A FRAMEWORK FOR FULFILLING COMPLEX, STAKEHOLDER SPECIFIC INFORMATION REQUIREMENTS BY MINING MULTIPLE SOFTWARE REPOSITORIES

C .K. Kumarasinghe[1], C. P. Wijesiriwardana[1] and M. I. M. Nasmin[1]

[1]Department of Information Technology, Faculty of Information Technology, University of Moratuwa
Sri Lanka.
Email: kaushalyak@uom.lk, chaman@uom.lk, nasmin1990@gmail.com

ABSTRACT

Recent studies have highlighted the importance of obtaining useful and up-to-date information about a software project to its stakeholders such as developer, testers, and project managers. Providing information about a software project, which is specific to a particular stakeholder, is a challenging task. This is mainly because historical data about software projects is scattered across multiple repositories such as version control systems, bug tracking systems, and email archives. Therefore, obtaining required information is cumbersome. To tackle this problem, we propose a semi-automated approach for analyzing software projects; hence it allows providing useful and timely information to the stakeholders. This paper presents two main contributions to the software engineering community. Firstly, it consists of a comprehensive study to categorize the questions asked by software practitioners based on the task complexity, stakeholder type, and task category. We believe that, this classification could be used as a software analysis handbook for the practitioners. Secondly, our proposed approach suggests a flexible model to answer the questions and validate the accuracy of the responses. The model has been evaluated with a set of experiments; in which we obtained encouraging results.

***Keywords*:** software analysis, mining software repositories, software evolution

1. INTRODUCTION

Recent years have witnessed extensive studies on mining software repositories for extracting information related to software evolution [4]. These repositories include version controlling systems (i.e. Github, SVN), bug repositories (i.e. Bugzilla, JIRA), and build process monitoring tools (i.e. Jenkins) where the information seekers can find out various types of valuable information related to the evolution of a particular software.

There are several challenges in software evolution analysis [1]. Providing information, which is specific to a particular stakeholder, is one of the major challenges in software evolution analysis. The software development process consists of many stakeholders such as project managers, business analysts, software architects, software engineers, quality assurance engineers, who mine software repositories from their own perspectives. For an example, a project manager may be interested in finding out the workload and deadlines of the team members whereas a software engineer may want to find out the changes made by the other software engineers for the software module he/she is working on. Another challenge in software evolution analysis is the requirement of extracting data from multiple data streams. For an example, if someone wants to find the number of bugs reported between two builds, he/she should mine information from both bug repositories and build process monitoring systems. Moreover, such information should be presented to the information seeker as a cumulative output of those multiple data streams.

In this paper, we address the above challenges in current state-of-the-art research and present a software evolution analysis framework that is capable of facilitating complex, stakeholder specific information requirements that need to be answered by analyzing multiple software repositories. Moreover, we describe our experimental procedure and obtained results for software evolution analysis.

The rest of this paper is organized as follows. Section 2 reviews the related research on fulfilling the information needs of different stakeholders for software evolution analysis. The Section 3 describes the research problem addressed through this paper. The Section 4 presents the proposed service-oriented approach for mining multiple software repositories to fulfill the information requirements of different stakeholders. Section 5 presents the results and the evaluation of the proposed method followed by the discussion and future work in Section 6.

2. RELATED WORK

Fulfilling the information requirements of the stakeholders in software development lifecycle is an essential but a challenging task in software evolution analysis. Several recent studies have examined that the questions asked by different stakeholders [1] for fulfilling different types of information requirements such as information related to bug reports and source codes, progress of the software development process and quality of the software. In the first phase of our research, we analyzed those questions and categorized them according to the complexity, the type of the stakeholder and task category. In the next step, we conducted an analysis of the available tools for answering the questions (Q1 – Q16) asked by the development team.

Q1	What is the size of project x?
Q2	Who is working on project x?
Q3	Who is working on what?
Q4	What is the number of commits made by developer x in project y?
Q5	What are the coworkers working on right now?
Q6	How much work people have done?
Q7	Who changed this code?
Q8	Who to assign a code review to?
Q9	Which code review has been assigned to which person?
Q10	Who is working on the same class?
Q11	What are the changes of newly resolved work items related to me?
Q12	Who has worked with this package?
Q13	Who is accessing a particular API?
Q14	What is the average line of code of developer x per day?
Q15	Who has changed the code between two successful builds?
Q16	What is the amount of bug records between two successful builds?

