Information and Software Technology 80 (2016) 73–88

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Functional size approximation based on use-case names

Mirosław Ochodek*

Faculty of Computing, Institute of Computing Science, Poznan University of Technology, ul. Piotrowo 2, Poznań 60-965, Poland

ARTICLE INFO

Article history:
Received 25 April 2016
Revised 16 August 2016
Accepted 19 August 2016
Available online 21 August 2016

Keywords:
Functional size measurement
Approximate software sizing methods
COSMIC
IFPUG Function Point Analysis
Size estimation methods
Use cases

ABSTRACT

Context: Functional size measures, such as IFPUG Function Points or COSMIC, are widely used to support software development effort estimation. Unfortunately, applying the COSMIC or IFPUG Function Point Analysis methods at early stages of software development is difficult or even impossible because available functional requirements are imprecise. Moreover, the resources that could be allocated to perform such measurement are usually limited. Therefore, it is worth investigating the possibility of automating the approximation of IFPUG Function Points or COSMIC early in software projects.

Objective: Given a UML use-case diagram or a list of use-case names, approximate COSMIC and IFPUG FPA functional size in an automatic way.

Method: We propose a two-step process of approximating the functional size of applications based on use-case goals. In the first step, we process the names of use cases, expressed in a natural language and assign each of their goals into one of thirteen categories. In the second step, we employ information about categories of use-case goals and historical data to construct prediction models and use them to approximate the size in COSMIC and Function Points. We compare the accuracy of the proposed methods to the average use-case approximation (AUC), which is their most intuitive counterpart, and the automatic method proposed by Hussain, Kosseim and Ormandjieva (HKO).

Results: The prediction accuracy of the two proposed approximation methods was evaluated using a cross-validation procedure on a data set of 26 software development projects. For both methods, the prediction error was low compared to AUC and HKO.

Conclusion: Developers who document functional requirements in a form of use cases might use the proposed methods to obtain an early approximation of the application size as soon as use-case goals are identified. The proposed methods are automatic and can be considered as a replacement for AUC.

1. Introduction

One of the most important problems considered in Project-Portfolio Management (PPM) is the selection of projects to the project pipeline [1]. To make a good decision, one needs to know at least the business value and effort associated with each project. Obviously, in the context of PPM exact values are not known, and one has to rely on their indicators. One of such indicators, commonly accepted in software development projects, is the functional size of an application [2].

Several functional size measurement (FSM) methods have been proposed so far. The most recognized among them is IFPUG Function Point Analysis (FPA). The method was proposed by Albrecht in 1979 [3]. It has been widely accepted by industry, and several different variants have been proposed so far, e.g., NESMA FPA [4], FISMA FPA [5]. The IFPUG FPA method inspired other FSM methods such as Mark II FP [6] and COSMIC [7].

In the context of PPM, the decision maker is often presented with a business problem description and outline of the solution [8,9]. As regards software projects, one method of presenting a solution outline is a UML use-case diagram [10], augmented with some comments. To ensure the usefulness of such diagrams, the names of use cases must correctly reflect the user goals. Therefore, it would be beneficial to investigate if the names of use cases provide information that might be valuable in the context of size approximation[1] and effort estimation. Therefore, the following problem can be formulated:

Core problem: *Given a list of use-case names and historical data regarding functional size measurement, provide an approximation of the*

* Corresponding author.
E-mail address: mochodek@cs.put.poznan.pl, Miroslaw.Ochodek@cs.put.poznan.pl

[1] To avoid confusion between the terms: *effort estimation* and *size estimation*, it is commonly accepted to use the term *size approximation* when referring to the latter [11].

http://dx.doi.org/10.1016/j.infsof.2016.08.007
0950-5849/© 2016 Elsevier B.V. All rights reserved.

functional size of an application, expressed in COSMIC or IFPUG Function Points.

There are two types of methods solving the above-stated problem: (1) automatic and (2) requiring expert judgment. In this paper, we are interested in the former ones. Investigating this problem not only provides insight into how good computers can be at solving AI-like problems, but it can also be useful in a situation when there are many project proposals on the table, and one needs to evaluate them quickly (as it is in the case of PPM).

As follows from the overview of functional size approximation methods prepared by COSMIC Consortium [11] and from the literature review performed by the author, the methods applicable to Core problem are the average use-case approximation (AUC) [12,13] and the HKO method (Hussain–Kosseim–Ormandjieva) [14]. The former ignores the names of use cases—it takes into account only their number. It is interesting to see how taking into account use-case names can improve the accuracy of functional size approximation. The HKO method is based on frequency of linguistic features, and originally was proposed as a solution to a different problem. However, it seems inappropriate to arbitrary reject it from the set of possible approaches.

In this paper, we introduce a categorization scheme of use-case goals reflected in use-case names. It aims at capturing the relationship between the goals of use cases and the semantics of use-case scenarios. Similar approaches that rely on the categorization of use-case transactions and actions have been successfully applied to other problems, such as effort estimation based on use-case scenarios [15,16] and inferring about events in use cases [17]. The approach seems promising also in the context of the considered problem. Therefore, it seems beneficial to investigate the following research questions:

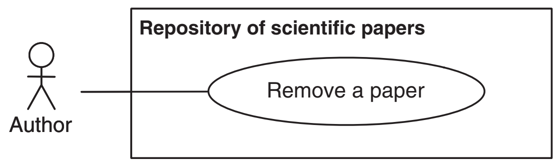
RQ1: Is it possible to propose a categorization scheme of use-case goals that would support functional size approximation based on use-case names?
RQ2: How to automatically categorize use-case names (expressed in a natural language) according to the proposed categories of use-case goals?
RQ3: How to automatically approximate COSMIC and IFPUG FPA functional size of an application based on use-case names labeled with the categories of goals and historical data concerning functional size measurement?

The paper is organized as follows. In Section 2, we present background information regarding use cases and the FSM methods considered in the study. We also discuss related studies concerning functional size approximation. Section 3 presents the research methodology, including the framework used to evaluate the accuracy of the approximation methods. In Section 4, we present the data set of twenty-six software development projects considered in the study. Section 5 introduces thirteen categories of use-case goals and discusses their usefulness in the context of size approximation (RQ1). Section 6 addresses question RQ2. We propose a method for automatic classification of use-case goals, and evaluate its prediction accuracy. Section 7 focusses on RQ3 and proposes two functional approximation methods. Evaluation of the prediction accuracy of these methods is presented in Section 8. Section 9 discusses the threats to validity of the study. Ths paper is concluded in Section 10.

2. Background and related work

2.1. Use cases

Use cases are a popular [18,19] scenario-based technique of describing the interaction between end-user (actors) and the system; which leads to obtaining an important *goal* from the user's perspective. These are called user-level use cases (there are also business-level use cases that describe the interaction between people and organizational units). The nature of user-level use cases is accurately captured by the OTOPOP rule (one time, one place, one person) [20]. The rule states that the goal of a user-level use case should be attainable by a single user during a single session with the system.



UC1: Remove a paper

Actors: Author

Main scenario:

1. Author requests to view his/her papers.
2. System presents the author's papers.
3. Author selects a paper to be removed from the repository.
4. Author asks the system to remove the selected paper.
5. System removes the paper from the repository.
6. System informs the author that the paper has been removed.

Alternative scenarios:

1.A. Author does not have any papers in the repository.
  1.A.1. System informs the author that he/she does not have any papers in the repository.
  1.A.2. Use case finishes.

Fig. 1. An example of a use case and use-case diagram.

State-of-the-art guidelines for writing use cases [21,22] advise us to document use cases with:

- A *name* that accurately expresses the goal of an actor. It should be formulated using a simple [verb + object] clause with an implied subject being the actor of the use case (e.g., "submit a paper");
- *Actors* participating in the use case (people, devices, or external systems);
- The *main scenario*, i.e. the most common sequence of steps allowing the actor to obtain the goal; according to Övergaard and Palmkvist [23], use-case scenarios may present different levels of details regarding the internal processing within the system, i.e., black-box (only the interaction between the actor and the system is described), gray-box (the most crucial system operations are presented that are important for the interaction with the actor), and white-box (detailed information about the internal processing is revealed).
- *Alternative scenarios* that are executed in response to the occurrence of some specified events (e.g., a user provided an invalid input).

An example of a use case specified according to the aforementioned guidelines is presented in Fig. 1.

Use cases are often visualized using UML use-case diagrams (UCD), which show actors and their goals. An example of a UCD is presented in Fig. 1. This diagram helps to get an overview of a use-case model without overwhelming the reader with the details of use-case scenarios.

There are three main types of relationships that can be defined between use cases: inclusion, extension, and generalization [22]. They are introduced in use-case models to reduce redundancies of scenarios between use cases. In the paper, we will reject included and specialized use cases from the analysis unless they represent complete, user-level use cases on their own.

2.2. Functional size measurement

In this section, we will briefly present the IFPUG FPA and COSMIC functional size measurement methods. We will also discuss the existing approaches to approximate functional size, with the focus on use cases.

2.2.1. Function Point Analysis

The Function Point Analysis (FPA) method was introduced by Alan Albrecht in 1979 [3]. In 1984, the International Function Point User Group (IFPUG) was founded, whose goal has been to maintain and popularize Albrecht's method. Since 1988, it has been publishing revised versions of the IFPUG FPA method on a regular basis. The IFPUG FPA method has been also approved as an international standard (ISO/IEC 20926:2009) [24]. Parallel to the development of IFPUG FPA, several different variants of the method were proposed, e.g., NESMA FPA [4], FISMA FPA [5].

IFPUG FPA includes two main Basic Functional Component (BFC) types, i.e., data functions (DF) and transactional functions (TF). Data functions are classified as internal logical files (ILF) or external interface files (EIF). There are also three types of transactional functions: external inputs (EI), external outputs (EO), and external inquiries (EQ). Each transactional function is a unique *elementary process* (EP). EP is defined as "the smallest unit of activity that is meaningful to the user, which must be self-contained and leaves the business of the application being counted in a consistent state" [24].

In order to calculate the functional size of an application, each transactional and data function is assigned into one of three complexity classes (low, average, high). Data functions are classified based on the number of data element types (DET) and record element types (RET). The complexity of transactional functions is assessed based on the number of DETs and logical file types referenced (FTR). Depending on the type and complexity, a function contributes a certain amount of Function Points (FP) to the functional size of the application, which is calculated as a sum of individual contributions of the functions included in the measurement scope.

Several authors investigated the possibility of mapping between use-case models and BFCs of the IFPUG FPA method [25–29]. In particular, they proposed rules and guidelines on how to measure the size of applications based on use-case models. Most of the authors concluded that use-case scenarios are too abstract to be used for identifying and measuring elementary processes. The same relates to data functions. Use cases might help to identify potential logical files, but they should not provide detailed information about the data model.

Several authors investigated inter-correlations between the components upon which the FP counting is based [30,31]. These studies suggest that it seems possible to simplify the method. One of the simplifications proposed by Lavazza et al. [31] was to count only the size of transactional functions (FP_{TF}). This variant seems well suited for use cases because it rules out the necessity of analyzing data models.

2.2.2. The COSMIC method

The COSMIC method is a popular, second generation FSM method. It has been developed by the Common Software Measurement International Consortium (COSMIC). The method has the status of an international standard (ISO/IEC 19761:2003) [7].

The BFCs considered in the COSMIC method are data movements (Entry, Exit, Read, or Write). The data movements are identified and counted within the scope of *functional processeses*. A functional process is defined as a set of data movements, representing an elementary part of the Functional User Requirements (FUR). Each functional process is unique within the FUR and can be defined independently of any other functional process. Every functional process has a triggering Entry and a set of data movements that is needed to meet its FUR for all the possible responses to its triggering Entry [32].

The unit of measure is called COSMIC Function Point (CFP). A single CFP corresponds to a movement of data attributes belonging to a single data group. Therefore, the functional size is calculated as the total number of data movements within all the functional processes within in the scope of measurement.

The mapping between COSMIC and use-case models has been also considered [33]. The authors seem unanimous in their opinions that availability of detailed use-case scenarios is necessary to apply the COSMIC method based on use cases [34–37]. It is required that use-case scenarios provide detailed information about all the data movements, including local reads and writes.

2.3. Early and rapid functional size approximation methods

Size approximation methods can be classified as *early* or *rapid* [38]. A method is considered early if it allows for approximating functional size before the FURs are specified at the level of granularity accepted by the FSM method. Rapid means that the method allows us to approximate the size more efficiently (processing time & cost) than in the case of measurement (usually compromising the accuracy of measurement).

