An Enhancement of Software Development Process using Reuse and Visualization Techniques

Anas Bassam AL-Badareen

Department of Software Engineering, Faculty of Information Technology, Aqaba University of Technology, Aqaba, Jordan

Published: 30 January 2021

Abstract

Software reuse is one of the main techniques used to enhance the productivity of software development and the quality of software products. This technique concerned in extracting software components from legacy systems and reusing them in the development of new software systems. The process of analyzing and understanding the legacy systems in order to identify and extract the intended components is difficult and time consuming. Whereas several methods and tools were proposed and developed in order to enhance the process of software reuse, there is no attention paid to the understandability of the legacy systems and the reusable components. This study proposes a new method for software reuse enhanced by visualization technique. It concerns about improving the understandability of software systems, and reduce time and resources required for reuse.

Keywords: *Software Development, Software Process, Software Reuse, Software Visualization.*

1. Introduction

In recent decades, software reuse and software complexity have been rapidly increased. Software reuse is considered as one of the main aspects used to enhance the productivity and reduce the cost of software development. The white box reuse is the most used technique in software reuse. It aims to retrieve and modify software components from legacy systems and use them in developing new software systems. The process of modifying software components is a difficult task that needs to understand the whole legacy system in order to identify the requirements of the new modification. Software visualization is one of the

Cite this article: AL-Badareen, A. B. (2021). An Enhancement of Software Development Process using Reuse and Visualization Techniques. *European International Journal of Science and Technology*, 10(1), 1-8.

most used techniques to help the developers understand the system structure easily allowing for efficient software reuse results.

White box software reuse is performed based on retrieving software components from reuse repositories or legacy systems. In this type of reuse, the components that are retrieved to be reused in new software development were developed normally and the requirements of the new software system were not considered (Al-Badareen, Selamat, Jabar, Din, & Turaev, 2011) . Therefore, a modification of the retrieved components is required (Systematizing pragmatic software reuse, ACM 2012). The ability of the retrieved components to be modified to satisfy the new requirements is essential in software reuse.

Software visualization is mostly needed and it is widely used in software maintenance, reverse engineering, and re-engineering, where large and complex software systems are needed to be understood from the source code (Francese, Risi, & Scanniello, 2015). Software visualization is the process of presenting a software system visually. It is used to understand software system efficiently and effectively. It helps for a better understanding of the system architecture. The software architecture involves the software elements, their properties and the relationships among them (Fittkau, Krause, & Hasselbring, 2015; Sharafi, 2011).

This work investigates using visualization for improving the understandability of the legacy system in context of software reuse. In addition, visualization is used in order to reduce cost and effort of software reuse and allows for efficient reuse results.

2. Literature Review

Instead of building a new system from a scratch, software reuse enables the system developers to use the existing components of the software (Belli, 2013). Software reuse is often addressed at the level of code or low-level design. Therefore, reuse reduces the time of designing, writing and testing new code (Keswani, Joshi, & Jatain, 2014).

Software reuse has been addressed as a fundamental activity utilized in IT and non IT organizations. Kaindl (Kaindl, 2013) presented and compared three approaches for software reuse in contest of business process and requirements. First approach deals with requirements reuse in the context of product lines, the second approach is for software reuse involving case-based reasoning, and the third approach strives for automating software development through reusing business knowledge.

After building a new system based on a reusable software system it is important to evaluate the performance of the reusable software component. Mahmood et al. (Mahmood, Ahmed, & Alshayeb, 2013) proposed an attribute-based framework for enabling the reuse to take place in the appropriate environment. However, the framework suffers from measuring PC performance requirements based on the new generated software.

Kalotra and Kaur (Kalotra & Kaur, 2014) have conducted a comparative study on performance analysis of four reusable software components; hibernate, spring, baits and Eclipse link. Authors of this paper have also presented a way to build a comparative analysis, in which the average execution time, average heap usage, and average CPU utilization while software reuse are addressed. Mikkonen et al. (Mikkonen, Salminen, & Taivalsaari, 2014) demonstrated the feasibility of the Web as a platform for applications that are constructed using global, shared online components and resources. They have also address the present state of the practice regarding on-demand software.

Achieving a systematic software reuse plays an important role for building successfully new generated software. Several methods have been therefore proposed for achieving a systematic software reuse. Wang et al. (Wang, Ruan, Wang, Li, & Yang, 2013) discussed a method called "Model 1". They have adapted software developed by "Model 1" to the same software but using SSH2 (Struts 2-Spring – Hibernate). The adaption has shown many disadvantages for "model 1".In addition, the adaption is required several rules to fits the SSH2 framework. The rules could be appropriate for e-government at systems, but they are not suitable for other systems.