Several researchers have been working on developing tools that can be used to fulfill the information requirements of the stakeholders. Brandtner et al. in [1] presents a service-oriented approach for fulfilling the information requirements of stakeholders called SQA-Mashup. This works as a quality awareness platform, which integrates information from Continuous Integration Tools according to the information requirements of stakeholders and presents as a single service. The flexibility of web service integration is achieved through a mashup-based approach [2, 3]. A mashup facilitates a pipe and filter based integration of data sources. From the backend, an integration pipe is described as a series of pipe and filter steps in which the execution of the pipe is triggered by a web service call. The tool also provides two views (from the front end) for developers and testers.

Fritz et al. in [4] presents an information fragment model and a prototyping tool that automates the composition of the required information. This introduces an information fragment model, a relatively simplified approach for answering the questions.

Despite the success of the above methods, most existing approaches suffer from several critical limitations. They typically facilitate information seekers to answer simple questions based on predefined workflows. Hence, users are unable to perform complex analysis, which require processing information from multiple software repositories. Further, these tools are designed for fulfilling information requirements of few stakeholders, which may not ultimately address the information needs of all interested parties in the software development process. In particular, according to the best of our knowledge there is no tool that is capable of answering the complex, information seeker specific questions, which are required to be answered by mining multiple software repositories.

Table 1 : Comparison of Available Tools

Feature	**Sonarqube**	**SQA-Mashup**	**SQuORE**	**Squale**
Service Oriented	✓	✗	✗	✗
CI Tool's API Integration	✗	✓	✓	✗
Customized View	✗	✗	✗	✗
Software Matrics	✓	✓	✓	✓
Data Integration	✗	✗	✗	✗

3. OUR APPROACH

To address these limitations, in this paper, we propose a semi-automated approach for analyzing software evolution. This paper provides two main contributions to the software engineering community. Firstly, it consists of a comprehensive study to categorize the questions asked by software practitioners based on the task complexity, stakeholder type and data source. We believe that this classification could be used as a software analysis handbook for the practitioners. Secondly, our proposed approach suggests a flexible model to answer the questions and validate the accuracy of the answers. Hence, our work differs from the state-of-the-art research considerably as this approach facilitates the stakeholders to answer complex, information seeker specific questions that need to be answered by mining multiple software repositories.

3.1 Our Classification

In this section, we present our classification of the questions asked by the development team. The classification was performed based on three criterions; the complexity of the task, stakeholder type, and data source.

The questions mentioned in section 2 can be categorized into two classes based on its complexity. The simple questions can be answered directly by fetching information from the corresponding data source. For an example, Q1, Q2, can be answered directly by calculating the number of lines from the software repository. Contradictorily, questions like Q5, Q15 cannot be directly answered from data sources. To answer such questions, the data fetched from the data sources should be further filtered based on the conditions.

These questions can also be categorized according to the type of stakeholder. For an example, questions such as Q2, Q3, Q15 are Project Manager specific questions while Q11, Q12, Q13 are Developer specific questions. Further a part of this question set can be answered by taking data from multiple data sources (i.e. Q16).

3.2 Tool Support

We use Application Programming Interfaces of various Continuous Integration tools to gather data. We process the data and transfer to a common data format. In the pipeline integration we bind the data to pipes. The stakeholders can drag and drop the pipes so that they can tailor their needed information. In the data extraction process we will use a web client to generate the information needed to create the default view of any stakeholders.

Data Extraction and Processing
We use the APIs of the CI-tools to gather data from the CI tools; we also implement a new module and take the information from the API. Most of the CI tools provide a rest API. Eg GitHub. We gather commit data from GitHub using its API. Jenkins will provide Build related information such as the build date and Build versions. JIRA will provide bug related data such as the fixed bugs, assignee and assign date, etc.

