International Journal of Computer and Information Technology (ISSN: 2279 – 0764)
Volume 10 – Issue 1, January 2021

Reuse Alternatives based on the Sources of Software Assets

Anas Bassam AL-Badareen
Department of Software Engineering
Aqaba University of Technology
Aqaba, Jordan
Email: abdareen [AT] aut.edu.jo

Abstract— **Since the idea of software reuse appeared in 1968, software reuse has become a software engineering discipline. Software reuse is one of the main techniques used to enhance the productivity of software development, which it helps reducing the time, effort, and cost of developing software systems, and enhances the quality of software products. However, software reuse requires understanding, modifying, adapting and testing processes in order to be performed correctly and efficiently. This study aims to analyze and discuss the process of software reuse, identify its elements, sources and usages. The alternatives of acquiring and using software assets either normal or reusable assets are discussed. As a result of this study, four main methods are proposed in order to use the concept of reuse in the software development process. These methods are proposed based on the source of software assets regardless the types of software assets and their usages.**

Keywords-Software Reusability; Software Library Build for Reuse; Reuse Alternatives

I. INTRODUCTION

As software system rapidly growth, developing software system from scratch become more costly

As software systems grow, it becomes more and more costly to develop them from scratch for every new system.

New horizons are opened, since the idea of software reuse appeared in 1968 [1], which was introduced in the NATO Software Engineering Conference in 1968[2]. Software reuse is an important and relatively new approach to software engineering [3]. It represents the ability to use part or a whole of software system in the development of new system [4-7]. This process is related to the functionality of the software system [8]. Software reuse enhance the productivity, maintainability, portability and therefore the overall quality of the end product [9] [10]. This technique reduces the effort, cost, and time of developing new software system. The USA department of defense saves 300$ million yearly by increasing the level of reuse only 1% [11].

Software reuse is not limited to the source code; it considers all information related to the process of developing software system, which includes the requirements, analysis, design, and test [12] [13]. Although, source code is the most commonly reused [14], algorithm and any document produced during software development life cycle are also considered in the reuse [15].

The reuse concept concerns about designing and developing software assets and using these assets in the development of new software systems in the future. Therefore, the reuse process is divided into two main phases, develop for reuse and develop by reuse [16] [17]. Develop for reuse concerns about developing a software asset that could be used in the future in the development of new software system [18, 19]. In this phase, the ability of adapting the developed asset is considered, in order to work with different systems in different environment. Develop by reuse concerns about adapting existing software asset in new system in order to achieve specific requirements [20] [21]. In this phase, the ability of adapting and using the existing software asset is considered, in addition to the ability to achieve its intended functionality.

AL-Badareen et al. [16, 18] proposed a framework for extracting, storing and retrieving normal and reusable software assets during software development lifecycle. The study presents new alternatives for software reuse, which are not considered in Tomer et al. [22]. Therefore, AL-Badareen et al. [23] proposed new model for evaluating the cost of software reuse taking into account the new alternatives. The proposed model presents the probable alternatives of developing and reusing software components.

The main problem faced the software developers is the lack of clear methods for the use of existing software assets that could be used in the development of software systems. Moreover, a systematic method for handling the different types of software assets exist in different sources is needed. Therefore, this study discusses the types of software assets that could be used in the development of software system, the sources of software assets and the methods of acquiring and using software assets in the development of new software systems.

This study discusses four alternatives that consider the concept of software reuse regardless the types of software assets or their sources. The proposed alternatives are defined based on the sources available for software assets and the probable ways of enhancing the development processes. The remains of the paper is structure as follows: section two discusses the elements of software reuse including the sources of the software assets, the types of software assets and the reuse operations; section three discusses the reuse alternatives and the proposed methods; and section four conclude the work and discusses the future directions.

II. Reuse Elements

A. Sources of Software Assets

According to the definition of software reuse, the reusable asset represents any existing software asset that used in the development of new software system. Existing software assets could be found in the legacy software systems, software libraries and the market.

1. Legacy system represents any software system was developed in the past, its internal content and structure are available. A part of the legacy system and its related documents or the whole of it could be used in the development of new system. However, a modification might be required on the software assets in order to meet the requirements of the new system, and for adaptation a modification might be required on the software asset or on the architecture of the new software system.