A common approach to functional size approximation is to derive a scaling factor as an average size of requirements. In the context of this study, the most interesting example of this approach is the average use-case approximation method (AUC) [12] because it can be easily applied to approximate functional size based on use-case diagrams [13].

Fixed size classification and equal size bands are two examples of approximation by classification [11]. In these approaches, we create a certain number of complexity classes of functional components. Then we determine a scaling factor for each of these classes. In the equal size bands approach, the boundaries between classes (size bands) are determined in the calibration process so that the total size of all the functional components in each band is the same. Then the average size of functional components belonging to a band is used to approximate the size of the estimated functional component that is expected to belong to that band.

NESMA proposed two early and rapid methods applicable to Function Points: NESMA Indicative (FPA_i) and Estimated FPA [39]. The FPA_i method requires identifying potential logical files based on a data model. Then, the approximate size is obtained by multiplying the number of ILFs by 35 and the number of EIFs by 25. The Estimated NESMA method requires data and transactional functions to be identified and classified. All transactional functions (EI, EQ, EO) are considered to be of average complexity, and all data functions (ILF, EIF) are assumed to be of low complexity.

Early & Quick Function Points (E&Q FP) [40] combines different prediction approaches to allow approximation of size based on requirements expressed at a different level of granularity. Initially, the method was proposed to approximate IFPUG Function Points, but in 2004, Conte et al. [41] adapted that technique to the COSMIC method.

Another approximation method for IFPUG FPA and COSMIC is the EASY (Early & Speedy) Function Points method [42]. The method allows expressing uncertainty about the expected size of a function. The measurer provides the probability that the function will have a certain size. The probability level is further used to weight the size. The EASY approximation approach provides some of the commonly used probability distributions to ease the process of modeling distributions.

Table 1
Applicability of COSMIC and FPA functional size approximation methods to use cases.

Approx. method	Accepted input			Discussion
	Use-case goals only	UC goals & judgment	Use-case scenarios	
Average use-case (AUC) [12,13]	●	●	●	The approach can be employed to approximate functional size based on the number of use cases.
Hussain et al. (HKO) [14]	◐	◐	●	The approach could be potentially adapted to approximate functional size based on use-case scenarios. The method assumes that there is a relationship between the *frequency* of linguistic features in requirements and functional size. Therefore, the method might not be appropriate to approximate functional size based on use-case names because they are expressed using single, short sentences having a commonly-agreed structure.
Fixed size classification [11]	○	●	●	The approach can be applied to approximate functional size based on use cases. However, it would require judgment about the complexity of use-case scenarios.
Equal size bands [38]	○	●	●	Similarly to fixed size classification, this approach can be adapted to approximate functional size based on use cases. It would also require judgment about the complexity of use-case scenarios.
EPCU [43]	○	●	●	The method can be employed to approximate size based on use cases at early stages of software development. However, the measurer has to judge the complexity of use-case scenarios and the number of objects of interest. The method was calibrated using industrial data [38]; however, it might require local calibration.
E&Q Function Points [40,41]	○	●	●	The method could be adapted to approximate functional size based on use cases. However, there is no direct mapping between use cases and the levels of aggregation proposed in the method. The using of the method would require expert judgment about the level of aggregation of a given use case and its complexity. The method might also require some local calibration.
NESMA FPA$_i$ & Est. FPA [39]	○	◐	●	These methods require identification of candidates for either transactional or data functions. Therefore, they require knowledge about use-case scenarios.
EASY Function Points [42]	○	◐	●	The method might be used to approximate size based on use cases. However, the measurer would need to identify functional / elementary processes based on use-case scenarios and judge their complexity.
De Vito and Ferrucci [44]	○	○	●	The early & rapid method can be used to automatically approximate COSMIC size based on preconditions and use-case scenarios.
Bagriyanik and Karahoca [45]	○	○	◐	The results of early validation suggest that the method might be capable of automatically approximating functional size with nearly perfect accuracy [45]. However, the method requires a large variety of artifacts as an input, e.g., use cases, services, application interaction diagrams, and conceptual data models.

Valdés et al. [43] proposed the Estimation of Projects in a Context of Uncertainty model (EPCU). It is a fuzzy logic-based model that enables COSMIC size approximation. EPCU considers two input variables: the functional process size, and the number of objects of interest related to the functional process. Both variables are evaluated subjectively using a 0-to-5 unit scale. The input variables are then fuzzified into linguistic values. After defuzzification, the expected size of a functional process is in the 2 CFP to 16.4 CFP range.

Recently, De Vito and Ferrucci [44] proposed a quick and early method for approximating COSMIC functional size based on use-case models. They proposed to analyze the flow of data groups in use-case preconditions and scenarios.

Hussain et al. [14] proposed an approach to approximate COSMIC functional size from informally written textual requirements. The authors use natural language processing (NLP) tools to analyze the frequency of syntactic linguistic features in functional requirements (the number of words, frequency of nouns phrases, the number of keywords, etc.). They use this information to classify requirements into complexity classes established based on historical data.

Bagriyanik and Karahoca [45] proposed a rapid approximation method that is capable of automatically approximate functional size based on a domain-specific requirements ontology. The first studies showed that the method could be effectively used to approximate COSMIC size. However, it seems oriented towards later phases of software development and maintenance because it requires detailed information about requirements, services, and conceptual data models.

In Table 1, we summarize the applicability of the presented size approximation methods to approximate functional size based on use cases. It seems that only the AUC and HKO methods are capable of automatically approximating functional size based on use-case names. Other considered methods, such as fixed size classification, equal size bands, EPCU, and E&Q FP, would require additional human judgment about the complexity of use-case scenarios.

3. Research methodology

Because the considered problem of automatic approximation of functional size based on use-case names requires developing a set of methods (artifacts), we decided to base our research procedure on Design Science Research (DSR) [46,47].

3.1. Overview of the research procedure

We began the research process with a literature study to search for functional size approximation methods that might be considered as solutions to Core problem. Based on the search results summarized in Table 1, we concluded that the only methods that meet the constraints of the problem are AUC and HKO. These methods employ a quantitative approach to size approximation and do not incorporate any specific knowledge about the structure of use cases. Therefore, we decided to investigate the possibility of using the information carried by names of use cases to support functional size approximation (RQ1).

We collected a sample of use cases from 26 projects (described in Section 4) and measured the functional size of their applications using COSMIC and IFPUG FPA (more precisely FP_{TF})

We developed a categorization scheme of use-case goals based on outcomes of our previous studies [16,48] and the considered sample of projects (see Section 5). We discussed the usefulness of the proposed categories in the context of the following criteria:

- Completeness — the extent to which the scheme allows categorizing use cases written in accordance with the guidelines for writing use cases;
- Discriminating efficiency — the extent to which the categories discriminate use cases with respect to their functional size.

Having proposed the categorization scheme, we considered the problem of automatically categorizing use cases based on their names (RQ2). We developed and evaluated a prediction method that processes use-case names with the use of natural processing tools (NLP) and automatically predicts categories of use-case goals (see Section 6).

Finally, we investigated the problem of approximating functional size based on a set of use cases labeled with categories of goals and historical data concerning functional size measurement (RQ3). We propose two prediction methods (see Section 7). The first one is a direct derivative of AUC and is based on calculating the average size of use cases within the categories of goals. This method allows us to investigate if the accuracy of AUC could be improved by simply employing information about use-case goals (see Section 8). The second prediction method is based on Bayesian Networks (BNs). This time, we want to see how much the accuracy could be improved by employing a potentially more robust prediction model than a simple averaging.

We believe that the dissemination of results might precede further steps of DSR, namely the implementation and evaluation of the size approximation methods in software development organizations.

3.2. Size-approximation evaluation framework

The framework for evaluating the accuracy of functional size approximation that we propose is based on the guidelines by Shepperd and MacDonell [49].

We assumed that each of the considered approximation methods P_i could be used to predict the functional size of a system j ($\hat{y}_j$). Then, the prediction error or Absolute Residual (AR) could be calculated as the difference between the actual size y_j and the predicted size (see Eq. 1).

$$AR_j = y_j - \hat{y}_j \tag{1}$$

In order to assess the accuracy of P_i based on a set of n predictions, we calculated the mean and median Absolute Residual (MAR and MdAR).

After Shepperd and MacDonell, we standardized MAR relatively to the random guessing (P_0). The resulting standardized accuracy measure (SA) was calculated according to Eq. 2.

$$SA_{P_i} = \left(1 - \frac{MAR_{P_i}}{\overline{MAR}_{P_0}}\right) \times 100 \tag{2}$$

$\overline{MAR}_{P_0}$ is the mean Absolute Residual of random guessing. Its value can be determined using the exact algorithm proposed by Langdon et al. [50] or based on the Monte Carlo simulation—as originally proposed by Shepperd and MacDonell. We used both of these approaches. The exact algorithm was used to calculate $\overline{MAR}_{P_0}$, while simulation was performed to determine the 5% quantile of random guessing, which is the 5% quantile of the MARs distribution obtained for random guessing in all rounds of the Monte Carlo simulation. In both cases, we allowed random guesses to alight on the correct answer.

The interpretation of SA_{P_i} is that the ratio represents how much more (or less) accurate the model P_i is compared to random guessing.

Despite its well-justified criticism, we decided also to consider the Mean Magnitude of Relative Error, which is given by Eq. 3.

$$MMRE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|/y_i}{n} \tag{3}$$

It has been shown that MMRE is an asymmetric measure and could lead to a flawed interpretation of the prediction results [51]. However, since it gives an indication of prediction accuracy relative to the actual size, it might be easier to interpret by practitioners than MAR or MdAR. Therefore, we decided to include it into the analysis as a secondary, supportive criterion.

We used the above criteria and the "leave-one-out" (LOOCV) cross-validation procedure to evaluate each of the proposed size-approximation methods P_i in the context of functional size measure m ($P_{i,\,m}$). Cross-validation is a statistical method for validating a predictive model. The set of observations is divided into two subsets. One of them, called a training set, is used to build a prediction model. The remaining one is used to validate the model. LOOCV is a form of cross-validation that leaves out a single observation for the validation purpose at a time.

As suggested by Shepperd and MacDonell, we began the validation of the method $P_{i,\,m}$ by verifying if it provides predictions with a higher accuracy than random guessing ($P_{i,\,m} \succ P_{0,\,m}$). We assumed that the approximation method outperforms random guessing if the SA calculated for $P_{i,\,m}$ is greater than the SA for the 5% quantile of random guessing (we denote the difference between both as $\Delta SA_{5\%}$). The role of the 5% quantile of random guessing in the evaluation framework is similar to the role of significance level (α) in statistical hypothesis testing. By comparing SA for the method with the SA for the 5% quantile of random guessing, we reduce the probability that the superiority of the method is observed by chance.

In the following step, we compared the accuracy of each proposed approximation method P_i with the accuracy of the average use-case approximation (AUC_m) and the HKO method (HKO_m). For each measure $m \in \{CFP, FP_{TF}\}$, we investigated hypotheses that $P_{i,\,m} \succ AUC_m$ and $P_{i,\,m} \succ HKO_m$ based on:

- The comparison of SA calculated for the methods;
- The results of one-tailed Wilcoxon signed-rank test for Absolute Residuals (AR);
- Cliff's δ [52] — A non-parametric effect size measure calculated for Absolute Residuals (AR) according to Eq. 4 (x_1 and x_2 are scores within groups 1 and 2; n_1 and n_2 are the sizes of the sample groups; # is the cardinality symbol). It estimates the probability that a value selected from one of the groups is greater than a value selected from the other group, minus the reverse probability. The measure ranges between +1 and −1. The extreme values indicate the absence of overlap between the two samples whereas zero suggests equivalence of samples distributions [53].

$$\delta = \frac{\#(x_1 > x_2) - \#(x_1 < x_2)}{n_1 n_2} \tag{4}$$

Cliff's δ can also be converted to the Number Needed to Treat (NNT) effect size (NNT=δ^{-1}) [54]. In the context of this study, NNT can be interpreted as the number of project proposals that would have to be in a project portfolio to expect that the size of one more of the proposals was approximated with a higher (or lower) accuracy than if the same proposals had been estimated with the use of a baseline approximation method.

