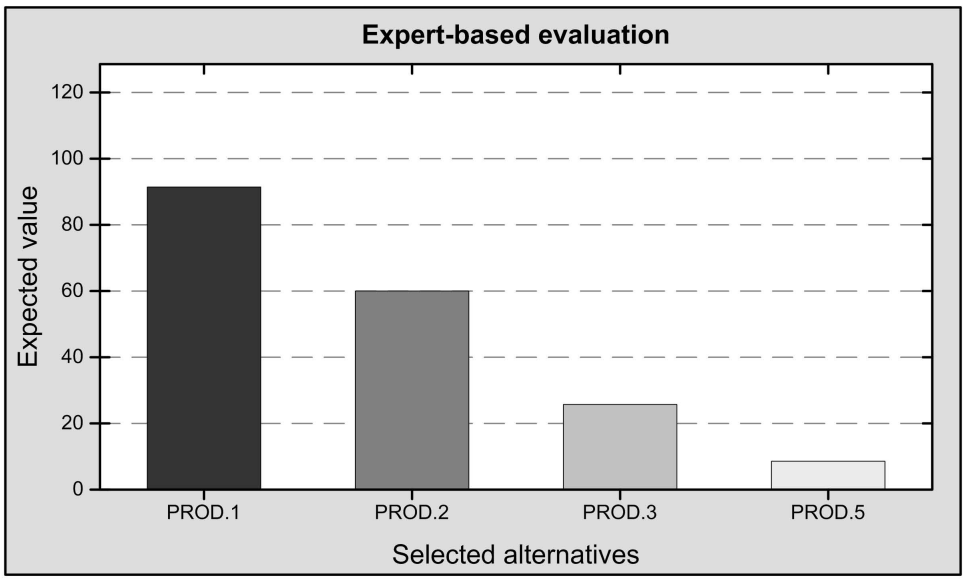

Figure 5: Overall preference

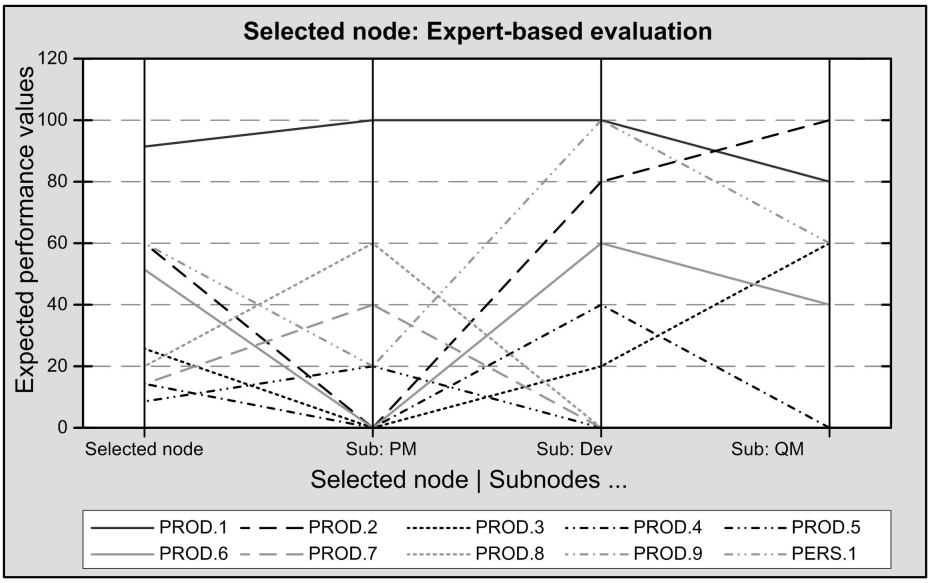

Figure 6: Node and subnode preference profile

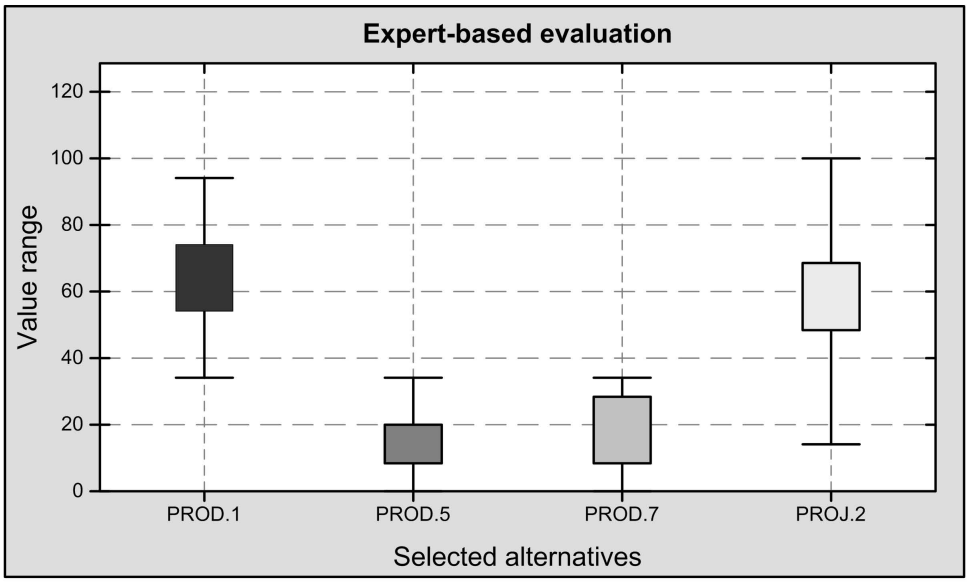

Figure 7: Overall preference range/uncertainty

The *overall preference range (uncertainty)* view provides graphical information about the variability range of the preferences of the alternatives in a selected node of the hierarchy when evaluation and decision have to be made under uncertainty. In addition, a t-test [29] can be performed in order to verify or falsify whether two alternatives do have a significant difference in their individual preferences. The *weight sensitivity* of the *preference* of the alternatives in the selected node analyzes the change of the total preferences of the alternatives when changing the current weight (marked by the red vertical line) of one specific subnode. Whenever lines of alternatives intersect at a specific point they are of identical preference, and by continuing in the change direction of the subnode weight, the preference between the alternatives will be changed (for two alternatives: turned around).

The pairwise preference comparability view shows for each pair of alternatives - in a direct comparison - whether one of the alternatives is to be preferred over the other (green areas), whether the two alternatives are indifferent (white area), or whether they are incomparable (yellow area). This method deploys ORESTE+ [27], which is a metric relaxation of the ordinal ORESTE procedure.

4 Empirical Study

The integrated factor selection method proposed in this paper was evaluated in an industrial context. We applied the method for the purpose of software effort estimation and compared it with isolated expert- and data-based selection methods. Data-based factor selection employed the RRF technique [25] implemented in the WEKA data mining software [37]. Expert-based factor selection was performed as a multiple-expert ranking (see Section 3.1 regarding the ranking process).