Recently, many researchers have discussed several approaches extending software development models using set of generic visualization recommendations (Schots, 2014; Vilela, Figueiredo, Castro, Soares, & Goncalves, 2015). In which, the aim was to allow context-aware application developers to use relationship between context and visualization techniques, and to increase awareness in software development scenarios. In the same concern, Vasconcelos et al. (Vasconcelos, Schots, & Werner, 2013) have introduced visualization techniques in terms of a software development. In this paper, a context-oriented visualization recommendation was established to improve awareness in software development. An extended software development context model was also described and used as a basis for developing a customizable context-aware system in context of software development activities.

An alternative context awareness model for software development using software visualization techniques has been presented in (Vasconcelos et al., 2013). In addition, Oliveira (de Oliveira, 2011) has described additional visualization recommendations. This includes examples of hypothetical context situations. A Context-Oriented Visualizations model was implemented in a simple way. However, it suffers from intuitive representation form of software components.

The idea of the code visualization has been introduced in many researches. Balogh and Beszedes (Balogh & Beszedes, 2013) have implemented a conversion tool. It processes the basic source code metrics as input and generates a Mine craft world with buildings, districts, and gardens. In contrast, Lopez-Herrejon and Egyed (Lopez-Herrejon & Egyed, 2013) have described three applications to raise the awareness of the visualization in the area of the software product lines and to spark the interest of the software visualization community. However, both research papers have not focused into the software reuse.

Software analysis plays an important role in building the software and in determining its main characteristics. Khan et al. (Khan, Barthel, Ebert, & Liggesmeyer, 2015) have presented a novel methodology integrating software analysis and software visualization tool via an interactive visual workflow approach. Their standard software analysis tools have been limited in supporting only the well-known metrics. Scarle and Walkinshaw (Scarle & Walkinshaw, 2015) have therefore presented the physics of software exploration system, where it was variably mapped to parameters of a physical model and visualized software via practical system. The resulting visualization was a dynamic scene, the relative positions of entities are not determined by a fixed layout algorithms but by intuitive physical notions. Using practical system has motivated Jovanović et al. (Jovanović, Starčević, & Jovanović, 2014) to propose a data visualization software architecture for unmanned aerial vehicles which as a practical system.

3. The Proposed method

As shows in Figure 1, software reuse consists of four main phases: analyze the legacy system and extract software components, store the extracted components, retrieve the stored components, and reuse the retrieved components in the development of new software system.

The analyze phase intends to read the source code, identify the components of the system (classes and method) and their relationships, and visualized the extracted components. The store phase allows developers to identify the required components, write descriptions about the selected components, and store them in the reuse library. The retrieve phase allows the developers to search in the library about software components and retrieve the required components with their details. The last phase described as reuse which intends to reuse the extracted components for the development of new software system.

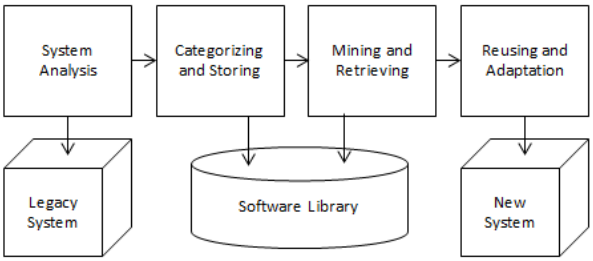

Fig. 1 Software Development Process using Reuse Techniques

The legacy system is a software system was developed in the past either in house or out sourcing. The components of the legacy system can be modified, adapted and used according to other systems' requirements. The main problem faced by developers while dealing with legacy systems is the understandability of its components. In which, developers have to understand the system architecture and the details of its components, in order to identify the intended new components and the required modifications.

The new system is a system currently under development. Developers normally develop the components of this system from scratch. Software reuse helps the developers to use existing components that have been developed in the past in order to save the cost of the development. The components are extracted from the legacy systems and reuse library.

The reuse library is a special database system used to store the source code and the descriptions of the software components. This library helps developers to classify, store and retrieve software components efficiently. Table 1 proposed by (Al-Badareen, Selamat, Jabar, Din, & Turaev, 2010) shows the general information required in Reusable Software Components Library. The library should include all information related to the stored components. The information includes the component description, the source code, the relationships between the component and other components, the data types required in the component, the values of the component's quality characteristics and the responsible information. This simplifies the process of searching, understanding, modifying and reusing the stored components.

1.1 The Analysis Process

System analysis is the process of analyzing the software system in order to identify its components and the relationships among them. The aim of this process is to help developers understand the software system, which is performed on the source code. Figure 2 shows the process of system analysis.