The magnitude of the Cliff's δ effect size can also be assessed with the use of thresholds proposed by Romano et al. [55]: $|\delta| < 0.147$ (NNT > 6.8) "negligible", $|\delta| < 0.33$ (NNT > 3.0) "small", $|\delta| < 0.474$ (NNT > 2.1) "medium", otherwise "large". The thresholds were derived from the thresholds for the Cohen's d effect size [56] under the assumption of normality of samples

Table 2
Application domain and basic description of the projects under study. *Type: N—new development; C—customization of an existing product; E—enhancement project; Origin: I—project developed by a software development company; U—project developed at the university by staff or students for internal use or external customer.*

ID	Developer	Type	Origin	Product description
P01	D1.1	N	U	Java, Oracle DBMS, Hibernate, GWT-based custom framework, JSON, OSGi, Backend application for the university admission system (Rich Internet Application).
P02	D1.1	E	U	Java, Oracle DBMS, Apache Struts 1.2, Java Swing, Web-based frontend for the university admission system. Some parts were re-used from the previous prototype version.
P03	D1.2	N	U	PHP, PHPLiteMVC, PostgreSQL DBMS, C#, Web-application and daemon for managing life-cycle of smart cards with an exemplary client application.
P04	D1.2	E	U	PHP, PHPLiteMVC, PostgreSQL DBMS, C#, Two client applications for the system P16 (web-based application for managing life-cycle of smart cards).
P05	D1.3	E	U	Java, GWT, PostgreSQL DBMS, Hibernate, Web-application for managing e-protocols for students grades
P06	D1.3	N	U	C#, ASP.Net, MS SQL DBMS, Web-based Customer Relationship Management (CRM) system.
P07	D1.3	N	U	Java, SmartGWT, PostgreSQL DBMS, Hibernate, Web-application for monitoring assignments of organizational duties.
P08	D1.3	E	U	PHP, Moodle, PostgreSQL DBMS, Web-application, a module to Moodle LMS enabling surveying students and alumni, which integrates with the University e-services.
P09	D1.3	E	U	Java, Oracle DBMS, Hibernate, GWT-based custom framework, JSON, OSGi, University admission system for foreign students.
P10	D1.3	N	U	Java, Ninja Framework, PostgreSQL DBMS, Hibernate, Web-application for collecting and raporting bibliometric information.
P11	D1.3	N	U	Java, PostgreSQL DBMS, Hibernate, Web-application for planning educational duties assignments.
P12	D1.3	N	U	Java, Java Servlets, Spring, Android, SQLite, MySQL, Mobile application for federation of libraries.
P13	D1.3	N	U	Java, Spring, PostgreSQL DBMS, Hibernate, Web-application for collecting information about a faculty.
P14	D1.3	N	U	Java, Spring, PostgreSQL DBMS, Hibernate, Web-application for storing and evaluating studies programmes.
P15	D1.3	N	U	Java, Java Servlets, JSP, Java Swing, SOAP, MySQL DBMS, Web and standalone application for managing members of the organization.
P16	D1.3	N	U	Ruby, PostgreSQL DBMS, Web-application that collects information about alumni from web pages.
P17	D1.3	N	U	Java, Hibernate, Axis, PHP, C++, Bibliometric Information System. A system for collecting information regarding publications and citations. (Limited GUI)
P18	D1.3	N	U	Java, Apache Struts 2, Hibernate, Web-based system supporting the assignment of B.Sc and M.Sc. theses projects.
P19	D2	N	I	C#, ASP.Net, MS SQL DBMS, custom web-framework and Web-based e-commerce solution.
P20	D3	N	I	Python, Django, PostgreSQL, Web application for collecting and tracking projects metrics.
P21	D3	N	I	Python, Plone, Zope, Web application developed based on existing CMS solution.
P22	D4	N	I	PHP, PostreSQL DBMS, Yii web-framework, ExGWT (administration panel) , An e-learning web platform.
P23	D4	N	I	PHP, PostreSQL DBMS, Yii web-framework, Web application for handling customers' orders.
P24	D5	N	I	Java, PHP, MySQL DBMS, Eclipse Rich Client Platform (RCP), Web-based repository of invoices with additional standalone client application.
P25	D6	C	I	Python, Plone, Zope, Content Management System (CMS).
P26	D7	N	I	Delphi, Firebird DBMS, Bank system integrating payments into virtual accounts in one real account.

distributions (e.g., $d = 0.2$ corresponds to $\delta = 0.147$). Although neither Cohen nor Romano et al. proposed their thresholds in the context of Software Engineering research, the thresholds seem acceptable from the perspective of this study. For instance, the borderline between the "small" and "negligible" effect size is at NNT = 6.8, which means that there should be at least seven proposals in a project portfolio to expect improvement in the approximation accuracy for one more project comparing to a baseline method. Conversely, the "large" effect size would require only 2 to 3 project proposals to achieve the same improvement. Another argument for using the thresholds proposed by Romano et al. is that they were also employed in other research in the area of functional size measurement [57].

4. Characteristics of projects

The study is based on the analysis of data from 26 projects. The data set consists of 427 use cases. The set is an extension of the Ochodek's data set [16]. A brief description of the projects and their expected outcomes is presented in Table 2.

The projects were developed by six software development companies (D2-D7) and Poznan University of Technology—PUT (D1). In the case of PUT, the projects were developed by a team established to develop a system for handling student admission process at the University (D1.1); Department of Software Development—a unit at PUT, hiring professional software developers to deliver software for internal purposes of the University (D.1.2); Software Development Studio (SDS) [58] (D1.3); or as a B.Sc. project for the internal use at the University (D1.4).

All of the products delivered by the projects could be classified as "business application software" [59]. However, their business domains differ. The use cases seem similar when it comes to use-case writing style. Most of the user-level use cases are defined at a similar level of granularity, convergent with the OTOPOP rule. Similarly, nearly all of the use cases are somehow inspired by the flow of user interface (UI). However, only in a few of them we found explicit references to UI elements (e.g., user chooses the title from the combo box).

Although most of the use cases were following the already mentioned guidelines for writing use cases, for some of them we found anomalies related to their proper naming. We observed that 4% of use cases had misleading names that did not correspond to the semantics of their scenarios. Another 2% were so-called CRUD (Create, Retrieve, Update, Delete) or partial-CRUD use cases [23] whose names suggested only one of the CRUD operations.

The measurement was performed based on use-case models and supplementary material available for the projects, e.g., data models, user interface (UI) designs, application screens, and working application. The results of the measurement are presented in Table 3.

5. Categories of use-case goals

Use-case scenarios describe how the goal expressed in the name of a use case might be achieved by an actor. Therefore, we could expect that based on the goal of a use case we should be able to infer about the complexity of its scenarios.

The proposed categorization scheme for use-case goals aims at capturing the relationship between the goals of use cases and

Table 3
Projects characteristics (*the total size of an application; mean use-case size and standard deviation*).

ID	User-level use cases	COSMIC total	COSMIC mean	COSMIC SD	FP_{TF} total	FP_{TF} mean	FP_{TF} SD
P01	42	693	16.5	9.0	389	9.3	4.9
P02	17	211	12.4	5.7	108	6.4	2.8
P03	36	342	10.7	7.7	220	7.2	3.0
P04	23	159	7.4	4.9	115	5.5	3.6
P05	19	151	7.9	4.3	108	5.7	3.4
P06	26	161	6.2	1.4	109	4.2	2.1
P07	13	113	8.7	5.3	83	6.4	4.9
P08	8	78	9.8	3.5	47	5.9	2.4
P09	13	138	10.6	6.0	112	8.6	4.6
P10	9	70	7.8	4.2	47	5.2	2.5
P11	10	81	8.1	4.2	52	5.2	2.8
P12	11	45	4.1	1.4	35	3.2	0.4
P13	12	91	7.6	3.2	57	4.8	2.4
P14	13	85	6.5	4.1	58	4.5	2.1
P15	14	86	6.1	4.5	59	4.2	2.9
P16	7	49	7.0	3.2	37	5.3	2.4
P17	6	54	9.0	4.2	42	7.0	3.9
P18	18	116	6.4	4.3	90	5.0	3.8
P19	31	399	12.9	6.0	269	8.7	4.6
P20	11	125	11.4	8.5	84	7.6	4.7
P21	19	217	12.3	17.0	111	6.6	3.7
P22	25	142	5.7	2.0	108	4.3	1.5
P23	9	59	6.6	2.1	46	5.1	2.0
P24	10	45	4.5	1.2	32	3.2	0.4
P25	10	85	8.8	3.7	57	6.0	3.2
P26	15	136	9.1	6.3	88	5.9	4.7

semantics of activities in their scenarios. Use-case scenarios are composed of actions that form larger structures called use-case transactions. A use-case transaction is defined similarly to the transactional function in IFPUG FPA as “the smallest unit of activity that is meaningful from the actor’s point of view that is self-contained and leaves the business of the application being sized in a consistent state” [60].

In the previous studies [16,48], we investigated the possibility of categorizing use-case transactions based on the semantics of actions they contain. As a result, we proposed twelve semantic types of transactions: Create (C), Retrieve (R), Update (U), Delete (D), Link (L), Delete Link (DL), Asynchronous Retrieve (AR), Dynamic Retrieve (DR), Transfer (T), Check Object (CO), Complex Internal Activity (CIA), and Change State (CS). These types can be used to describe the structures of use-case scenarios. For instance, the scenarios of the use case presented in Fig. 1 are composed of two transactions: Retrieve (R)—the listing of the author’s papers, and Delete (D)—removing a paper from the repository. To shorten the description of the structure, we use an alphabetically-ordered list of abbreviated names of transaction types (e.g., D|R).

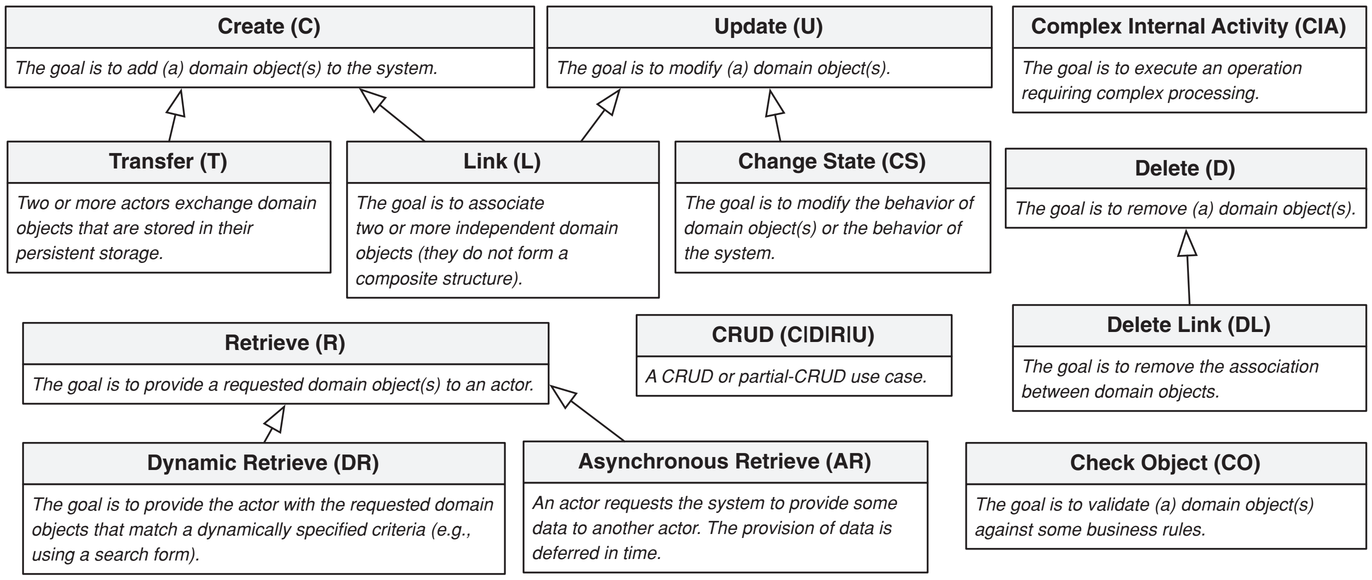
In our sample of projects, we observed that ~60% of user-level use-cases have only one transaction in their scenarios. In these cases, the type of transaction clearly corresponds to the goal of the use case. For use cases that include two or more transactions, there is usually a single, dominant transaction, strongly related to the goal of the use case, and a set of supporting transactions. For instance, in the previously discussed example of use case presented in Fig. 1, the Delete (D) transaction is the dominating one, while Retrieve (R) plays an auxiliary role. Finally, we observed that around 11% of use cases have a number of equally important transactions. These were CRUD or partial-CRUD use cases.

Based on theses observations, we proposed thirteen categories of use-case goals that correspond to the types of dominant transactions in use-case scenarios. The categorization scheme is presented in Fig. 2.

