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/

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:

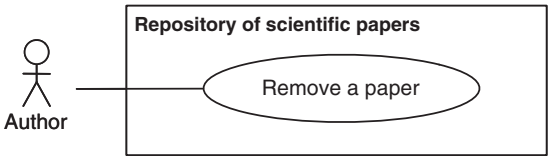
RQ1: Is it possible to propose a categorization scheme of use-case goals that would support functional size approximation based on use-case names?
RQ2: How to automatically categorize use-case names (expressed in a natural language) according to the proposed categories of use-case goals?
RQ3: How to 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.