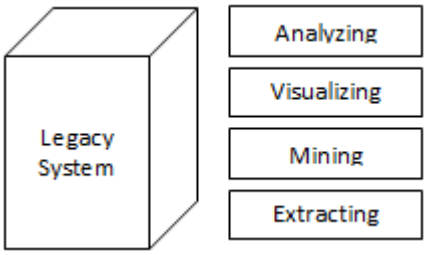

Fig 2: The Process of System Analysis

System analysis is the process of reading the source code and identifying the main components of the system (classes), their contents and their relationships. The visual representation aims to help developers to understand the system's architecture, the main functions and the relationships among the system's components. At this level, the developers will be able to identify the classes that could be reused. Extracting the classes and define intended classes and any other classes that are required along with their functions.

After identifying classes, a detailed description on the contents of the classes and their relationships is required. This allows developers to modify classes in the new system properly. Class analysis identifies methods, variables, relationships among methods, and relationships with other classes. The visual representation of the class helps developers to understand its architecture, and how it could be modified and reused. As a result of this process, developers will be able to make any modifications that might be required for adaptation, and will be able to use the extracted components properly.

1.2 The Store Process

Storing the extracted components in the reuse library required some information about the asset/components that are intended to be stored, in order to classify the component and adding information that describe its functionality and characteristics. This helps the developers searching, identifying, modifying and adapting the intended components into the new system. As presented in Figure 3, the process starts with analyzing the extracted component that intended to be stored, visualizing the component in order to identify its characteristics, classifying the component based on its extracted characteristics, and finally, storing the component and its details in the library.

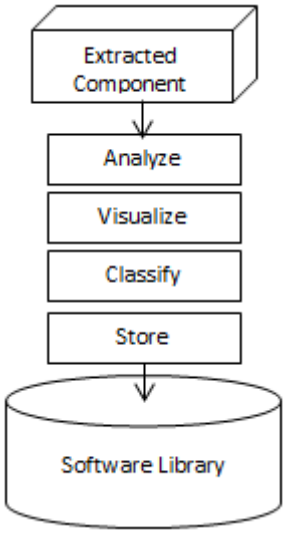

Fig. 3: The Process of Storing the Extracted Component

1.3 The Retrieve Process

Software retrieve process is made during the development of new software system in order to extract the components that could be used in the development of new software system. As presented in Figure 4, the process starts with searching for the available components that are stored in the library and that could be used in the development of the new software system. The retrieved component is analyzed and visualized in order to measure the degree of its suitability to the requirements of the new software system, and in order to identify any modifications that are required for adaptation.

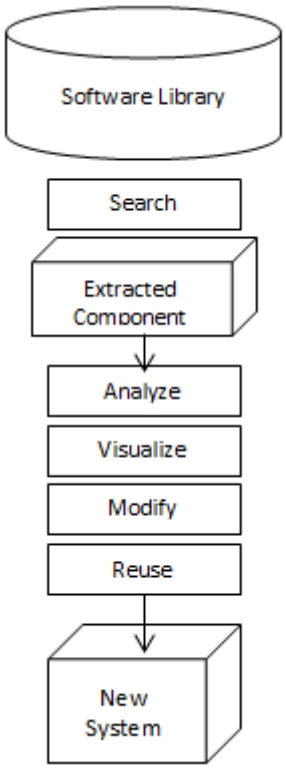

Fig. 4 The process of retrieving software component

4. Conclusion

This work has considered on the enhancement of software reuse process. Developers continuously face problems in analyzing and extracting reusable software components from legacy systems. It is difficult to understand large size of software systems in short time and extract some useful parts, also it is difficult to retrieve and understand the reusable components already stored in the library. Therefore, a new method for software reuse integrated with software visualization was proposed in this work.

For future work, the proposed method is required to be evaluated by a team of experts in software development in order to measure its suitability and consistency with the development process. Moreover, the applicability of the proposed method is required by applying it on a case study.

5. References

[1] Al-Badareen, A. B., Selamat, M. H., Jabar, M. A., Din, J., & Turaev, S. (2010). *Reusable software components framework.* Paper presented at the European Conference of Computer Science (ECCS'10).

[2] Al-Badareen, A. B., Selamat, M. H., Jabar, M. A., Din, J., & Turaev, S. (2011). An Evaluation Model for Software Reuse Processes. In *Software Engineering and Computer Systems* (pp. 586-599): Springer.

[3] Balogh, G., & Beszedes, A. (2013). *CodeMetropolis-code visualisation in MineCraft.* Paper presented at the Source Code Analysis and Manipulation (SCAM), 2013 IEEE 13th International Working Conference on.