We also considered categorizing use-case goals using the types of transactional functions in the IFPUG FPA method — EI, EO, and EQ (see Section 2.2.1). Unfortunately, we observed that it was impossible to distinguish between EO and EQ types of use cases only based on their names. For instance, a dominating transaction in a use case entitled “View a paper” could be either EQ (e.g., the paper is fetched and presented as it is) or EO (some calculations are performed before the paper is presented to the actor, e.g., the keywords are determined based on the analysis of the frequency of words in the text).

5.1. Completeness of categories

Several authors tried to categorize different elements of use cases, e.g., actions [17,61], events [62], transactions [16,48], or scenarios [63]. There have also been studies aiming at developing catalogs of use-case patterns [23,64]. Some of these patterns might be used to categorize use-case goals. However, by definition, they focus only on describing solutions to the most commonly observed problems (often domain-specific). Thus, a categorization scheme based on such patterns seem incomplete.



Create (C)
The goal is to add (a) domain object(s) to the system.
Update (U)
The goal is to modify (a) domain object(s).
Complex Internal Activity (CIA)
The goal is to execute an operation requiring complex processing.
Transfer (T)
Two or more actors exchange domain objects that are stored in their persistent storage.
Link (L)
The goal is to associate two or more independent domain objects (they do not form a composite structure).
Change State (CS)
The goal is to modify the behavior of domain object(s) or the behavior of the system.
Delete (D)
The goal is to remove (a) domain object(s).
Retrieve (R)
The goal is to provide a requested domain object(s) to an actor.
CRUD (C|D|R|U)
A CRUD or partial-CRUD use case.
Delete Link (DL)
The goal is to remove the association between domain objects.
Dynamic Retrieve (DR)
The goal is to provide the actor with the requested domain objects that match a dynamically specified criteria (e.g., using a search form).
Asynchronous Retrieve (AR)
An actor requests the system to provide some data to another actor. The provision of data is deferred in time.
Check Object (CO)
The goal is to validate (a) domain object(s) against some business rules.

Fig. 2. Categorization scheme of use-case goals (arrows show the specialization/generalization relationships between the categories).

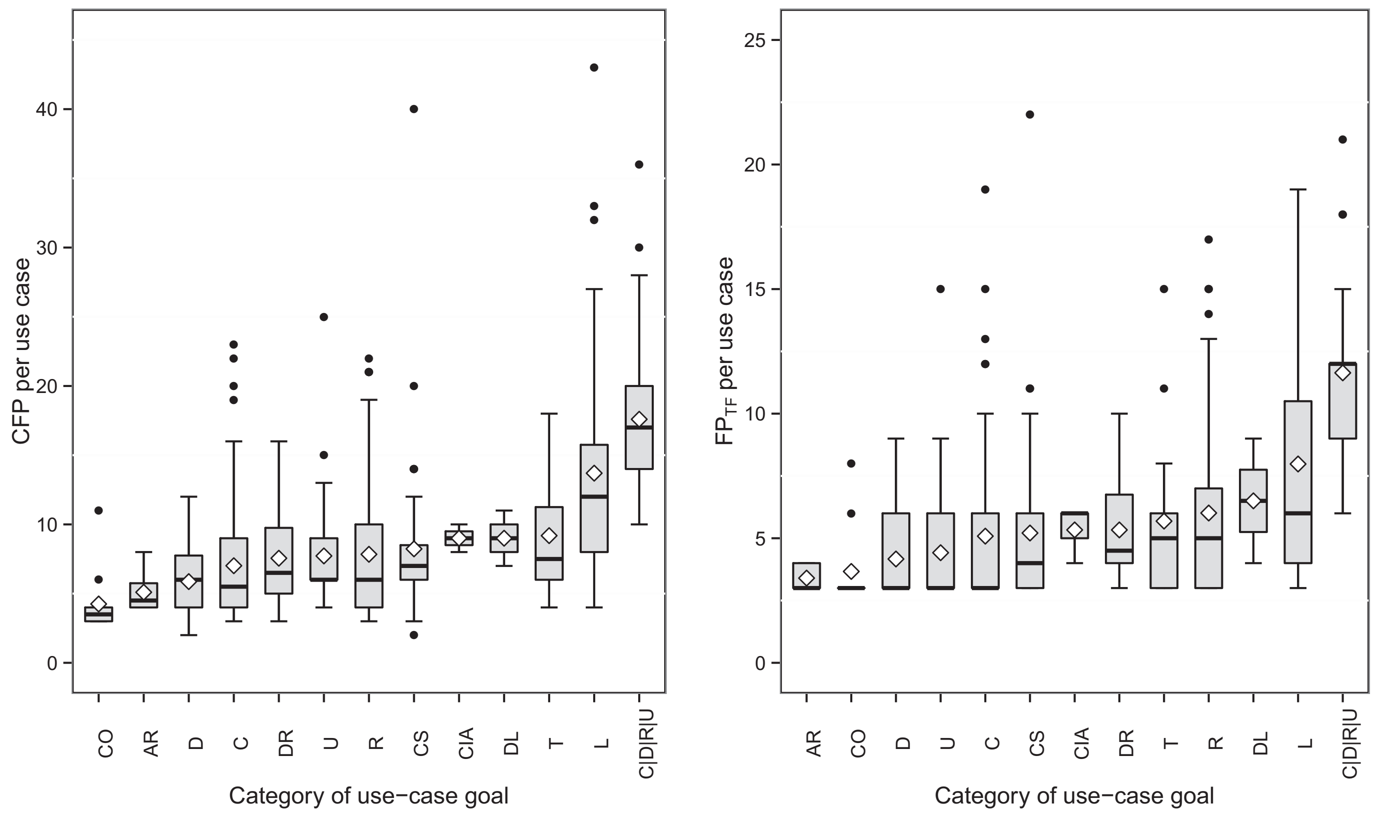


Fig. 3. Distributions of functional size of use cases depending on the category of use-case goal (ordered asceding by mean size of use case ◇).

One way of ensuring the completeness of a categorization scheme is to propose categories that are abstract and independent of business domain, such as the ones included in the proposed categorization scheme. Using the proposed categorization scheme, we were able to categorize all of the 427 use cases coming from few different organizations and describing systems operating in different domains. Therefore, it seems that the proposed categories of use-case goals are sufficiently generic to be used in different contexts. Nevertheless, we still consider the proposed categorization scheme open for extensions.

5.2. Discriminating efficiency of the categories

A categorization scheme should consist of categories that are meaningful and allow categorizing use cases. However, to be useful in the context of size approximation, it also has to efficiently discriminate use cases with respect to their functional size (the more the average functional size of use cases differ between the categories, the better approximation accuracy can be expected). Otherwise, a method using the categorization scheme would become equivalent to the AUC method.

Although the guiding criterion for the categorization scheme was not the efficiency in discriminating use cases with respect to their functional size but the nature of use-case goals, the box plots presented in Fig. 3 show that the mean functional size of use cases differed for all but two pairs of the proposed categories. The series of Kruskal-Wallis tests ($\alpha = 0.05$) with Holm–Bonferroni correction [65] and Conover's post-hoc analysis [66] indicated that the differences could be perceived as statistically significant for 59% and 54% pairs of the categories of use-case goals (COSMIC and FP_{TF}, respectively). Unfortunately, some of the categories included only a small number of use cases, negatively affecting statistical power of the tests, i.e., decreasing the chance of detecting a difference if it truly exists in the population. Finally, the observed Cliff's δ effect size was on average equal to 0.49 for COSMIC and 0.42 for FP_{TF} (SD = 0.28 for both measures). It means that on average one needs to pick randomly only three pairs of use cases from two different categories to expect that the set of use cases for one of them will contain one more use case of greater (or smaller) size than the set for the other category.

We believe that the observed differences could be explained by the fact that the proposed categories discriminate efficiently between different structures of use-case scenarios. Fig. 4 presents a contingency table which shows frequency distributions of the structures of use-case scenarios (columns) in use cases having different categories of goals (rows). By investigating the columns, we observe that nearly all of the scenario structures (~90%) appear only in a single category of use cases.

6. Automatic classification of use-case goals

To consider the question RQ2, we investigated the possibility of developing an automatic method that would be capable of categorizing use cases based on their names. The overview of the proposed solution is presented in Fig. 5.

In the first step of the proposed method, we analyze use-case names in a processing pipeline composed of natural processing tools (NLP) provided by the Stanford Parser kit [67]. The pipeline includes sentence segmentation, word tagging, part-of-speech tagging, lemmatization, and dependency parsing (i.e., finding the subjects, predicates, and objects of a sentence).

Based on the results of natural language processing, we extract linguistic features that are further used by a prediction algorithm:

- The number of predicates (numerical);
- The presence of prepositions *to* or *from* (boolean);
- The presence of predicates having antonym forms (boolean);
- The object of a sentence is plural (boolean);

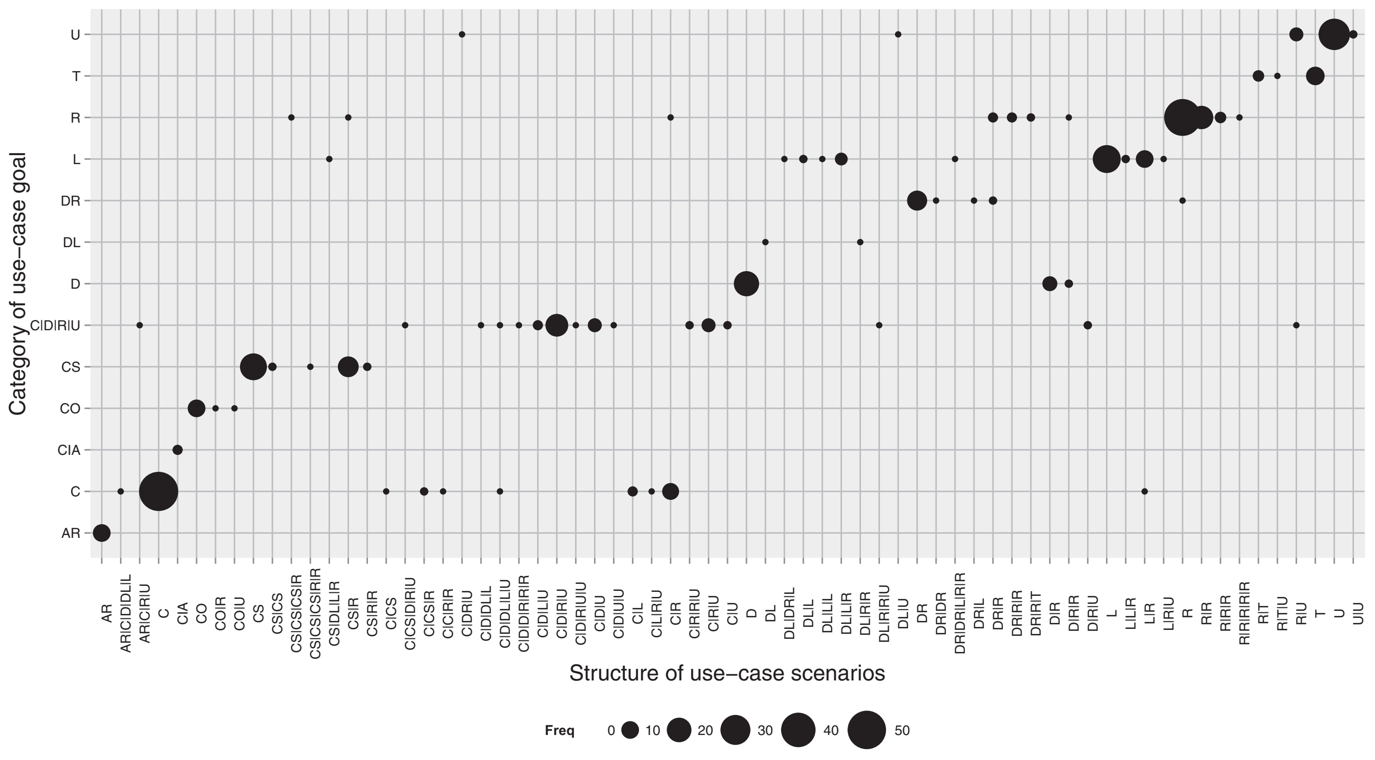


Fig. 4. A contingency table showing multivariate frequency distribution of use-case structures and categories of use-case goal.