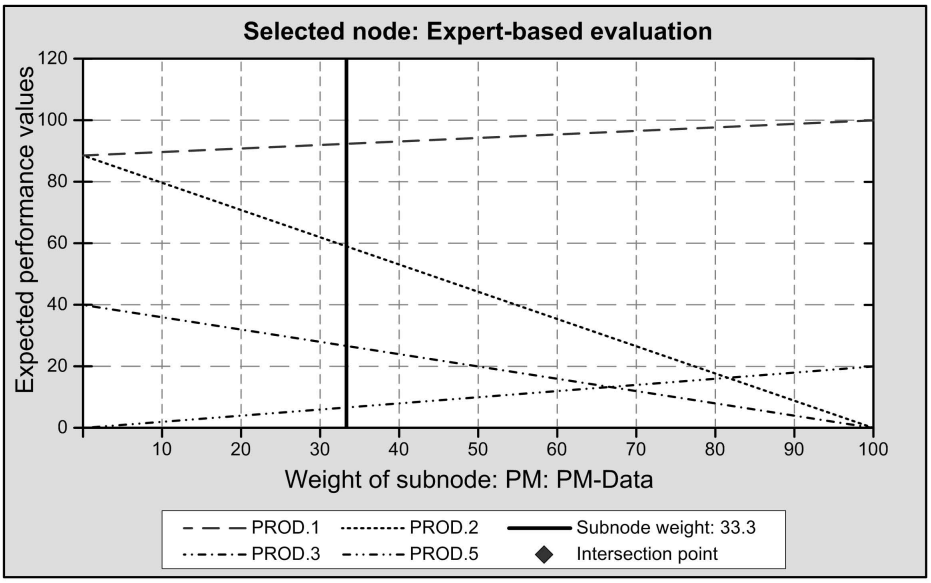

Figure 8: Preference weight-sensitivity

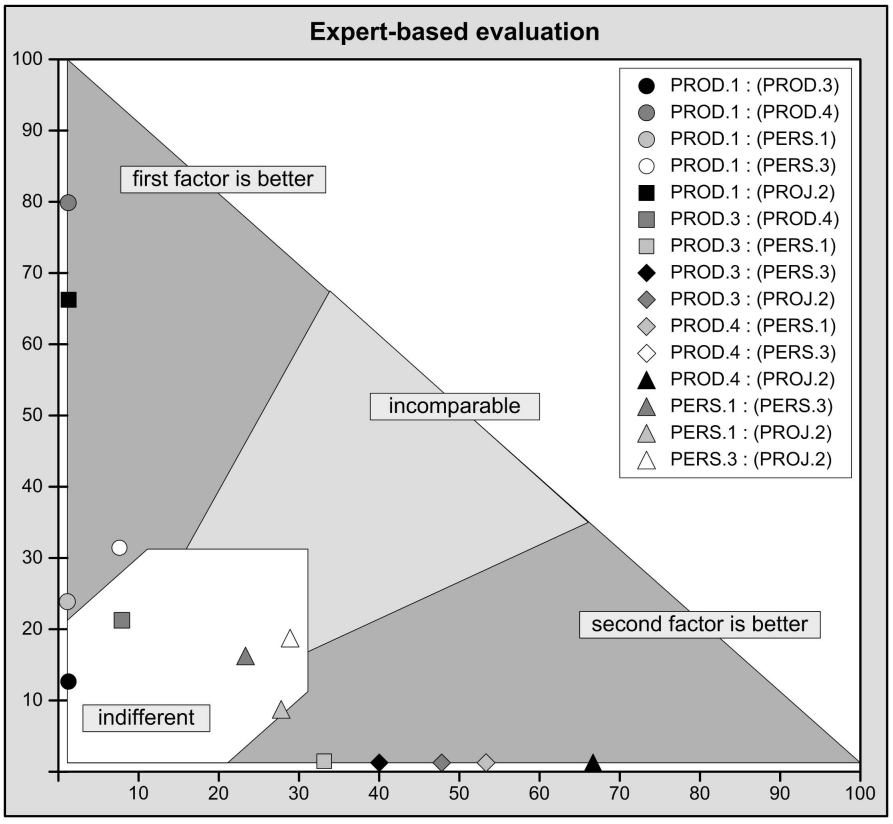

Figure 9: Pairwise preference comparability

4.1 Study Objectives and Hypotheses

The objective was to evaluate, in a comparative study, expert- and data-based approaches and the integrated approach for selecting the most relevant productivity factors in the context of software effort estimation. For that purpose, we defined two research questions and related hypotheses:

Q1. *Do different selection methods provide different sets of productivity factors?*

> **H1.** Expert-based, data-based, and integrated methods select different (probably partially overlapping) sets of factors.

Q2. *Which method (including not reducing factors at all) provides the better set of factors for the purpose of effort estimation?*

> **H2.**The integrated approach provides a set of factors that ensure higher performance of effort estimation than factors provided by expert- and data-based selection approaches when applied individually.

Some effort estimation methods such as stepwise regression [9] or OSR [8] already include embedded mechanisms for selecting relevant productivity factors. In our study, we wanted to evaluate in addition how preliminary factor selection done by an independent method influences the performance of such estimation methods. This leads us to a general research question:

Q3. *Does application of an independent factor selection method increase the prediction performance of an estimation method that already has an embedded factor selection mechanism?*
Answering such a generic question would require evaluating all possible estimation methods. This, however, is beyond the scope of this study. We limit our investigation to the OSR estimation method [8] and define a corresponding research hypothesis:

> **H3.** Application of an independent factor selection method does not increase the prediction performance of the OSR method.

Finally, in order to validate and replicate the results of the most recent research regarding the application of data-based factor selection to analogy-based effort estimation (e.g., [8, 15]), we define the following question:

Q4. *Does application of a data-based factor selection method increase the prediction performance of an analogy estimation method?*

> **H4.** Application of a data-based factor selection method increases the prediction performance of a k-NN estimation method.

4.2 Study Context and Empirical Data

The empirical evaluation was performed in the context of Toshiba Information Systems (Japan) Corporation (TJSYS). The project measurement data repository contained a total of 76 projects from the information systems domain. Figure 10 illustrates the variance of development productivity measured as function points (unadjusted, IFPUG) per man-month.