2. However, AL-Badareen et al. [16, 18] discussed the consideration of the reusability during the development of new system in order to be reused in the future. In this method, the developer made a decision of developing reusable assets instead of normal asset in order to be categorized in the library and reused in the future. However, extra cost and effort of the development of software system is paid, hopping to reduce the time, cost and effort of developing new software system in the future. Normal assets could be categorized in the library without considering the reusability in order to increase the probability of finding these assets in the future, share assets with others, and save the time of mining and retrieving from legacy software system.

3. Software Library is storage where the developed software assets and its related documents are categorized and classified. That is in order to be mined and retrieved efficiently. However, the documents of the software asset could be the planning, requirements, design, source code, and anything related to its development process.

4. The market, where an acquisition of close source software assets is done. Commercial off-the-shelf (COTS) is a ready-made software component available for sale in the market. This source of software assets saves the time and efforts of developing software assets from scratch, where internal structure and component of the asset are not available. Therefore, a modification of this type of assets is not allowed, and any requirements for adaptation have to be made on the architecture of the new software system only.

B. Types of Software Assets

Software assets can be classified into three main types based on the reusability characteristics, normal assets, reusable assets with internal contents or reusable assets with market content.

1. Normal assets are software assets are developed in the past without any consideration of the reusability characteristics. These assets are developed for the purpose of achieving the requirements of a legacy software system. The assets could be found either in legacy software system or stored in software library. However, an extra cost, time, and efforts are required in order to modify these assets in order to achieve the requirements of the new software system and modification for adaptation might be required.

2. Reusable assets with internal contents are software assets were developed in the past and the reusability characteristics were considered. These assets are open source which are developed in house or shared by other teams and categorized in the software library. This type of assets is considered as the most flexible and useful assets in software reuse, which it could be used with or without any modifications (either as black box or white box) to meet the new requirements or for adaptation.

3. Reusable assets with market content or called Commercial Off-The-Shelf (COTS). This type of assets is acquired from the market in order to save the time, effort and cost of developing new software asset. However, the internal content of the assets are not available and therefore, a modification for adaptation or to meet new requirements are not applicable. Moreover, the quality of the assets is unknown.

C. Reuse Operations

The reuse operation is the elementary activities of using software assets in the development of new software system. These operations could be either transition or transformation based on the type of activities. Transition operations concern about the movement of software assets, where transformation concern concerns about the production of software asset.

Transition Operation

Transition operation is a process of transferring software asset among software products or among software products and software library.

- *Cataloging (C):* is a process of categorizing and storing software assets in the software library. It concerns about storing software assets and their related details in the software library in a way that it could be found and retrieved efficiently.

- *Mining (M):* is a process of searching and identifying software assets in a legacy software system (open source software). This requires analyzing the software system in order to identify its contents and identifying the required software asset.

- *Catalog Acquisition (CA):* is a process of searching, identifying and acquiring software assets in the software library. This task is performed based on the requirements of the new software system, where the asset information that is stored in the library is used in the search.
- *External Acquisition (XA):* is a process of searching, identifying and acquiring software assets in the market. This process is performed in the market in order to buy new closed source software asset (COTS).
- *Copy and Paste (CP):* is a process of copying software asset from legacy system and use it directly in the development of new software system. The main idea of this process is that it's based on the personnel knowledge of the software asset and its source.

Transformation Operation

Transformation operation is a process of producing software asset or modifying existing asset. This process is performed in order to achieve the requirements of the development of new software system.

- *New Development (ND):* is a process of developing new software asset from scratch, in order to perform specific task for the purpose of achieving requirements of new software system. Therefore, the reusability characteristics are not considered in this process.
- *New for Reuse (NR):* is a process of developing new reusable software asset, in order to perform specific task for the purpose of achieving requirements of new software systems. Therefore, the reusability characteristics and the ability of the software asset to work with different software systems and in different systems platforms are considered.
- *Adaptation for Reuse (AR):* is a process of modifying software asset in order to achieve the new requirements of the new software system. Therefore, the ability of the software asset to satisfy the requirements of the new software system is considered. Moreover, new reusable asset is produced by considering the reusability characteristics in addition to the requirements of the new software system.
- *Black Box reuse with Modifying System Architecture (BBMSA):* is a process of using commercial software assets (Commercial Off-The-Shelf, COTS) in the development of new software system, where a modification on the system architecture is made for adaptation. That is the commercial software assets are acquired from the market without their internal contents and a modification is not allowed.
- *Black Box as is (BBAI):* is a process of using software assets in the development of new software system without any modifications. This process is performed when the intended software asset satisfies all of the requirements of the new software system including the adaptation requirements.
- *White Box (WB):* is a process of using software assets in the development of new software system, where a modification for adaptation is performed. This process is performed on the software assets where their internal contents and architectures are available and the modifications of the software assets are allowed.