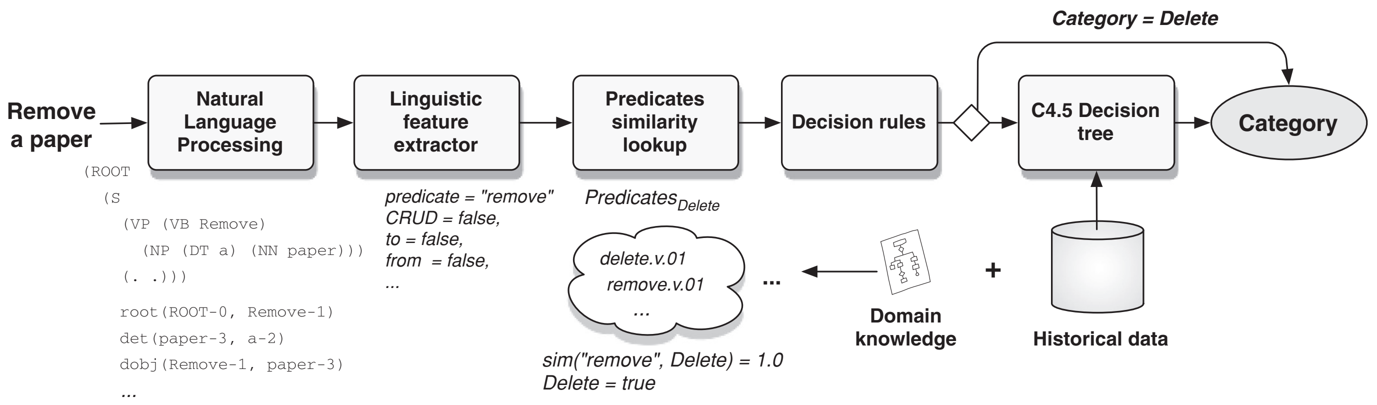


Fig. 5. The process of automatic categorization of use cases based on their names presented on the example of the use case from Fig. 1.

- The object of a sentence representing association between domain objects, e.g., reference, association, binding, connection (boolean);
- The presence of CRUD or partial-CRUD abbreviations (boolean);
- The presence of a noun representing a container object, e.g., a list, set, group (boolean);
- An object of a preposition is the name of the “system” actor (boolean).

Except for the number of predicates, all of the above-presented features have a qualitative nature.

In the next step, we analyze if predicates describe activities related to specific categories of use-case goals. We create a set of predicates for each category *C* that describe activities related to the goal the category represents ($Predicates_C$). Because a single word can have many meanings, we do not store a predicate itself but a reference to its set of synonyms (synset) available in the WordNet lexical database [68]. For instance, the set for the category Transfer (T) might consist of the synsets[2]: import.v.02, export.v.02, move.v.02, transfer.v.02, synchronize.v.06. Because WordNet allows us to establish similarity of meanings, storing synsets instead of words helps to tackle with new verbs that were not present in the training set.

The sets of predicates created for the purpose of this study consisted of common verbs that seem related to the proposed categories of goals. However, one can extend the sets to include terminology related to a specific business domain.

The predicate sets are used to derive additional thirteen boolean features. A positive value of a given feature *C* (Create, Retrieve, etc.) indicates that the predicates in a use-case name refer to activities related to the goal of the category *C*. This feature is determined based on measuring the similarity between predicates

[2] WordNet identifies a set of synonyms (a synset) by a triple: [word].[part-of-speech].[meaning] (e.g., import.v.02).

in the name of the use case and the synsets in the set of predicates for the category C—denoted as *sim(predicates, C)*.

The process of calculating *sim(predicates, C)* is two-fold (see Eq. 5). First, we use Lin's similarity measure[3] (sim_{Lin}) [70] to assess similarity between each pair of predicates and synsets in $Predicates_C$. Afterwards, we calculate *sim(predicates, C)* as a maximum sim_{Lin}. Finally, if *sim(predicate, C)* is greater than or equal to 0.95, the value of the feature *C* is set to "true"; otherwise, it is set to "false" (see Eq. 6).

$$sim(predicates, C) = max(sim_{Lin}(p_i, s_j)), \quad p_i \in predicates, s_j \in Predicates_C \tag{5}$$

$$C = \begin{cases} true & : sim(predicates, C) \geq 0.95 \\ false & : sim(predicates, C) < 0.95 \end{cases} \tag{6}$$

The last considered feature is the number of CRUD elementary operations (numeric). It is determined using the above-described features, by counting the number of predicates that represent Create, Retrieve, Update, or Delete activities.

The final prediction algorithm consists of two main steps. First, a vector of features describing a use-case name is processed by a chain of decision rules presented in Listing 1. If none of the rules apply, the vector is passed to a prediction model built based on C4.5 decision trees [71]. (We used the implementation of the C4.5 (revision 8) available in the WEKA package [72].)

```
Data: name — a vector of features describing the name of a use
      case;
Result: the category of use-case goal.
if name.no_of_predicates > 1 and
  name.no_of_crud_operations > 1 then
    return "C|D|R|U";
if name.Link or
  (name.CRUD and name.association_object) or
  (name.Create and name.association_object) then
    return "Link";
if name.Transfer then
    return "Transfer";
if name.CRUD or name.crud_abbrv then
    return "C|D|R|U";
if name.Dynamic_Retrieve then
    return "Dynamic Retrieve";
if name.Asynchronous_Retrieve then
    return "Asynchronous Retrieve";
if name.Change_State then
    return "Change State";
if name.Create then
    return "Create";
if name.Check_Object then
    return "Check Object";
if name.Update then
    return "Update";
if name.Delete_Link and
  (name.association_object or name.from) then
    return "Delete Link";
if name.Delete then
    return "Delete";
if name.Complex_Internal_Activity then
    return "Complex Internal Activity";
if name.antonyms then
    return "Change State";
return C4.5(name)
```

Listing 1. A chain of decision rules for automatic categorization of use-case goals.

6.1. Evaluation of the automatic categorization

We employed the LOOCV cross-validation method to evaluate the prediction performance of the proposed classification method. In each run, the method was trained on use cases from n-1 projects. It was then used to predict the categories of goals of the use cases belonging to the remaining project. We evaluated the prediction performance based on the analysis of the confusion matrix presented in Fig. 6 and popular prediction performance measures that can be derived from confusion matrices, i.e., accuracy, recall (sensitivity), precision, specificity, and F-score (see Appendix A).

Because we considered a multiclass prediction problem, we had to evaluate the prediction performance at two levels: macro (overall prediction performance) and micro (prediction performance of categorizing use cases to certain categories).

The accuracy was equal to 0.78 (95% confidence interval (CI) between 0.73 and 0.81). The macro recall and precision were 0.73 and 0.77, respectively. The macro F-score was equal to 0.74. These results seem promising, especially when taking into account that the method was evaluated based on use cases coming from real software projects that have some defects and anomalies (discussed in Section 4).

At the micro level, the most difficult task was to discriminate between categories and their sub-categories (e.g., discriminate between Retrieve and Dynamic Retrieve or Asynchronous Retrieve). The observed recall for main categories was on average equal to 0.85, comparing to 0.60 for their sub-categories. The specificity was, however, similar for both categories and sub-categories and equal to 0.98 and 0.99, respectively. These results do not seem surprising because the slight differences in the meaning of these categories might not be reflected in the names of use cases.

Taking into account the problems related to distinguishing between main and sub-categories, we performed a similar analysis for a simplified categorization scheme including only the main categories of goals (C, R, U, D, CIA, CO, C|D|R|U). We observed that the

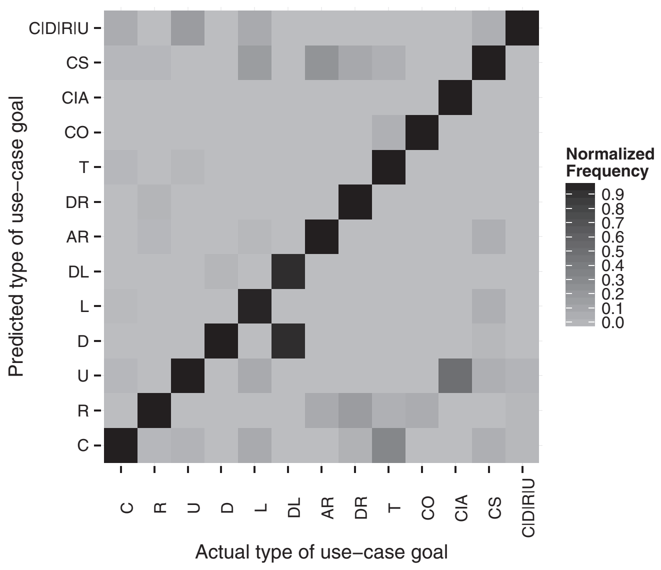


Fig. 6. Confusion matrix summarizing the quality of predicting categories of use-case goals based on use-case names.

[3] We use the implementation of Lin's similarity measure available in the NLTK library [69].

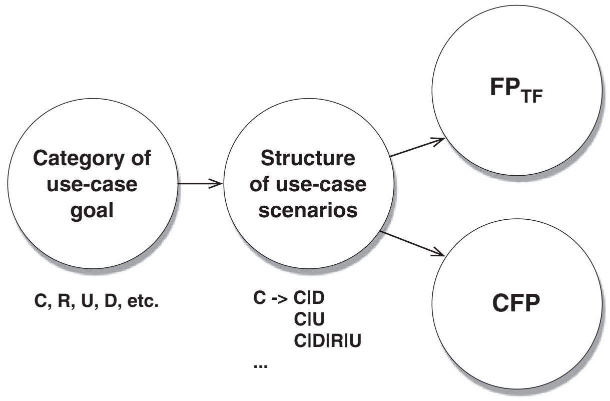


Fig. 7. A graphical representation of Bayesian Network Use-Case Goal AproxImatioN (BN-UCGAIN).

accuracy of the automatic classification increased from 0.78 to 0.83 (95% CI between 0.79 and 0.87). However, it turned out later that reducing the number of categories had a negative impact on the accuracy of the functional size approximation. Therefore, we decided to use the full categorization scheme.

7. Functional-size approximation models

To investigate if the proposed categorization scheme might support approximation of functional size, we proposed and evaluated two approximation methods that incorporate information about use-case goals.

The first method is called Average Use-Case Goal-aware Approximation (AUCG). It mimics the concept of the average use-case approximation (AUC). The main difference is that it approximates functional size based on the average size of use cases belonging to a given category. If historical data for the category are not available, it tries to calculate the average size of use cases belonging to its parent categories. If the category does not have parent categories or there is no historical data for its parents, we fall back to the average use-case size approximation.

The second model—Bayesian Network–based Use-Case-Goal-aware ApproxImatioN (BN-UCGAIN)— is based on Bayesian Networks (BNs) [73]. A BN is a probabilistic graphical model that support reasoning under uncertainty. Such a network has the form of directed acyclic graph (DAG), with nodes representing random variables and edges representing conditional dependencies between the variables. Each node is associated with a node-probability table (NPT). The table binds a set of node's parent states with the probability of the states of the node.

The proposed BN-UCGAIN model, presented in Fig. 7, includes four random variables:

- Category of use-case goal—it is a categorical variable concerning thirteen use-case goals defined in the categorization scheme.
- Structure of use-case scenarios—it is a categorical variable representing types of semantic transactions in use-case scenarios. For instance, C|U would represent a use case that has Create (C) and Update (U) transactions in its scenarios.
- CFP and FP$_{TF}$ variables represent COSMIC and FPA functional size of use cases. Because BNs have difficulties in modelling continuous probability distributions (unless the distribution is Gaussian) we decided to discretize these variables using the equal size bands algorithm proposed by Vogelezang and Prins [38].

Once the conditional probabilities are computed based on historical data, the Bayesian Network can be used to infer about states of nodes based on provided evidence. For instance, we may provide a category of use-case goal as evidence and query for the probabilities of other nodes. It allows approximating the size of a use case according to Eq. 7.

$$Size(uc) = \sum_{i=1}^{n} \widehat{P_{B_i}}(uc) \times \mu_{Size}(B_i) \qquad (7)$$

where

- *Size(uc)* is the functional size of the use case *uc* (COSMIC or IFPUG FP$_{TF}$);
- n is the number of equal size bands ($n = 4$);
- B_i is the ith size band;
- $\widehat{P_{B_i}}(uc)$ is the inferred probability that the actual size of *uc* falls within the boundaries of the size band B_i;
- $\mu_{Size}(B_i)$ is the average size of use cases in the size band B_i.