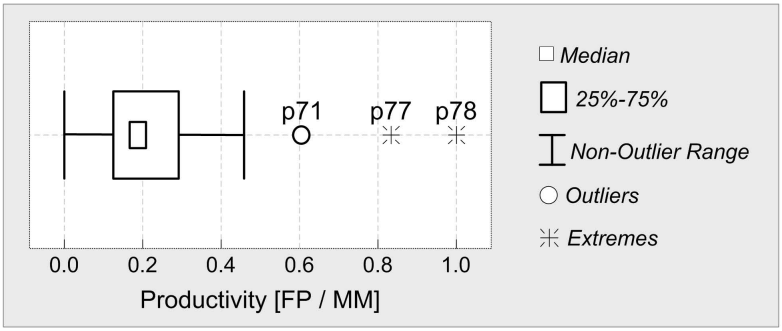

Figure 10: Development productivity variance; data presented in a normalized form

Expert assessments regarding the most relevant factors were obtained from three experts (see Table 1). During the group meeting (brainstorming session), an initial set of

	Expert 1	**Expert 2**	**Expert 3**
Position/Role	Project manager	Developer	Quality manager
Experience [#working years]	8	15	3
Experience [#performed projects]	30	15	40

Table 1: Experts who participated in the study

factors was identified. It was then grouped into project-, process, personnel-, and product-related factors as well as context factors. The first four groups refer to the characteristics of the respective entities (software project, development process, products, and stakeholders). The latter group covers factors commonly used to limit the context of software effort estimation or productivity modeling. The *application domain*, for instance, is often regarded as a context factor, i.e., an effort model is built for a specific application domain. Finally, experts were asked to select the 5 most important factors from each category and rank them from most relevant (rank = 1) to least relevant (rank = 5).

4.3 Study Limitation

Unfortunately, the measurement repository available did not cover all relevant factors selected by the experts. It was also not possible to collect the data ex post facto. This prevented us from doing a full comparative evaluation of the three factor selection methods considered here for the purpose of software effort estimation. In order to at least get

an indication of the methods' performance, we decided to compare them (instead of all identified factors) on the factors identified by experts for which measurement data were available. This would represent the situation where those factors cover all factors available in the repository and identified by experts.

4.4 Study Design and Execution

4.4.1 Data Preprocessing

Measurement data available in the study suffered from incompleteness (44.3% missing data). An initial preprocessing was thus required in order to apply the data analysis techniques selected in the study. We wanted to avoid using simple approaches to handling missing data such as list-wise deletion or mean imputation, which significantly reduce data quantity and increase noise. Therefore, we decided to apply the *k-Nearest Neighbor* (*k-NN*) imputation method. It is a common *hot deck* method, in which k nearest projects minimizing a certain similarity measure (calculated on non-missing factors) are selected to impute missing data. It also proved to provide relatively good results when applied to sparse data in the context of software effort prediction [20]. Moreover, other more sophisticated (and potentially more effective) imputation methods required removing factor collinearities beforehand. Such a preprocessing step would, however, already be a kind of factor selection and might thus bias the results of the actual factor selection experiment. We adopted the k-NN imputation approach presented in [13].

We assumed a *missing at random* (*MAR*) missingness mechanism, which means [18] that the cause of the missing data is completely unrelated to the missing values; it may be related to the observed values of other variables. This assumption is weaker than *missing completely at random* (*MCAR*); however, it is more realistic and seems not to have a significant impact on the accuracy of the k-NN imputation method [30].

In order to assure maximal performance of the imputation, before applying it, we removed factors and projects with a large missing data ratio so that the total ratio of missing data was reduced to around one third; however, with minimal loss of non-missing data. We applied the following procedure: We first removed factors where 90% of the data were missing and next, projects where more than 55% of the data were still missing. As a result, we reduced the total rate of missing data to 28.8%, while losing a minimal quantity of information (removed 19 out of 82 factors and 3 out of 78 projects). The remaining 28.8% of missing data were imputed using the k-NN imputation technique.