III. Reuse Alternatives

The alternatives of software reuse are identified based on the source of the software assets. In this section, the probable ways that software assets could be going through are discussed.

A. Software Assets Based on Legacy system

Since, the legacy system represents any software system was developed in the past either by the development team (in house) or acquired from outsource. The main idea of the legacy system that software system is open source, which means all of its content and structure are available for acquisition and modification. The whole of the legacy software system or part of it could be considered as software asset for the reuse. Software assets that are acquired from legacy system are normal assets and any consideration for the reuse purposes is not considered.

As proposed in figure 1, software assets are acquired from the legacy system for two main purposes, to be reused in the development of new software system and/or categorized in the software library. For the development of new software system, the acquired software asset is tested based on the requirements of the new software system and a modification for adaptation could be made. For categorizing software asset in the software library, software asset could be categorized as it is or tested based on the reusability requirements in order to produce and categorize a reusable software asset instead of normal.

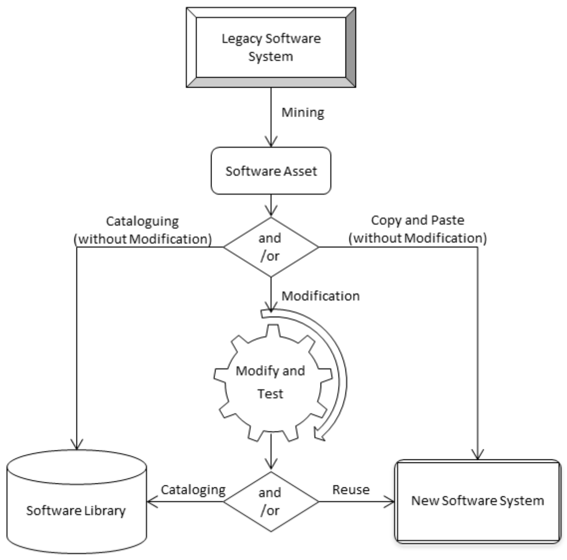

Fig. 1 Software Assets Based on Legacy Software System

B. Software Assets Based on the Market (COTS)

Acquiring software assets from the market has its advantages in avoiding the time, efforts and cost of developing the asset from the scratch. As proposed in figure 2, this type of assets could be used in the development of software system as it is and/or categorizing it in the software library for the future use.

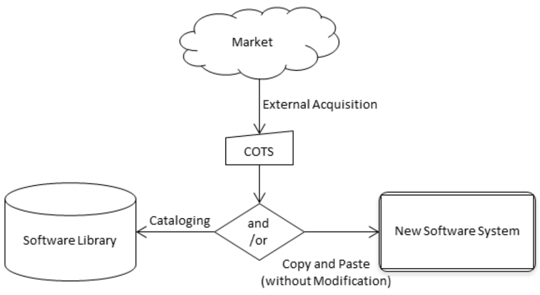

Fig. 2 Software Assets Based on the Market (COTS)

C. Software Assets Based on Software Library

Software Library is defined as any storage used to categorize and store software assets. This storage is defined to reduce the time and efforts of finding and retrieving software assets and their related documents, increase the probability of finding and retrieving software assets were developed in the past, and share software assets with others. However, software assets categorized in the library could be one of three types: open source normal asset, open source reusable asset, closed source reusable asset (COTS).

Open source normal asset is a software asset developed during the development of software system in the past in order to satisfy the system requirements. This asset was categorized in software library directly during the development of software system or retrieved from a legacy system for the reuse purpose and then categorized in the software library. Open source reusable asset is a software asset developed during the development of software system and the reusability characteristics were considered, and then categorized in the software library. The other alternative, a normal asset was retrieved from legacy system modified in order to produce reusable asset. Closed source software asset, or called Commercial of the Shelf (COTS) is software asset acquired from the market in order to achieve certain system requirements and categorized in the software library.