We also took into account the concept of *uniqueness*. It is defined differently by each of the considered measurement methods, but its general idea is that each elementary or functional process shall be measured only once. Unfortunately, we observed that the descriptions of the same process might be repeated in many use cases to help the readers understand the context. To mitigate this problem, we introduced a *uniqueness coefficient*. It is calculated as a quotient between the size of an application taking and not taking into account the uniqueness rule. We used the coefficient to scale the final approximation of size. However, for the projects in the considered data set, the uniqueness coefficient was nearly always very close to one. Thus, its real impact on the approximation accuracy was negligible.

8. Evaluation of the size approximation accuracy

We used the evaluation framework described in Section 3.2 to find out if the methods incorporating information about use-case goals (AUCG and BN-UCGAIN) are more accurate in approximating the functional size than the AUC and HKO methods.

In the study, we considered two variants of each proposed method. The variants denoted as AUCG and BN-UCGAIN employed the automatic categorization method presented in Section 6. A new instance of the category prediction method was created in each run of cross-validation based on the training set. To provide additional baselines for comparison, we also considered two additional variants of the proposed methods denoted as AUCG* and BN-UCGAIN*. These variants did not use the automatic categorization method. Instead, they assumed that the categories of use cases were known a priori. Thus, they represent an ideal case when all categories of use-case goals are already identified.

We also re-implemented the NLP processing chain of the HKO method based on the papers by the authors of the method [14,74].

Table 4
Comparing AUC, AUCG, and BN-UCGAIN accuracy results ($\Delta SA_{5\%}$ is a difference between SA and SA$_{5\%}$ calculated for the 5% quantile of random guessing).

Approx. method	Measure	MdAR	MAR	MMRE (%)	SA (%)	$\Delta SA_{5\%}$ (%)
HKO	CFP	35.00	58.73	43.27	49.80	19.95
AUC	CFP	24.33	46.33	33.12	60.40	30.55
AUCG*	CFP	21.80	31.39	21.69	73.17	43.32
AUCG	CFP	18.55	29.80	21.15	74.53	44.68
BN-UCGAIN*	CFP	21.38	28.62	18.95	75.53	45.68
BN-UCGAIN	CFP	16.91	26.64	19.18	77.23	47.38
HKO	FP$_{TF}$	22.50	32.96	32.23	53.78	23.78
AUC	FP$_{TF}$	13.24	24.55	26.90	65.58	35.57
AUCG*	FP$_{TF}$	8.74	16.23	16.82	77.24	47.23
AUCG	FP$_{TF}$	11.55	16.23	18.51	77.24	47.24
BN-UCGAIN*	FP$_{TF}$	8.39	15.65	16.23	78.06	48.05
BN-UCGAIN	FP$_{TF}$	12.79	15.18	17.18	78.71	48.71

Table 5
Significance testing results (p-values) and effect sizes (Cliff's δ) for comparison between AUC, AUCG, BN-UCGAIN (* denotes variants with a priori known categories of use-case goals).

Approx.		p-values, Wilcoxon signed-rank test (AR)						Cliff's δ effect size (AR)					
method	Measure	HKO	AUC	AUCG*	AUCG	BN-UCGAIN*	BN-UCGAIN	HKO	AUC	AUCG*	AUCG	BN-UCGAIN*	BN-UCGAIN
HKO	CFP		0.98	1.00	1.00	1.00	1.00		0.18	0.35	0.36	0.40	0.42
HKO	FP_{TF}		0.96	1.00	1.00	1.00	1.00		0.25	0.41	0.45	0.44	0.45
AUC	CFP	0.02		0.99	0.99	1.00	1.00	–0.18		0.19	0.21	0.25	0.28
AUC	FP_{TF}	0.04		1.00	0.97	1.00	0.99	–0.25		0.19	0.12	0.20	0.17
AUCG*	CFP	<0.01	0.01		0.71	1.00	0.98	–0.35	–0.19		0.05	0.12	0.14
AUCG*	FP_{TF}	<0.01	<0.01		0.22	0.90	0.53	–0.41	–0.19		–0.04	0.04	0.01
AUCG	CFP	<0.01	0.01	0.30		0.84	1.00	–0.36	–0.21	–0.05		0.08	0.08
AUCG	FP_{TF}	<0.01	0.03	0.79		0.88	0.99	–0.45	–0.12	0.04		0.04	0.04
BN–UCGAIN*	CFP	<0.01	<0.01	<0.01	0.17		0.49	–0.40	–0.25	–0.12	–0.08		–0.01
BN–UCGAIN*	FP_{TF}	<0.01	<0.01	0.10	0.12		0.24	–0.44	–0.20	–0.04	–0.04		–0.01
BN–UCGAIN	CFP	<0.01	<0.01	0.02	<0.01	0.52		–0.42	–0.28	–0.14	–0.08	0.01	
BN–UCGAIN	FP_{TF}	<0.01	0.01	0.48	<0.01	0.77		–0.45	–0.17	–0.01	–0.04	0.01	

We used the tool to analyze the names of use cases and extract 13 syntactic linguistic features considered in the method. Then, in each run of the cross-validation procedure, we trained a classifier based on the features extracted from use cases in the training set. As in the original study, use cases were classified into four size classes determined based on the quantiles of use-case functional size distribution. We used the median size of use cases belonging to a given size class to approximate the size of use cases belonging to that class.

8.1. Validating prediction capabilities of the methods

In the first step of validation, we investigated if all of the considered approximation methods are more accurate than random guessing. As is shown in Table 4, all of the methods, including HKO and AUC, outperformed random guessing by 50–79% (SA: $\mu = 70.1\%$, SD = 10.2). Also, all of the $\Delta SA_{5\%}$ values were greater than zero ($\Delta SA_{5\%}$ ranged between 20–49%, $\mu = 40.2\%$, SD = 10.2). Therefore, we can conclude that it is highly probable that all of the considered methods are not guessing, but truly predicting functional size.

8.2. Comparing the accuracy of the methods

The highest accuracy was observed for BN-UCGAIN. Its standardize accuracy (SA) was higher than AUC's by 16.83% and 13.14% for COSMIC and FP_{TF}, respectively. Also, the observed predictions errors were lower for BN-UCGAIN than for AUC. When BN-UCGAIN was compared with the HKO method, the differences were even bigger. SAs for BN-UCGAIN were higher by 27.43% for COSMIC and 24.93% for FP_{TF}. Based on the results of Wilcoxon signed-rank tests for AR, reported in Table 5, we could state that the observed differences were unlikely due to chance. All of the p-values would allow rejecting the null hypotheses with $\alpha \leq 0.05$. The corresponding magnitude of the effect size for these comparisons could be interpreted as "small" for comparison with AUC (Cliff's δ were equal -0.28 and -0.17; $|\delta| < 0.33$) and "medium" for comparison with the HKO method (Cliff's δ equal to -0.42 and -0.45; $0.33 \leq |\delta| < 0.474$).

We obtained similar results for the comparison between AUCG and AUC. All of the observed differences between ARs seemed unlikely due to chance (p-values ≤ 0.05). The observed effect sizes could be interpreted as "small" for COSMIC ($\delta = -0.21$; $|\delta| < 0.33$) and negligible for FP_{TF} ($\delta = -0.12$; $|\delta| < 0.147$). In the case of the comparison with the HKO method the observed effect size was greater and could be interpreted as "medium" ($\delta = -0.36$ and -0.45; $0.33 \leq |\delta| < 0.474$).

The variants with the a priori known categories of use–case goals (AUCG* and BN-UCGAIN*) seemed to perform similarly to AUCG and BN-UCGAIN. None of the observed differences could be considered as statistically significant. Also, the observed magnitude of effect size might be interpreted as "negligible" ($|\delta| < 0.147$).

8.3. Discussion of results

We used Hasse diagrams [75] to summarize our findings related to the accuracy of the methods. While constructing the diagrams, we assumed that there is a partial preference between P_i and P_j ($P_i \succ P_j$) if the observed p-value for the one-tailed Wilcoxon signed-rank test[4] was ≤ 0.05 and Cliff's δ suggested at least "small" magnitude of effect size (NNT ≤ 6.8). We believe that even "small" effect size could have a practical meaning in this case. Especially, if we consider the limited amount of information available for the methods.

Based on the diagrams presented in Fig. 8, we conclude that the proposed methods seem to outperform AUC for both COSMIC and FP_{TF}. The only observed exception was the AUCG method in the context of FP_{TF}. However, in that case, the observed p-value was equal to 0.03 and the Cliff's δ was equal to -0.12 ($|\delta| < 0.147$).

As expected, the HKO method was outperformed by all of the considered approximation methods. This result is not surprising because the method was not intended to be used to approximate functional size based on use-case names. The quantitative approach it employs seems well-suited for processing requirements with longer, unstructured descriptions. Unfortunately, use-case names are usually expressed by single sentences (with a common structure). The results of the study suggest that using qualitative features to build predictive models seems to be a better approach for classifying use-case names.

9. Threats to validity and study limitations

There are several threats to the validity and limitations of this study that should be taken into account while interpreting the results of the study. We will discuss these threats based on the guidelines provided by Wohlin et al. [76].

9.1. Construct validity

There are several threats related to construct validity. The first one regards the lack of formal standards on how to author use cases. Use cases can differ visibly among organizations, especially

[4] We decided to rely on the comparison between ARs rather than directly comparing SAs because SA is calculated using sample's MAR and do not take into account uncertainty related to potential differences between the sample and the population it comes from.

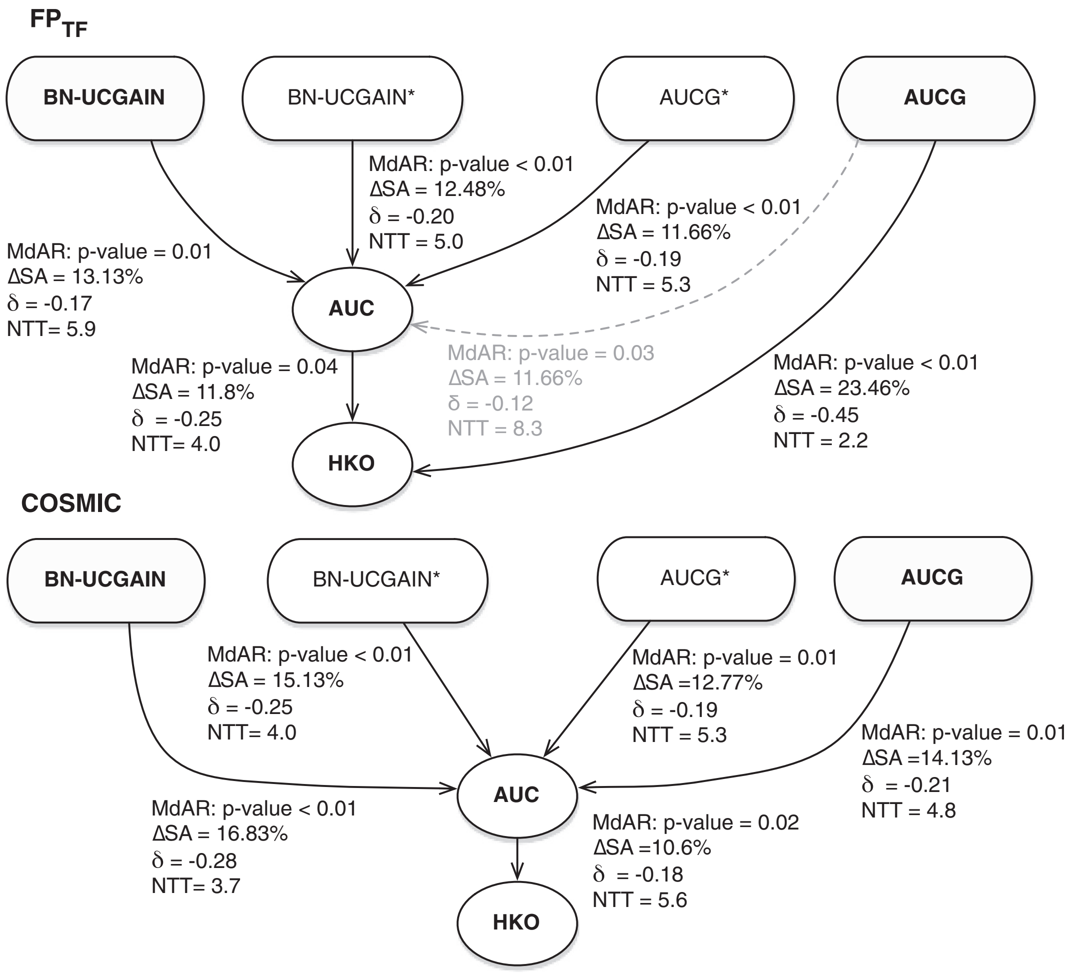


Fig. 8. Hasse diagram showing preference relationships between the functional size approximation methods.

concerning their level of granularity. That might cause difficulties in measurement and necessity of calibrating use-case based approximation methods [11,77,78]. We based our view on use cases on the state-of-the-art books by Cockburn [22] and Adolph et al. [21]. The user-level use cases in the considered sample of projects seemed convergent with the concept of use cases given by these books. Unfortunately, we cannot state without a doubt that this is the only valid way of authoring user-level use cases.