4.4.2 Empirical Evaluation

Let us first define the following abbreviations for the factor sets used in the study:

FM: factors covered by measurement data.
FM$_{\mathbf{R}}$: relevant FM factors selected by the RReliefF method (factors with $weight > 0$)
FM$_{\mathbf{R10}}$: the 10% most relevant FMR factors
FE: factors selected by experts
FI: factors selected by the integrated method

FT: all identified factors ($FM \cup FE$)
FC: factors selected by experts for which measurement data are available ($FM \cap FE$)
FC_{E25}: the 25% most relevant FC factors selected by experts
FC_{R25}: the 25% most relevant FC factors selected by the RRF method
FC_{I25}: the 25% most relevant FC factors selected by the integrated method

Hypothesis H1. In order to evaluate H1, we compared factor sets selected by the data-based, expert-based, and integrated methods (FM_R, FE, and FI). For the 10 most relevant factors shared by all three factor sets, we compared the ranking agreement using Kendall's coefficient of concordance [29].

Hypothesis H2. In order to evaluate H2, we evaluated the estimation performance of two data-based estimation methods: *k-Nearest Neighbor* (*k-NN*) [28] and *Optimized Set Reduction* (*OSR*) [6]. We applied them in a leave-one-out cross validation on the following factor sets: FM, FC, FC_{E25}, FC_{R25}, and FC_{I25}.

Hypothesis H3. In order to evaluate H3, we compared the estimation performance of OSR (which includes an embedded, data-based factor selection mechanism) when applied on the FM and FM_{R10} factor sets.

Hypothesis H4. In order to evaluate H4, we compared the estimation performance of the k-NN method when applied on the FM and FM_{R10} factor sets.

To quantify the estimation performance in H2, H3, and H4, we applied the common accuracy and precision measures defined in [9]: *magnitude of relative estimation error* (*MRE*), *mean* and *median of MRE* (*MMRE* and *MdMRE*), as well as *prediction at level 25%* (*Pred.25*). We also performed an *analysis of variance* (*ANOVA*) [29] of MRE to see if the error for one approach was statistically different from that of another one. We interpret the results as being statistically significant if the results could be due to chance less than 2% of the time ($p < 0.02$).

5 Results of the Empirical Study

Hypothesis H1*: Expert-based, data-based and integrated methods select different (probably partially overlapping) sets of factors.*
After excluding the dependent variable (development productivity) and project ID, the measurement repository contained data on 61 factors. Experts identified a total of 34 relevant factors, with only 18 of them being already measured (FC). The RRF method selected 40 factors (FM_R), 14 of which were also selected by experts. The integrated approach selected 59 factors in total, with only 14 being shared with the former two selection methods. Among the FC factors, as many as 8 were ranked by each method within the top 10 factors (Table 2). Among the top 25% FC factors selected by each method, only one factor was in common, namely *customer commitment and participation.*

There was no significant agreement (Kendall = 0.65 at p = 0.185) between data- and expert-based rankings on the FC factors. The integrated method introduced significant agreement on ranks produced by all three methods (Kendall = 0.72 at p = 0.004).

Interpretation (H1): Data- and expert-based selection methods provided different (partially overlapping) sets of relevant factors. Subjective evaluation of the shared factors suggests that both methods vary regarding the assigned factor's importance; yet this could not be confirmed by statistically significant results. The integrated method introduced a consensus between individual selections (significant agreement) and as such might be considered as a way to combine the knowledge gathered in experts' heads and in measurement data repositories.

Productivity factor	$\mathbf{FC_E}$	$\mathbf{FC_R}$	$\mathbf{FC_I}$
Customer commitment and participation	**3**	**3**	**3**
System configuration (e.g., client-server)	5	**2**	5
Application domain (e.g., telecommunication)	**1**	6	**1**
Development type (e.g., enhancement)	7	**1**	**4**
Application type (e.g., embedded)	**2**	7	**2**
Level of reuse	9	**4**	9
Required product quality	6	10	7
Peak team size	8	9	8

Table 2: Comparison of the ranks on FC factors (top 25% marked in bold)

Hypothesis H2*: The integrated approach provides a set of factors that ensure higher performance of effort estimation than factors provided by expert- and data-based selection approaches when applied individually.*

A subjective analysis of the estimates in Table 3 suggests that the k-NN provided improved estimates when applied on a reduced FC factors set (FC_{E25}, FC_{R25}, and FC_{I25}), whereas OSR does not consistently benefit from independent factor reduction (by improved estimates). The analysis of the MRE variance, however, showed that the only significant ($p = 0.016$) improvement in estimation performance of the k-NN predictor was caused by the integrated factor selection method. The OSR predictor improved its estimates significantly ($p < 0.02$) only on the FC_{E25} factors set.

