A Recommendation System of Reuse Opportunities based on Lexical Analysis

Johnatan Oliveira, Eduardo Figueiredo
Software Engineering Laboratory (LabSoft), Department of Computer Science
Federal University of Minas Gerais (UFMG)
Belo Horizonte, Brazil
{johnatan-si, figueiredo}@dcc.ufmg.br

ABSTRACT

Software reuse is a development strategy in which existing software components, called reusable assets, are used in the development of new software systems. Extraction methods may be used in different contexts including derivation of software product lines. However, few methods have been proposed in literature for reusable asset extraction and recommendation of these reuse opportunities. In this paper, we propose a method for extraction of reuse opportunities based on naming similarity of three types of object-oriented entities: classes, methods, and attributes. These entities compose a repository with reuse opportunities. The proposed method should be integrated into a recommendation system that aims to assist developers by: (i) recommending a general partial design for a specific domain and (ii) indicating reuse opportunities. At the moment, we have a prototype tool to support our method in extracting reuse opportunities. We evaluate the method and tool with 38 e-commerce information systems mined from GitHub. As a result, we observe that our method is able to identify classes and methods that are relevant in the e-commerce domain.

1. INTRODUCTION

Software reuse is a development strategy in which existing software components, called reusable assets, are used in the development of new software systems [6]. It has been studied and pointed as an alternative to traditional development aiming to increase software quality and decrease development efforts by using previously developed, and sometimes already tested, software components [9, 11]. One of the many techniques to support software reuse is Software Product Line (SPL). An SPL is a set of software systems that share a common set of features, then new software may be developed using these commonalities [8].

Considering the continuous software development in companies, the increasing number of developed artifacts may prevent the identification of reusable assets to support a SPL derivation. Therefore, methods to provide recommendations of reusable assets are desirable to assist developers. In this context, a recommendation system could automatically identify reusable assets and suggest these artifacts for developers, helping to increase the capacity and effectiveness of the nominating process and manipulation of large volumes of data [12]

In general, recommendation systems are based on content filtering given a data requested. In this study, we propose a recommendation system that aims to retrieve reuse opportunities. Therefore, a method for extraction of these reuse opportunities is required to provide inputs for the recommendation system. Some methods have been proposed to support the extraction of reuse opportunities from software systems [7, 12].

In literature, many different approaches are used for identification of reuse opportunities, such as natural language processing [10], formal specifications [1], machine learning [3], and other Information Retrieval (IR) approaches [6]. However, to the best of our knowledge, we did not find a method for extraction of reuse opportunities and recommendation of such opportunities considering most frequent elements, such as classes, methods, and attributes, from systems of the same domain.

In this paper, we present the first stage for development of a recommendation system to support software reuse. This stage consists of developing a method for extraction of reuse opportunities, called JReuse. Given a set of software systems, JReuse aims to identify methods named with the same token in lexical similarly named classes from different systems. The main goal of JReuse is to identify classes, methods, and eventually attributes, that may be recommended as reuse opportunities. We also present a prototype tool that implements our method.

We conduct an evaluation of our method through an experiment with 38 Java information systems in the e-commerce domain mined from GitHub. As a result, we observe that our method is able to identify reuse opportunities using naming similarity analysis. We also observe that JReuse provides meaningful classes and methods in the analyzed domain that may be indicated as reuse opportunities to developers of new e-commerce systems.

The remainder of this paper is organized as follows. Section 2 presents the proposed approach. Section 3 presents a preliminary evaluation. Section 4 discusses related work. Finally, Section 5 concludes this paper.

2. THE PROPOSED APPROACH

We proposed a research methodology divided in seven study steps as presented in Figure 1. The steps colored in green have been completed. The yellow step is ongoing. The red step describes how we intend to evaluate the recommendation system that we aim to develop. We designed all steps to provide us background about the research topic, related work, recommendation systems, and other concepts that are relevant to our research. We discuss each study step as follows.

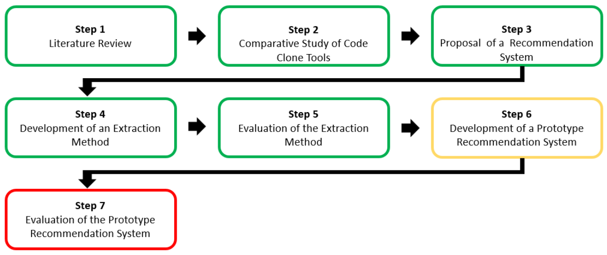
Step 1
Literature Review
Step 2
Comparative Study of Code Clone Tools
Step 3
Proposal of a Recommendation System
Step 4
Development of an Extraction Method
Step 5
Evaluation of the Extraction Method
Step 6
Development of a Prototype Recommendation System
Step 7
Evaluation of the Prototype Recommendation System

Figure 1: Study Steps

Step 1: In this step, we conducted a Systematic Literature Review [4] to know about the topic of research, strategies for reusable assets identification, and available tools to provide the appropriate structure for reuse of software artifacts. This step provided us related work and a comparison of researches that explores reuse opportunities in software systems.