The second group of potential threats to validity relates to the subjectivity of functional size measurement methods. Both COSMIC and IFPUG FPA are well-documented methods. However, measurement specialists have to interpret the rules in the context of a given application. As a result, they can unintentionally bias the results. In the study, the functional size of the applications was measured by the author of the paper who is an experienced IFPUG Certified Function Point Specialist. However, although having a certified professional measured the functional size of application mitigates the risk, it obviously does not eliminate it.

Another threat to construct validity relates to the availability and quality of the input data for measurement. The availability of artifacts differed between the projects. In 17 out of 26 projects, use cases were augmented with the information about the flow of application screens. Also, for 9 of the projects we had access to screen designs or working application. Although we decided not to measure the size of data in the case of IFPUG FPA method, the information about data models was still valuable. Detailed data models were available for 14 projects and more business data models for 7 projects. For the remaining 5 projects, we had to derive the data models based on other artifacts (e.g., description of the product, use cases, other requirements).

There is also a threat related to the process of identifying a dominant transaction in use cases consisting of more than one transaction. The dominant transaction was determined based on the analysis of use-case scenarios and the name of a use case. We encountered only two types of multi-transaction use cases: use cases with a single dominating transaction or CRUD/partial-CRUD use cases. There are at least two threats to validity related to the process of determining a dominant transaction. First of all the process is subjective. Therefore, there is a chance of misjudging the role of a given transaction in a use case. The second relates to the fact that there could be more types of multi-transaction use cases with many, equally important transactions than the observed in the study CRUD-like use cases.

An additional threat to construct validity relates to the implementation of the HKO method. We reimplemented the method based on its description in the author's original papers [14,74]. We also used the same software packages, e.g., Stanford Parser and WEKA. The method requires some expert-based tuning when selecting the probability thresholds for constructing lists of keywords. However, according to our trials, this step did not have a visible impact on the results.

9.2. Internal validity

We identified two threats related to the internal validity of the study. The first one relates to the evaluation of the prediction accuracy of use-case goals. The proposed NLP tools were capable of analyzing use-case names expressed in English while most of the use cases were written in Polish. Therefore, some of the names had to be translated. We tried to translate them literary even when the names were not convergent with the guidelines for writing use cases.

Another issue relates to the homogeneity of the projects' data set. Most of the size approximation methods have to be calibrated locally to be used effectively [11]. Therefore, it is assumed that the learning data sets are homogeneous. Homogeneity of the data set also allows limiting the influence of potential confounding factors. As we stated in Section 4, the data set considered in the study contains data from projects developed by different organizations; however, the use cases share many similarities concerning use-case writing style. The main feature that differentiates use cases in the projects is the level of details when describing the system internal processing (for more details, see Section 2.1). The gray-box style of use cases was the most common in the data set (13 projects). Nearly similarly popular was the black-box style (11 projects). Finally, there were only 2 projects with the occurrence of white-box-style use cases. However, we expect that these differences should not have a visible impact on applying COSMIC or IFPUG FPA methods.

Another argument for homogeneity of the data is the fact that we observed relatively small prediction errors, even for the less accurate methods.

9.3. Conclusion validity

One of the threats concerns the number of statistical tests performed in the study. We used two Kruskal-Wallis tests to check if there is a difference in size of use cases depending on the category of use-case goal (a single test per measure). To mitigate this problem, we set the significance level α to 0.05 and then used the Holm–Bonferroni method [65] to control the familywise error rate (FWER) at the same level.

The same problem relates to the number of statistical tests that were performed during the evaluation of the accuracy of the approximation methods. We performed two one-tailed Wilcoxon signed-rank tests per each pair of approximation methods and functional size measures. However, to examine our main hypothesis that the proposed automatic methods predict functional size with a higher accuracy than AUC and HKO, we needed to perform only eight statistical tests (2 proposed methods $\times$ 2 methods to compare with $\times$ 2 functional size measures). After adjusting p-values for these tests with the Holm-Bonferroni method, all of them would still allow rejecting null hypotheses with $\alpha = 0.05$. Nevertheless, we believe that the results of these statistical tests should be taken with caution.

Another potential threat relates to sample size. The considered data set included only twenty-six data points what limits the possibility of detecting statistically significant differences if they truly exist. Unfortunately, this problem seems typical for studies involving predictions based on use cases, and especially to those requiring access to full documentation of use cases.

9.4. External validity

The most important threat to external validity relates to the potential influence of different approaches to elicit and document use cases. For instance, in the study, we assumed that user-level use cases should be convergent with the OTOPOP rule and the accepted guidelines for writing use cases. Unfortunately, we lack approved standards for writing use cases. As a result, use cases can differ visibly between and within organizations [11,77,78]. For instance, some organization might create more abstract use cases than the user-level ones. As a result, each of these use cases might concern multiple, similarly important user goals. In such situation, the proposed approach might not provide satisfactory results even if calibrated locally.

Another threat relates to the completeness of the proposed categories of use-case goals. Although we aimed at proposing a categorization scheme that is independent of domain, we evaluated it only for software products that could be characterized as business applications. Therefore, we have to accept the threat that use cases might differ in other domains in such a way that it would affect the applicability of the proposed methods.

In the context of Function Point Analysis, a visible limitation of the proposed methods is that they allow approximating functional size expressed in FP_{TF} instead of the regular FP. Consequently, it limits the possibility of using them for benchmarking or effort estimation (i.e., determining productivity based on historical data) unless an organization collects and stores FP_{TF} for its projects.

The necessity of collecting historical data about the functional size of use cases imposes another limitation of the proposed methods. To use the methods, organizations have to collect information about use cases, i.e., their names, the categories of their goals, types of transactions, and functional size (either COSMIC or FP_{TF}). Therefore, even if organizations are already collecting data related to functional measurement, they may need to revisit theirs past projects to supplement the missing information.

10. Conclusions

The main finding of the study is that information carried by the names of use cases might be used to support early approximation of COSMIC and IFPUG FPA functional size. Moreover, the size can be approximated in an automatic way.

The analyses reported in the paper allow us to provide the following answers to the research questions stated in the introduction:

RQ1: *Is it possible to propose a categorization scheme of use-case goals that would support functional size approximation based on use-case names?*

The proposed categorization scheme that relates goals of use cases to the dominant transactions in their scenarios seems effective in discriminating between use cases of different functional size. The on-average difference between use cases from different categories seemed significant for 54–59% pairs of categories. The observed standardized effect size was "medium" (Cliff's δ). The proposed categories of use-case goals seem independent of a business domain; therefore, they might be applicable in different project contexts.

RQ2: *How to automatically categorize use-case names (expressed in the natural language) according to the proposed categories of use-case goals?*

Use-case names are expressed in a natural language. They usually have a simple form of a single, well-structured sentence. Therefore, it seems more appropriate to categorize them based on qualitative features, such as the meaning of sentence predicates, rather than using quantitative measures, such as the number of words. The proposed method of automatic categorization employs natural language processing tools to derive a set of qualitative features from use-case names, including the analysis of the meaning of the predicates of sentences. It uses a set of decision rules and C4.5 decision trees to categorize use cases in an automatic way. The method was evaluated on a set of 427 use cases coming from 26 software projects. The prediction accuracy of the proposed method seems promising. The accuracy was at the level of 0.78, recall 0.73, precision 0.77, and F-score 0.74.

RQ3: *How to automatically approximate COSMIC and IFPUG FPA functional size of an application based on use-case names labeled with the categories of goals and historical data concerning functional size measurement?*

The proposed categorization scheme can be used to develop different prediction methods. In the paper, we propose two such methods. The first one is called AUCG. It mimics the idea of average use-case approximation (AUC) and approximate size based on average size of use cases belonging to a certain category. The second one, called BN-UCGAIN, has a form of Bayesian Network. The prediction accuracy of the methods was evaluated on the previously mentioned sample of projects, and compared to the accuracy of AUC and the HKO methods. The results showed that both proposed methods— AUCG (a direct derivative of AUC) and more advanced BN-UCGAIN, seem to outperform AUC and HKO. The observed differences in prediction errors were statistically significant. The observed standardized effect sizes ranged from “small” to “medium”.

We believe that the proposed approach can support practitioners in approximating the functional size and may help in estimating effort at early stages of software development. Assuming that historical data concerning functional size measurement is available, it allows approximating sizes of applications as soon as the names of use cases are known. Therefore, it seems to be a good alternative to the AUC method. The approach is automatic and allows reestimating the functional size whenever use-case model changes.

The presented results open some new directions of research. The first one relates to use-case goals categorization scheme. Although we considered some possibilities of simplifying the categorization scheme, we were not able to find any scheme that would allow achieving a higher accuracy of the functional size approximation. Therefore, it could be beneficial to explore other possibilities of classifying goals of use cases, also including domain-specific types of goals.

Another interesting direction for future research is to investigate the possibility of applying a similar approach to approximate size and estimate effort using the popular use-case-based methods, such as Use Case Points [79].

Acknowledgments

I would like to thank Professor Jerzy Nawrocki, Krystyna Ciesielska, Sylwia Kopczyńska, and Michał Maćkowiak for their valuable remarks and help in improving the paper.

Appendix A. Prediction performance measures

The prediction performance measures used in Section 6.1 are calculated according to Eqs. A.1–A.5. The measures can be derived from confusion matrices, such as the one presented in Fig. A.9.

The formula for calculating accuracy (see Eq. A.1) is the same for the micro (a single category) and macro levels of assessment (overall performance). Eqs. A.2–A.5 give formulas for calculating recall, precision, specificity, and F-score at the micro level. Their macro-level variants are calculated as average values by making the *positive* category each of the possible categories in turn.

$$Accuracy = \frac{\sum_{i=1}^{n} I(t_i = p_i)}{n} \tag{A.1}$$

where

- n is the number of use cases,
- t is a vector of the actual categories of use cases,
- p is a vector of the predicted categories of use cases,
- I is a function returning 1 if its argument is true and 0 otherwise.

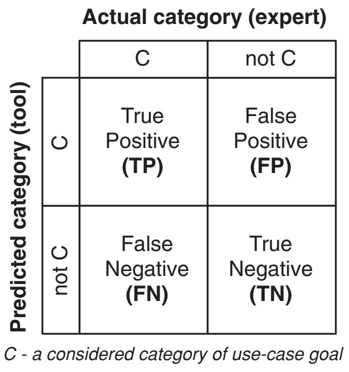


Fig. A1. Confusion matrix. *T (true)/F (false) indicates whether the output of the predicting tool is/is not consistent with the expert's opinion; P (positive)/N (negative) — the category predicted by the tool is/is not C.*