However, as proposed in figure 3, software assets acquired from software library could be used directly in the development of new software system, if the software asset achieves the requirements without modifications, or if the software asset is closed source and modifications are not allowed. Open source normal assets could be modified to achieve the requirements of the new software system and/or modified to produce reusable software asset to be categorized in the library and/or sent to the market as a commercial asset (COTS).

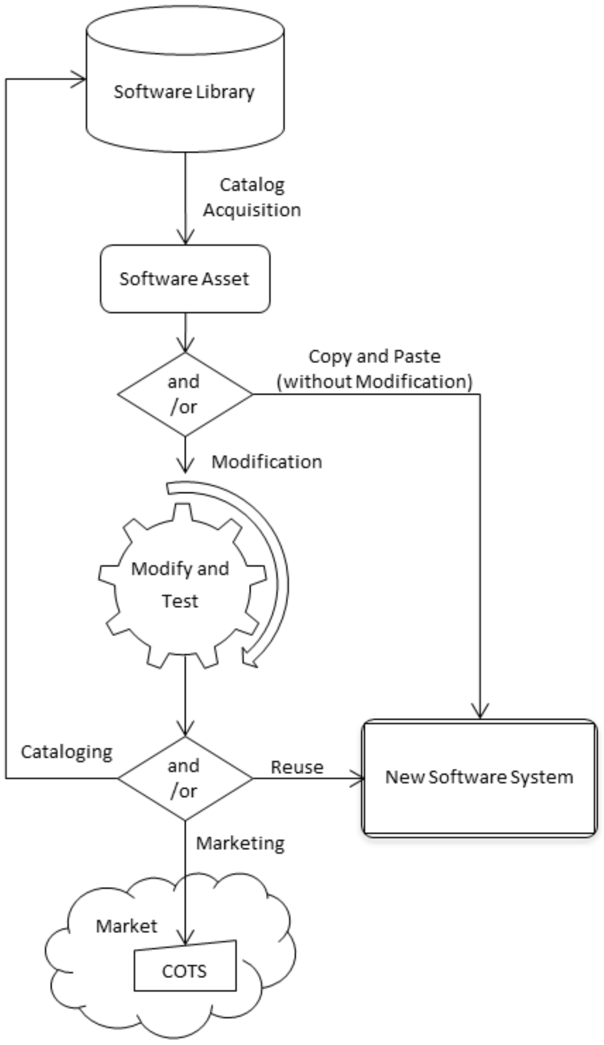

Fig. 3 Software Assets Based on Software Library

D. Software Assets Based on the Development of New Software System

In the development of new software system, the components (assets) of the software system are developed and acquired in different ways from different sources. Although, the reuse strategy is used in the development of new software system, new software assets are developed from scratch. These assets are developed when the search of existing assets that satisfy the new requirements does not return with any results. The main idea of developing new assets is to satisfy the requirements of the new software system, where amendment on the development process could enhance the reusability concept in the organization. As proposed in figure 4, the development of new software asset could be done through different alternatives.