Step 2: We performed a comparison of code clone detection tools to assess whether these tools are able to identify similar source code in different software systems. We also evaluated plagiarism detection tools from systems of the same domain. In this step, we observed some desirable points in methods for extraction of reusable assets. Once none of the tools we found fitted to our purpose [2], we considered the desirable points to propose a method for extraction and recommendation of reuse opportunities.

Step 3: In this step, we defined a strategy for the recommendation system we aim to develop. We investigated the required input, source code processing, target programming language, and the reuse opportunities repository to support the development of a recommendation system for software reuse.

Step 4: After Step 3, we were able to understand the main requirements of a method for reusable assets extraction. We used the e-commerce information system domain as reference to a preliminary study on the frequency of similarly named entities in source code: classes, methods, and attributes. The extraction method that we propose in this Step 4 receives as input a set of information systems. Then, the method processes names of classes to identify similarly named classes in different system. Considering classes with similar names, our strategy identities similarly named methods and attributes. The results of this analysis are persisted in database.

Step 5: We conducted an experiment in order to assess the effectiveness of the proposed method for reuse opportunities extraction using a data set composed of 38 information systems from the e-commerce domain. We mined these systems from GitHub. Section 3 describes the results of our preliminary evaluation.

Step 6: A recommendation system is under development to implement the approach previously described. We designed this system to identify reuse opportunities using the method proposed in Step 4, to provide a partial design of information systems, and to recommend source code for developers. Figure 2 illustrates the proposed recommendation system. The recommendation system receives different systems as input. Then, the extraction method applies a similarity function to compute similarity between entities. We defined a threshold according to empirical studies: similarity must be equal or greater than 80% for all entities. In Figure 2, the part in dashed line is a task that we have to do.

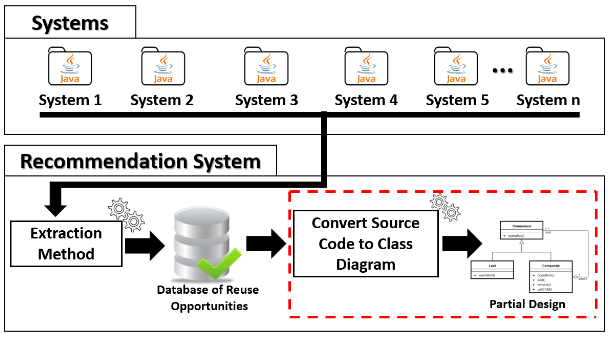

Figure 2: Recommendation System

Step 7: After the development of our recommendation system in Step 6, we are able to conduct experiments to evaluate the quality of the recommendations.

We systematically designed and conducted Steps 1 to 7 to achieve success in our research. Through this study, we are developing solutions to the problems identified in both literature and industry. The forecast for getting the Master's in Computer Science is December 2016, and we planned our study activities according to this schedule.

3. PRELIMINARY EVALUATION

In this study, we evaluate our method for extraction of reuse opportunities from systems of the same domain. The identification of attributes is under development. Therefore, let us consider, for this preliminary evaluation, only the identification of classes and methods through lexical similarity analysis. We are interested in whether JReuse is able to identify recurrent classes and methods in a specific software domain. We are also interested in assessing the relevance of results provided by our method. For this purpose, we chose the e-commerce domain for evaluation.

We analyzed the frequency of similarly named classes in the 38 e-commerce information systems from our data set. Initially, JReuse identified 38 classes as reuse opportunities. To summarize data, we adopted the following exclusion criteria for classes to be considered in this evaluation: the class should occur in at least three different systems. Since our method compares classes in pairs, two occurrences may not be significant to a reuse recommendation. We discarded 22 classes based on this minimum threshold.

We concluded that the top-five most frequent classes for ecommerce domain, with more than five occurrences, are User, Product, Order, Category, and Customer. In fact, according to the viewpoint of four independent software engineers – these classes are meaningful and relevant to the e-commerce domain. For instance, classes such as Product, Order, Category, and Customer are elementary entities to be expected in an e-commerce system. In turn, User is the most frequent class identified in this preliminary study by the tool and, although it is not specific of e-commerce domain, Costumer is an expected class in information systems in general.