$$Recall = \frac{\sum TP}{\sum TP + \sum FN} \tag{A.2}$$

$$Precision = \frac{\sum TP}{\sum TP + \sum FP} \tag{A.3}$$

$$Specificity = \frac{\sum TN}{\sum FP + \sum TN} \tag{A.4}$$

$$F - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{A.5}$$

References

[1] H.A. Levine, Project Portfolio Management: A Practical Guide to Selecting Projects, Managing Portfolios, and Maximizing Benefits, John Wiley & Sons, 2007.
[2] ISO/IEC, Information technology — Software measurement — Functional size measurement — Part 1: Definition of concepts, 2007.
[3] A. Albrecht, Measuring application development productivity, Proc. Joint SHARE/GUIDE/IBM Appl. Dev. Symp. (1979) 83–92.
[4] ISO/IEC, ISO/IEC 24570:2005: Software engineering — NESMA functional size measurement method version 2.1 — Definitions and counting guidelines for the application of Function Point Analysis, 2005.
[5] ISO/IEC, ISO/IEC 29881:2010: Information technology — Systems and software engineering — FiSMA 1.1 functional size measurement method, 2010.
[6] C. Symons, Software Sizing and Estimating: Mk II FPA (Function Point Analysis), Wiley-Interscience, 1991.
[7] ISO/IEC, ISO/IEC 19761:2011: Software engineering – COSMIC: a functional size measurement method, 2003.
[8] OGC, Managing Successful Projects with PRINCE2, The Stationery Office, 2009.
[9] OGC, Managing of portfolios, The Stationery Office, 2011.
[10] OMG, OMG Unified Modeling Language^{TM}(OMG UML), superstructure, version 2.3, 2010,
[11] COSMIC, COSMIC Guideline for Early or Rapid Functional Size Measurement using approximation approaches, edited by F. Vogelezang, 2015, 10.13140/RG.2.1.4195.0567
[12] COSMIC, The COSMIC Functional Size Measurement Method version 3.0 Advanced and Related topics, edited by A. Lesterhuis and C. Symons, 2007,
[13] A. Živkovič, I. Rozman, M. Heričko, Automated software size estimation based on Function Points using UML models, Inf. Software Technol. 47 (13) (2005) 881–890.
[14] I. Hussain, L. Kosseim, O. Ormandjieva, Approximation of COSMIC functional size to support early effort estimation in Agile, Data & Knowledge Eng. 85 (2013) 2–14.
[15] M. Ochodek, J. Nawrocki, Enhancing use-case-based effort estimation with transaction types, Found. Comput. Decis. Sci. 35 (2) (2010) 91–106.
[16] M. Ochodek, J. Nawrocki, K. Kwarciak, Simplifying effort estimation based on Use Case Points, Inf. Software Technol. 53 (3) (2011) 200–213, doi:10.1016/j.infsof.2010.10.005.
[17] J. Jurkiewicz, J. Nawrocki, Automated events identification in use cases, Inf. Software Technol. 58 (2015) 110–122.
[18] C. Neill, P. Laplante, Requirements Engineering: The State of the Practice, Software, IEEE 20 (6) (2003) 40–45.
[19] S. Tiwari, A. Gupta, A systematic literature review of use case specifications research, Inf. Software Technol. 67 (2015) 128–158.
[20] NESMA, FPA applied to UML/Use cases, version 1.0, 2008.
[21] S. Adolph, P. Bramble, A. Cockburn, A. Pols, Patterns for Effective Use Cases, Addison-Wesley, 2002.

[22] A. Cockburn, Writing Effective Use Cases, Addison-Wesley Boston, 2001.
[23] G. Övergaard, K. Palmkvist, Use Cases: Patterns and Blueprints, Addison-Wesley, 2004.
[24] ISO/IEC, ISO/IEC 20926:2009: Software and systems engineering — Software measurement — IFPUG functional size measurement method 2009, 2009.
[25] T. Fetcke, A. Abran, T. Nguyen, Mapping the OO-Jacobson approach into Function Point Analysis, in: Proceedings of TOOLS 23, 1997, pp. 192–202, doi:10.1109/TOOLS.1997.654721.
[26] B. Bernárdez, A. Durán, M. Genero, An empirical review of use case metrics for requirements verification, in: Proceedings of SMEF'04, Rome, Italy, 2004.
[27] T. Iorio, IFPUG Function Point analysis in a UML framework, in: Proceedings of SMEF'04, Rome, Italy, 2004.
[28] G. Cantone, D. Pace, G. Calavaro, Applying function point to Unified Modeling Language: Conversion model and pilot study, in: Proceedings of 10th International Symposium on Software Metrics, IEEE, 2004, pp. 280–291, doi:10.1109/METRIC.2004.1357912.
[29] V. Harput, H. Kaindl, S. Kramer, Extending Function Point Analysis to bject-oriented requirements specifications, in: 11th IEEE International Symposium on Software Metrics, IEEE, 2005, pp. 10–pp.
[30] B. Kitchenham, K. Känsälä, Inter-item correlations among Function Points, in: Software Engineering, 1993. Proceedings., 15th International Conference on, IEEE, 1993, pp. 477–480.
[31] L. Lavazza, S. Morasca, G. Robiolo, Towards a simplified definition of Function Points, Inf. Software Technol. 55 (10) (2013) 1796–1809.
[32] COSMIC, The COSMIC Functional Size Measurement Method v4.0.1, Measurement Manual, 2015.
[33] B. Marín, G. Giachetti, O. Pastor, Measurement of functional size in conceptual models: A survey of measurement procedures based on COSMIC, in: Software Process and Product Measurement, Springer, 2008, pp. 170–183.
[34] P. Habela, E. Głowacki, T. Serafinski, K. Subieta, COSMIC Function Points: Theory and Advanced Practices, CRC Press.
[35] M. Jenner, COSMIC-FFP and UML: Estimation of the Size of a System Specified in UML–Problems of Granularity, in: Proc. the Fourth European Conference on Software Measurement and ICT Control, 2001, pp. 173–184.
[36] V. Bévo, G. Lévesque, A. Abran, Application de la methode FFP a partir d'une specification selon la notation UML: Compte rendu des premiers essais d'application et questions, 9th International Workshop Software Measurement, Lac Supérieur, Canada, 1999.
[37] A. Sellami, H. Ben-Abdallah, Functional size of use case diagrams: a fine-grain measurement, in: Proceedings of ICSEA'09, IEEE, 2009, pp. 282–288, doi:10.1109/ICSEA.2009.96.
[38] F. Vogelezang, T. Prins, S.N. BV, Approximate size measurement with the COSMIC method Factors of influence, Proc. SMEF'07 (2007) 167–178.
[39] NESMA, The application of Function Point Analysis in the early phases of the application life cycle — A practical manual: Theory and case study, version 2.0, 2005.
[40] T. Iorio, R. Meli, F. Perna, Early & Quick Function Points® v3.0: enhancements for a Publicly Available Method, in: Proceedings of SMEF'07, 2007, pp. 179–198.
[41] M. Conte, T. Iorio, R. Meli, L. Santillo, E&Q: An Early & Quick Approach to Functional Size Measurement Methods, in: Proceedings of SMEF'04, Rome, Italy, 2004.
[42] L. Santillo, Easy Function Points–'Smart' Approximation Technique for the IFPUG and COSMIC Methods, in: Proceedings of IWSM-MENSURA 2012, IEEE, 2012, pp. 137–142, doi:10.1109/IWSM-MENSURA.2012.29.
[43] F. Valdés, A. Abran, Industry case studies of estimation models based on fuzzy sets, in: Proceedings of IWSM-MENSURA 2007, 2007, pp. 5–9.
[44] G. De Vito, F. Ferrucci, Approximate COSMIC Size: The Quick/Early Method, in: Software Engineering and Advanced Applications (SEAA), 2014 40th EUROMICRO Conference on, IEEE, 2014, pp. 69–76.
[45] S. Bagriyanik, A. Karahoca, Automated COSMIC Function Point measurement using a requirements engineering ontology, Inf. Software Technol. 72 (2016) 189–203. http://dx.doi.org/10.1016/j.infsof.2015.12.011.
[46] A.R. Hevner, S.T. March, J. Park, S. Ram, Design science in information systems research, MIS quarterly 28 (1) (2004) 75–105.
[47] R. Wieringa, Design Science Methodology for Information Systems and Software Engineering, Springer, 2014.
[48] M. Ochodek, B. Alchimowicz, J. Jurkiewicz, J. Nawrocki, Improving the reliability of transaction identification in use cases, Inf. Software Technol. 53 (8) (2011) 885–897.
[49] M. Shepperd, S. MacDonell, Evaluating prediction systems in software project estimation, Inf. Software Technol. 54 (8) (2012) 820–827. http://dx.doi.org/10.1016/j.infsof.2011.12.008.
[50] W.B. Langdon, J. Dolado, F. Sarro, M. Harman, Exact Mean Absolute Error of Baseline Predictor, MARP0, Inf. Software Technol. 73 (2016) 16–18. http://dx.doi.org/10.1016/j.infsof.2016.01.003.
[51] B.A. Kitchenham, L.M. Pickard, S.G. MacDonell, M.J. Shepperd, What accuracy statistics really measure? in: IEE Proceedings—Software Engineering, 148, IET, 2001, pp. 81–85.
[52] N. Cliff, Dominance statistics: Ordinal analyses to answer ordinal questions., Psychol. Bull. 114 (3) (1993) 494.
[53] M.R. Hess, J.D. Kromrey, Robust confidence intervals for effect sizes: A comparative study of Cohen's d and Cliff's delta under non-normality and heterogeneous variances, Annual meeting of the American Educational Research Association, San Diego, 2004.
[54] H.C. Kraemer, D.J. Kupfer, Size of treatment effects and their importance to clinical research and practice, Biol. psychiatry 59 (11) (2006) 990–996.
[55] J. Romano, J.D. Kromrey, J. Coraggio, J. Skowronek, Appropriate statistics for ordinal level data: Should we really be using *t*-test and Cohen's d for evaluating group differences on the NSSE and other surveys, in: Annual meeting of the Florida Association of Institutional Research, 2006, pp. 1–33.
[56] J. Cohen, Statistical power analysis for the behavioral sciences, 1977,
[57] S.D. Martino, F. Ferrucci, C. Gravino, F. Sarro, Web effort estimation: Function point analysis vs. {COSMIC}, Inf. Software Technol. 72 (2016) 90–109. http://dx.doi.org/10.1016/j.infsof.2015.12.001.
[58] S. Kopczyńska, J. Nawrocki, M. Ochodek, Software development studio—Bringing industrial environment to a classroom, in: Proceedings of EduRex 2012, IEEE, 2012, pp. 13–16, doi:10.1109/EduRex.2012.6225698.
[59] ISO/IEC, Information technology — Software measurement — Functional size measurement — Part 5: Determination of functional domains for use with functional size measurement, 2004.
[60] S. Diev, Software estimation in the maintenance context, ACM SIGSOFT Software Eng. Notes 31 (2) (2006) 1–8.
[61] C. Rolland, C. Achour, Guiding the construction of textual use case specifications, Data Knowledge Eng. 25 (1) (1998) 125–160.
[62] J. Jurkiewicz, J. Nawrocki, M. Ochodek, T. Głowacki, HAZOP-based identification of events in use cases, Empirical Software Eng. 20 (1) (2015) 82–109.
[63] M. Ridao, J. Doorn, J. do Prado Leite, Domain independent regularities in scenarios, in: Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on, IEEE, 2001, pp. 120–127.
[64] A. Issa, M. Odeh, D. Coward, Using use case patterns to estimate reusability in software systems, Inf. Software Technol. 48 (9) (2006) 836–845.
[65] S. Holm, A simple sequentially rejective multiple test procedure, Scand. J. Stat. (1979) 65–70.
[66] W.J. Conover, Practical Nonparametric Statistics, 3rd, John Wiley & Sons, 1999.
[67] M.-C. De Marneffe, B. MacCartney, C.D. Manning, et al., Generating typed dependency parses from phrase structure parses, in: Proceedings of LREC, 6, 2006, pp. 449–454.
[68] G.A. Miller, WordNet: a lexical database for English, Commun. ACM 38 (11) (1995) 39–41.
[69] S. Bird, E. Klein, E. Loper, Natural language processing with Python, "O'Reilly Media, Inc.", 2009.
[70] D. Lin, An information-theoretic definition of similarity., in: ICML, 98, 1998, pp. 296–304.
[71] J.R. Quinlan, C4. 5: programs for machine learning, Elsevier, 2014.
[72] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The WEKA data mining software: an update, ACM SIGKDD explor. newslett. 11 (1) (2009) 10–18.
[73] T.D. Nielsen, F.V. Jensen, Bayesian Networks and decision graphs, Springer Science & Business Media, 2009.
[74] I. Hussain, L. Kosseim, O. Ormandjieva, Using linguistic knowledge to classify non-functional requirements in SRS documents, in: Natural Language and Information Systems, Springer, 2008, pp. 287–298.
[75] B.A. Davey, H.A. Priestley, Introduction to lattices and order, Cambridge university press, 2002.
[76] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in Software Engineering, Springer Science & Business Media, 2012.
[77] F. Vogelezang, C. Symons, A. Lesterhuis, R. Meli, M. Daneva, Approximate COSMIC Functional Size–Guideline for Approximate COSMIC Functional Size Measurement, in: Proceedings of IWSM-MENSURA 2013, IEEE, 2013, pp. 27–32.
[78] V. Del Bianco, L. Lavazza, G. Liu, S. Morasca, A.Z. Abualkishik, Model-based early and rapid estimation of COSMIC functional size–An experimental evaluation, Inf. Software Technol. 56 (10) (2014) 1253–1267.
[79] G. Karner, Metrics for objectory. No. LiTH-IDA-Ex-9344:21, Master's thesis, University of Linköping, Sweden, 1993.