Interpretation (H2): The results obtained indicate that a factor set reduced through an integrated selection contributes to improved effort estimates. Yet, this does not seem to depend on any specific way of integration. The k-NN predictor, which uses all input factors, improved on factors reduced by the AvalOn method. The OSR method, however, improved slightly on the factors reduced by experts. This interesting observation might be explained by the fact that OSR, which includes an embedded, data-based factor selection mechanism, combined this with prior expert-based factor selection. Still, the effectiveness of such an approach largely depends on the experts who determine (pre-select) input factors for OSR (expert-based selection is practically always granted higher priority).

Predictor	Factors Set	MMRE	MdMRE	Pred.25
k-NN	FM	73.7%	43.8%	21.3%
	FC	52.6%	40.0%	26.7%
	FC_{E25}	46.3%	38.5%	33.3%
	FC_{R25}	48.3%	36.9%	29.3%
	FC_{I25}	**47.5%**	**33.3%**	**30.7%**
OSR	FM	59.7%	50.8%	17.3%
	FC	65.9%	59.2%	18.7%
	FC_{E25}	**30.7%**	**57.9%**	**24.0%**
	FC_{R25}	66.2%	52.1%	14.7%
	FC_{I25}	65.1%	57.9%	14.7%

Table 3: Comparison of various factor selection methods

Hypothesis H3*: Application of an independent factor selection method does not increase the prediction performance of the OSR method.*
A subjective analysis of OSR's estimation error (Table 3 and Table 4) suggests that it performs generally worse when applied on the factors chosen by an independent selection method. This observation was, however, not supported by the analysis of the MRE variance. The exception was the FC set reduced by experts (FC_{E25}), on which a slight, statistically significant improvement of the OSR's predictions was observed.

Predictor	Factors Set	MMRE	MdMRE	Pred.25	ANOVA
k-NN	FM	73.7%	43.8%	21.3%	p = 0.39
	FM_{R10}	56.8%	40.7%	22.7%	
OSR	FM	59.7%	50.8%	17.3%	p = 0.90
	FM_{R10}	68.1%	59.1%	16.0%	

Table 4: Results of data-based factor selection

Interpretation (H3). The results obtained indicate that no general conclusion regarding the impact of independent factor selection on the prediction performance of OSR can be drawn. Since no significant deterioration of estimation performance was observed, application of OSR on the reduced set of factors can be considered useful due to the reduced cost of measurement. Yet, improving OSR's estimates might require a selection method that is more effective than the selection mechanism embedded in OSR.

Hypothesis H4*: Application of a data-based factor selection method increases the prediction performance of a k-NN estimation method.*
A subjective impression of improved estimates provided by the k-NN predictor (Table 4) when applied on the reduced factor set (FM_{R10}) was, however, not significant in the sense of different variances of MRE (p = 0.39). Yet, estimates provided by the k-NN predictor improved significantly when used on the FC data set reduced by the integrated selection

method (p = 0.016). The two individual selection methods did not significantly improve performance of the k-NN predictor.

Interpretation (H4): Although a subjective analysis of the results (Table 3 and Table 4) suggests improved estimates provided by the k-NN predictor when applied on reduced factors sets, no unambiguous conclusion can be drawn. The performance of k-NN improved significantly only when applied on factors identified from the FC set by the integrated selection method (the FC_{I25} set). This might indicate that k-NN's performance improvement depends on the applied factor selection method (here, the integrated method was the best one).

Threats to Validity

We have identified two major threats to validity that may limit the generalizability of the study results. First, the estimation performance results of the factor selection methods investigated, compared on the FC set, might not reflect their true characteristics, i.e., as compared on the complete set of identified factors (threat to hypothesis H2). Yet, a lack of measurement data prevented us from checking on this. Second, the RRF method includes the k-NN strategy to search through the factor space and iteratively modify factor weights. This might bias the results of k-NN-based estimation by contributing to better performance of k-NN (as compared to OSR) on factors selected by RRF (threat to hypotheses H3 and H4).

6 Summary

In this paper, we proposed an integrated approach for selecting relevant factors influencing software development productivity. We compared the approach in an empirical study against selected expert- and data-based factor selection approaches.