Therefore, we conclude that our method is able to identify relevant classes for software reuse in 60% of the evaluated e-commerce systems (23 of the 38 systems collected from GitHub). The other less frequent classes (e.g., Item, Payment, and Cart) are also relevant to the e-commerce domain. Regarding frequent methods through systems, we filtered methods by the following inclusion criteria: it should appear in at least two classes from different systems. From 56 similarly named methods extracted, we discarded 34 according to the defined criteria, remaining 22 methods. Consider a pair (C, M) in which C is a class name, and M is a method name. For instance, the pairs (Product, getPrice) and (Product, getPrice) are the most frequent among the systems of our data set.

Considering the domain under analysis, it is meaningful, since the product abstraction requires a monetary value for the product. Moreover, the method getPassword occurs 8 times in User and 5 times in Costumer classes. This scenario is comprehensible considering that these classes intuitively represent entities that have access to the e-commerce system for purchase and payment, for instance. Therefore, in the viewpoint of the authors, the identified methods are relevant and consistent for the domain under analysis.

4. RELATED WORK

Many approaches have been proposed in literature to support the extraction of reusable assets. For instance, Kawaguchi et al. [5] propose an algorithm to categorize software projects automatically. This method aims to identify similar software systems using duplicated code detection.

Oliveira et al. [11] propose a tool for recommendation of reusable assets. Their tool applies a technique to support software reuse and extraction of reusable assets called Automatic Identification of Software Components.

In turn, our reusable asset extraction method and supporting tool aim to identify candidates for reuse from software systems from a specific domain, using lexical analysis. Our method also ranks software components identified as reusable assets by frequency in which components appear in different systems from the same domain. We expect that this approach is helpful in reuse recommendation by suggesting classes, methods, and attributes that are the most recurrent in information systems given a specific domain.

5. CONCLUSION

In this paper, we present the proposed Master's study and the steps needed to achieve success. As a preliminary study, we present in detail the method to extract reusable assets from software systems. We also present a prototype tool that implements the proposed method. Finally, we present the preliminary idea of a recommendation tool to suggest software reusable assets in a class diagram-based vision. Through our study, we were able to identify the main issues of software reuse and the reusable assets extraction process.

We evaluate our extraction prototype tool through an experiment with 38 information systems from e-commerce domain extracted from GitHub. Our findings point that the proposed method is able to suggest methods and classes that appear in different systems from the selected data set. We also observe that the most frequent methods and classes pointed by the tool as candidates to software reuse are relevant to most of the information systems from the e-commerce domain.

6. ACKNOWLEDGMENTS

This work was partially supported by CAPES, CNPq, and FAPEMIG.

7. REFERENCES

[1] G. Caldiera and V. Basili. Identifying and qualifying reusable soft. components. *Computer*, 24(2):61–70, 1991.

[2] J. A. de Oliveira, E. M. Fernandes, and E. Figueiredo. Evaluation of duplicated code detection tools in cross-project context, 2015.

[3] S. Kawaguchi, P. Garg, M. Matsushita, and K. Inoue. Mudablue: an automatic categorization system for open source repositories, 2004.

[4] S. Keele. Guidelines for performing systematic literature reviews in software engineering. In *Technical report, Ver. 2.3 EBSE Technical Report. EBSE.* 2007.

[5] C. Krueger. Soft. reuse. *Computing Surveys (CSUR)*, 24(2):131–183, 1992.

[6] A. Kuhn, S. Ducasse, and T. Gírba. Semantic clustering: Identifying topics in source code. *Info. and Soft. Tech.*, 49(3):230–243, 2007.

[7] J. Lee, K. Kang, and S. Kim. A feature-based approach to product line production planning. In *Soft. Product Lines*, pages 183–196. Springer, 2004.

[8] Y. Maarek, D. Berry, and G. Kaiser. An info. retrieval approach for automatically constructing soft. libraries. *Transactions on Soft. ENG. (TSE)*, 17(8):800–813, 1991.

[9] P. Mohagheghi and R. Conradi. Quality, productivity and economic benefits of soft. reuse: a review of industrial studies. *Empirical Soft. ENG. (ESE)*, 12(5):471–516, 2007.

[10] P. Mohagheghi, R. Conradi, O. Killi, and H. Schwarz. An empirical study of soft. reuse vs. defect-density and stability, 2004.

[11] M. Oliveira, E. Goncalves, and K. Bacili. Automatic identification of reusable soft. development assets: Methodology and tool, 2007.

[12] O. Zoeter. Recommendations in travel, 2015.