[4] Belli, F. (2013). *Dependability and Software Reuse--Coupling Them by an Industrial Standard.* Paper presented at the Software Security and Reliability-Companion (SERE-C), 2013 IEEE 7th International Conference on.

[5] de Oliveira, M. S. (2011). *PREViA: Uma Abordagem para a Visualização da Evolução de Modelos de Software*. Universidade Federal do Rio de Janeiro,

[6] Fittkau, F., Krause, A., & Hasselbring, W. (2015). *Exploring software cities in virtual reality.* Paper presented at the Software Visualization (VISSOFT), 2015 IEEE 3rd Working Conference on.

[7] Francese, R., Risi, M., & Scanniello, G. (2015). *Enhancing Software Visualization with Information Retrieval.* Paper presented at the Information Visualisation (iV), 2015 19th International Conference on.

[8] Jovanović, M., Starčević, D., & Jovanović, Z. (2014). Reusable Design of Data Visualization Software Architecture for Unmanned Aerial Vehicles. *Journal of Aerospace Information Systems*, *11*(6), 359-371.

[9] Kaindl, H. (2013). *Software Reuse Based on Business Processes and Requirements.* Paper presented at the Software Engineering Conference (APSEC), 2013 20th Asia-Pacific.

[10] Kalotra, M., & Kaur, K. (2014). *Performance analysis of reusable software systems.* Paper presented at the Confluence The Next Generation Information Technology Summit (Confluence), 2014 5th International Conference-.

[11] Keswani, R., Joshi, S., & Jatain, A. (2014). *Software reuse in practice.* Paper presented at the Advanced Computing & Communication Technologies (ACCT), 2014 Fourth International Conference on.

[12] Khan, T., Barthel, H., Ebert, A., & Liggesmeyer, P. (2015). *Visual analytics of software structure and metrics.* Paper presented at the Software Visualization (VISSOFT), 2015 IEEE 3rd Working Conference on.

[13] Lopez-Herrejon, R. E., & Egyed, A. (2013). *Towards interactive visualization support for pairwise testing software product lines.* Paper presented at the Software Visualization (VISSOFT), 2013 First IEEE Working Conference on.

[14] Mahmood, S., Ahmed, M., & Alshayeb, M. (2013). *Reuse environments for software artifacts: Analysis framework.* Paper presented at the Computer and Information Science (ICIS), 2013 IEEE/ACIS 12th International Conference on.

[15] Mikkonen, T., Salminen, A., & Taivalsaari, A. (2014). *Enabling Global, Dynamic Web-Based Software Reuse--Mashware Revisited.* Paper presented at the Software Engineering and Advanced Applications (SEAA), 2014 40th EUROMICRO Conference on.

[16] Scarle, S., & Walkinshaw, N. (2015). *Visualising software as a particle system.* Paper presented at the Software Visualization (VISSOFT), 2015 IEEE 3rd Working Conference on.

[17] Schots, M. (2014). *On the use of visualization for supporting software reuse.* Paper presented at the Companion Proceedings of the 36th International Conference on Software Engineering.

[18] Sharafi, Z. (2011). *A systematic analysis of software architecture visualization techniques.* Paper presented at the Program Comprehension (ICPC), 2011 IEEE 19th International Conference on.

[19] Vasconcelos, R. R., Schots, M., & Werner, C. (2013). *Recommendations for Context-Aware Visualizations in Software Development.* Paper presented at the 10th Workshop on Modern Software Maintenance.

[20] Vilela, J., Figueiredo, B., Castro, J., Soares, M., & Goncalves, E. (2015). *Usability and Software Architecture: A Literature Review.* Paper presented at the Components, Architectures and Reuse Software (SBCARS), 2015 IX Brazilian Symposium on.

[21] Wang, X., Ruan, H., Wang, Y., Li, H., & Yang, H. (2013). *A Software-Reuse Method from Model1 to SSH2.* Paper presented at the Computer Software and Applications Conference Workshops (COMPSACW), 2013 IEEE 37th Annual.

About the Author

Anas holds a PhD in Software Engineering (2013) from University Putra Malaysia, MSc in Information Technology (2008) from University Utara Malaysia and BSc in Software Engineering (2006) from Philadelphia University, Jordan. He is an Assistant Professor at the Faculty of Information Technology, director of admission and registration department at Aqaba University of Technology, Jordan. He has 10 years of industry experience in software engineering and software development as well as 8 years of research and teaching in universities. His research interests include internet of things, software quality, project management, formal methods, software maintenance and software reuse.