The investigation performed showed that expert- and data-based selection methods identified different (only partially overlapping) sets of relevant factors. The study indicated that the *AvalOn* method finds a consensus between factors identified by individual selection methods. It combines not only the sets of relevant factors, but also the individual relevancy levels of selected factors. We showed that in contrast to data- and expert-based factor selection methods, the integrated approach may significantly improve the estimation performance of estimation methods that do not include an embedded factor selection mechanism. Estimation methods that include such a mechanism may, however, benefit from integrating their capabilities with expert-based factor selection.

The study did not replicate the observation of similar investigations regarding improved estimation performance on the factor sets reduced by a data-based selection method. Neither of the estimation methods employed in the study (k-NN and OSR) improved significantly when applied on factor sets reduced by the *RReliefF* method. Although k-NN improved in terms of aggregated error measures (e.g., MMRE) the difference in the MRE variance was insignificant. The results obtained for the OSR method may indicate that the change of its prediction performance when applied on a reduced set of factors depends on the selection method used.

Finally, we also observed that the function point adjustment factor (FPAF) was not considered among the most relevant factors, although factor selection was driven by a variance on development productivity calculated from unadjusted function point size. Moreover, some of the factors considered as relevant (e.g., performance requirements) belong to components of the FPAF. This might indicate that less relevant sub-factors of the FPAF and/or the adjustment procedure itself may hide the impact of relevant factors. Considering sub-factors of FPAF individually might therefore be more beneficial.

In conclusion, factor selection should be considered as an important aspect of software development management. Since individual selection strategies seem to provide inconsistent results, integrated approaches should be investigated to support software practitioners in limiting the cost of management (data collection and analysis) and increasing the benefits (understanding and improvement of development processes).

7 Further Work

Further work shall focus on several aspects. First, a full evaluation of the three selection strategies presented on a complete data set (including data on all factors identified by experts) shall be performed.

Daily industrial practice requires an incremental approach to identify relevant productivity factors. After identifying a single most relevant factor or small group of (probably related) most relevant factors, corresponding project data should be collected in order to quantitatively validate the true impact on productivity. The identified factors may, for instance, be applied within an estimation model (such as CoBRA [35]) in order to see how much productivity variance they are able to explain across the considered development projects. This shall be the next subject of our further investigation.

Finally, methods for identifying and explicitly considering factor dependencies need to be investigated. Such information might not only improve performance in effort estimation and productivity modeling, but they also improve understanding of the interaction between organizational processes influencing development productivity.

Acknowledgments. We would like to thank Toshiba Information Systems (Japan) Corporation and all involved experts who greatly contributed to the study. We would also like to thank the Japanese Information-technology Promotion Agency for supporting the study. Finally, we would like to thank Sonnhild Namingha and Marcus Ciolkowski for reviewing the paper.

References

[1] D. Aha and R. L. Bankert. A comparative evaluation of sequential feature selection algorithms. In D. Fisher and H.-J. Lenz, editors, *Artificial Intelligence and Statistics*, chapter Chapter 4, pages 199–206. Springer-Verlag, 1996.

[2] M. Auer, A. Trendowicz, B. Graser, E. J. Haunschmid, and S. Biffl. Optimal project feature weights in analogy-based cost estimation: Improvement and limitations. *IEEE Transactions on Software Engineering*, 32(2):83–92, February 2006.

[3] B. W. Boehm. *Software Engineering Economics*. Prentice-Hall, Englewood Cliffs, NJ, 1981.

[4] B. W. Boehm, C. Abts, A. W. Brown, S. Chulani, B. K. Clark, E. Horowitz, R. Madachy, D. Reifer, and B. Steece. *Software Cost Estimation with CoCoMo II*. Prentice-Hall, 2000.

[5] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Advanced Books & Software. Chapman and Hall, New-York, 1984.

[6] L. Briand, V. Basili, and W. Thomas. A pattern recognition approach for software engineering data analysis. *IEEE Transactions on Software Engineering*, 18(1):931–942, November 1992.

[7] L. Briand and I. Wieczorek. Software resource estimation. In J. Marciniak, editor, *Encyclopedia of Software Engineering*, volume 2, pages 1160–1196. John Wiley & Sons, 2002.

[8] Z. Chen, T. Menzies, D. Port, and B. Boehm. Finding the right data for software cost modeling. *IEEE Software*, 22(6):38–46, November/December 2005.

[9] S. Conte, H. Dunsmore, and V. Shen. *Software Engineering Metrics and Models*. CA: Benjamin Cummings, 1986.

[10] M. Dash and H. Liu. Feature selection methods for classifications. *An International Journal on Intelligent Data Analysis*, 1(3):131–156, 1997.