Service Composition
There are certain quality measures stakeholders often need. We planned to provide those quality measures in a default view. Here we use different services from the CI tools and use a better service composition to determine the software quality. The characteristics and the parameters will be changed in this domain. Therefore, we need to change the composition algorithm to include those changes in this domain.

Pipe Implementation
We have used the pipeline implementation to make the job easy. Usually, piping is used in scenarios where multiple queries need to run a data set to generate the result. Quality information from various CI tools is bound to pipes. Piping technology allows a set of pipes to combine in a way. So the data in the pipes can deliver useful information. For example, Yahoo Pipes. Piping technology provides combining any number of pipes. We use this feature to allow stakeholders to customize the software quality information according to stakeholders' preference.

Pipe Execution and Presentation
Our solution is consisting two views they are customized views and default views. Default view provides answer for the static questions that are frequently needed for different stakeholders. The customized view shows the information based on the particular stakeholders needs and interest.

Customized View
Every Stakeholder has different information need. We consider only the developer's needs of software quality information and narrow down the scope. We did a survey to find the software quality information needs of developer's and could find that they are dynamic. Therefore we provide a way to customize the query.

4. EVALUATION

We conducted an extensive set of experiments to evaluate the applicability of the above approach in which the encouraging results validated the effectiveness of the proposed method. The evaluation consists of two main parts. Due to the space limitations we will only describe how to solve a simple question and a complex question using our tool.

Q4. What is the number of commits made by developer x in project y?

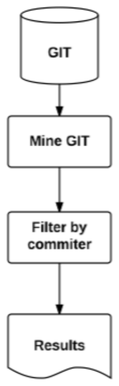

Figure 1 : Workflow to answer Q4

This question falls into the simple category and, therefore, the answer is straightforward. It requires gathering information from the Git repository and filtering the commits based on the name of the developer. Figure 1 is the high-level workflow to solve the problem.

Q16. What is the amount of bug records between two successful builds?

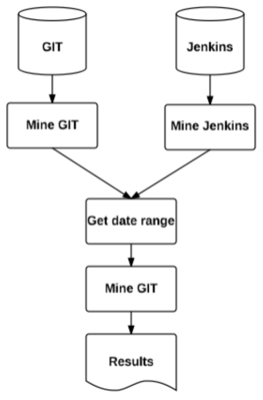

Figure 2 : Workflow to answer Q16

This question falls into the complex category and still our tool is capable of providing the solution to the interested stakeholder. It requires gathering data from Git repository and Jenkins repository. Several activities are performed in parallel and then the results are combined together. Figure 2 is the high-level workflow to solve the problem.

5. DISCUSSION AND FUTURE WORK

Several recent studies have examined the questions asked by different stakeholders in software evolution analysis. Despite the success of the above methods, most existing approaches suffer from several critical limitations. Firstly, these tools facilitate information seekers only to based on predefined workflows. Secondly, such tools are not designed to fulfill the information requirements of all the stakeholders. To tackle this problem, we propose a semi-automated approach for analyzing software projects; hence it allows providing useful and timely information to the stakeholders. This paper provides two main contributions to the software engineering community. Firstly, it consists of a comprehensive study to categorize the questions asked by software practitioners based on the task complexity, stakeholder type and task category.

6. REFERENCES

[1] M. Brandtner, E. Giger, H. Gall, *"Supporting continuous integration by mashing-up software quality information,"* Software Maintenance, Reengineering and Reverse Engineering, Software Evolution Week - IEEE Conference on, pp.184,193, 3-6 Feb. 2014

[2] L. Grammel, C. Treude and M. *"Anne Storey. 2010. Mashup environments in software engineering"* In Proceedings of the 1st Workshop on Web 2.0 for Software Engineering (Web2SE '10). ACM, New York, NY, USA, 24-25

[3] K. T. Stolee and S. Elbaum, *"Refactoring pipe-like mashups for end-user programmers"*. In Proceedings of the 33rd International Conference on Software Engineering (ICSE '11). ACM, New York, NY, USA, 81-90, 2011

[4] T. Fritz and G. C. Murphy, *"Using information fragments to answer the questions developers ask"* In Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1 (ICSE '10), Vol. 1. ACM, New York, NY, USA, 175-184, 2010