[11] A. E. and V. Kann. On the approximation of mini-mizing non zero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, Volume 209(1-2):237–260, 1998.

[12] T. S. Group. Chaos chronicles. Technical report, The Standish Group, West Yarmouth, MA, 2003.

[13] P. Jönsson and C. Wohlin. An evaluation of k-nearest neighbour imputation using likert data. In *Proceedings of the 10th IEEE International Software Metrics Symposium*, pages 108–118. IEEE Computer Society, 2004.

[14] J.R.Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[15] C. Kirsopp, M. Shepperd, and J. Hart. Search heuristics, case-based reasoning and software project effort prediction. In *Proceedings of the Genetic and Evolutionary Computation Conference*, page 1367–1374, 2002.

[16] J. Li and G. Ruhe. A comparative study of attribute weighting heuristics for effort estimation by analogy. In *Proceedings of the International Symposium on Empirical Software Engineering*, page 66–74, 2006.

[17] T. Liang and A. Noore. Multistage software estimation. In *Proceedings of the 35th Southeastern Symposium on System Theory*, page 232–236, 2003.

[18] R. Little and D. Rubin. *Statistical Analysis with Missing Data*. John Wiley & Sons, New York, 2nd edition edition, 2002.

[19] K. Maxwell, L. V. Wassenhove, and S. Dutta. Software development productivity of european space, military, and industrial applications. *IEEE Transactions on Software Engineering*, 22(10):706–718, October 1996.

[20] I. Myrtveit, E. Stensrud, and U. Olsson. Analyzing data sets with missing data: An empirical evaluation of imputation methods and likelihood-based methods. *IEEE Transactions on Software Engineering*, 27(11):999–1013, November 2001.

[21] I. G. nad A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[22] M. Ochs. Using sw risk management for deriving method requirements for risk mitigation in cots assessment & selection. In *Proceeding of the International Conference on Software Engineering and Knowledge Engineering*, 2003.

[23] M. Ochs, D. Pfahl, G. Chrobok-Diening, and B. Nothhelfer-Kolb. A cots acquisition process: definition and application experience. In *Proceedings of the 11th European Software Control and Metrics Conference*, 2000.

[24] M. Ochs, D. Pfahl, G. Chrobok-Diening, and B. Nothhelfer-Kolb. A method for efficient measurement-based cots assessment & selection – method description and evaluation results. In *Proceedings of the 7th International Syposium on Software Metrics*, 2001.

[25] M. Robnik-Sikonja and I. Kononenko. Theoretical and empirical analysis of relieff and rrelieff. *The Machine Learning Journal*, 53:23–69, 2003.

[26] T. Saaty. *The Analytic Hierarchy Process*. McGraw-Hill, New York, 1990.

[27] D. Schillinger. Entwicklung eines simulationsfähigen cots assessment und selection tools auf basis eines für software adequaten hierarchischen mcdm meta modells, 2006. Supervisors: Prof. Dr. D.H. Rombach, M. Ochs.

[28] M. Shepperd and C. Schofield. Estimating software project effort using analogies. *IEEE Transactions on Software Engineering*, 23(12):736–743, November 1997.

[29] D. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman & Hall/CRC, 2nd edition edition, 2000.

[30] Q. Song and M. Shepperd. A short note on safest default missingness mechanism assumptions. *Empirical Software Engineering*, 10(2), 2005.

[31] P. Spector. *Summated Rating Scale Construction*. Sage Publications, 1992.

[32] G. Subramanian and S. Breslawski. Dimensionality reduction in software development effort estimation. *Journal of Systems and Software*, 21(2):187–196, 1993.

[33] A. Trendowicz. Factors influencing software development productivity - state of the art and industrial experiences. Technical Report 08.07/E, Fraunhofer IESE, Kaiserslautern, Germany, 2007.

[34] A. Trendowicz. Software effort estimation - overview of current industrial practices and existing methods. Technical Report 06.08/E, Fraunhofer IESE, Kaiserslautern, Germany, 2008.

[35] A. Trendowicz, J. Heidrich, J. Münch, Y. Ishigai, K. Yokoyama, and N. Kikuchi. Development of a hybrid cost estimation model in an iterative manner. In *Proceedings of the 28th International Conference on Software Engineering*, page 331–340, 2006.

[36] P. Vincke. *Multicriteria Decision-aid*. John Wiley & Sons, Chichester, 1992.

[37] I. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2005.