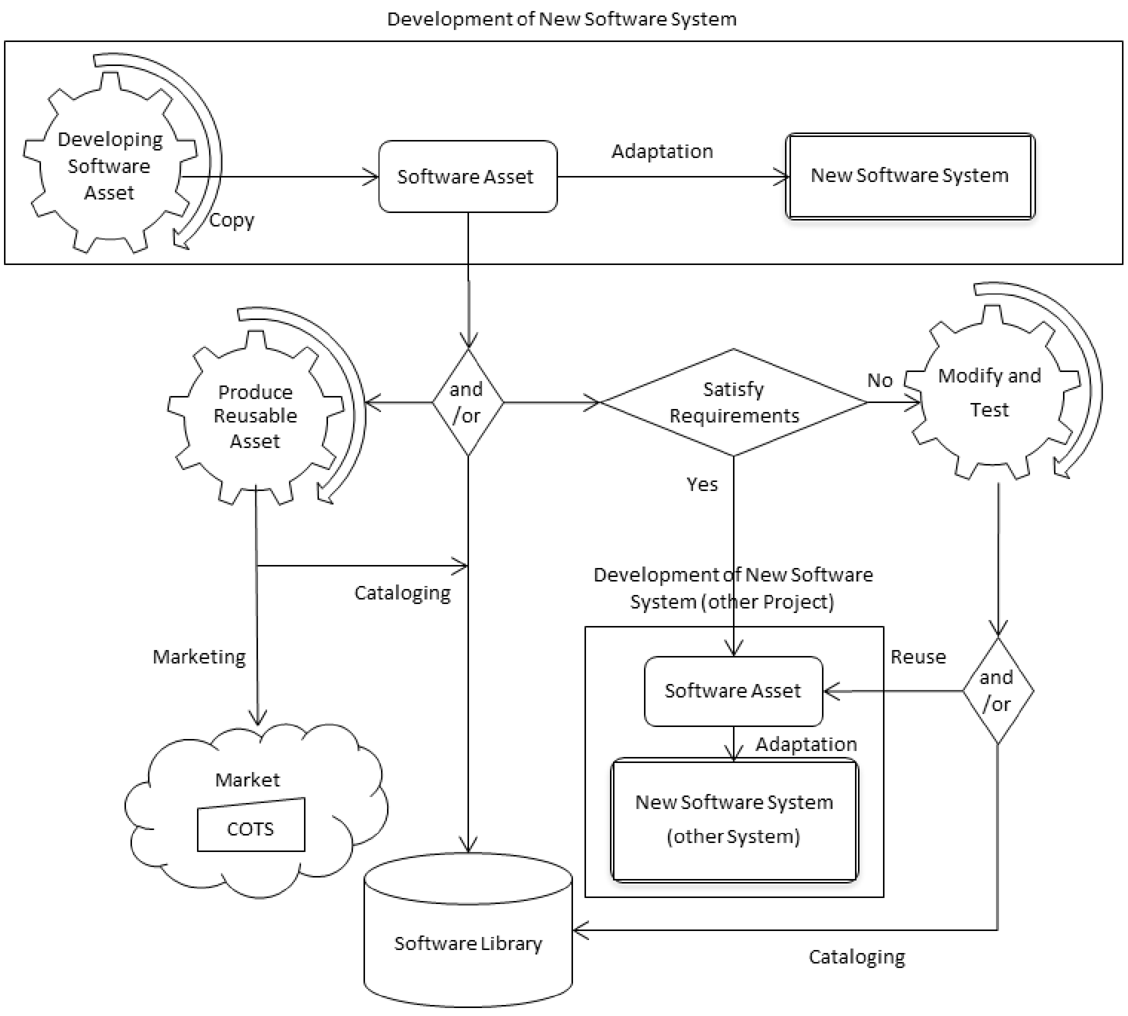

Fig. 4 Software Assets Based on the Development of New Software System

In the development of software system, the purpose of developing software assets is to achieve the requirements of the new software system. Nevertheless, the reusability characteristics could be considered in addition to the main objectives.

However, the produced assets are sent to be evaluated and might be modified according to the new requirements of the new software system, to the requirements of the software library, or to the requirements of the market (COTS).

IV. Conclusion

As software reuse is one of the most efficient techniques for enhancing software development process in terms of time, cost and efforts. This study discussed the alternatives applicable for using the concept of software reuse in the development of software systems. Four main methods were proposed based on the sources of software assets. Moreover, the probable usages of different types of software assets acquired from different sources were discussed.

For the future work, case study would be conducted in order to show the applicability of these methods. Moreover, the cost of following each alternative could be useful for the selection of probable alternatives; therefore, a comparison among the different alternatives discussed in this study could be done in the future.

REFERENCES

[1] P. Gomes and C. Bento, "A case similarity metric for software reuse and design," *Artif. Intell. Eng. Des. Anal. Manuf.*, vol. 15, no. 1, pp. 21-35, 2001, doi: http://dx.doi.org/10.1017/S0890060401151061.

[2] P. Naur, "Software engineering-report on a conference sponsored by the NATO Science Committee Garimisch, Germany," *http://homepages. cs. ncl. ac. uk/brian. randell/NATO/nato1968. PDF*, 1969.

[3] M. Burgin, H. K. Lee, and N. Debnath, "Software technological roles, usability, and reusability," in *Information Reuse and Integration, 2004. IRI 2004. Proceedings of the 2004 IEEE International Conference on*, 8-10 Nov. 2004 2004, pp. 210-214. [Online]. Available: 10.1109/IRI.2004.1431462. [Online]. Available: 10.1109/IRI.2004.1431462

[4] J. A. McCall, P. K. Richards, and G. F. Walters, "Factors in Software Quality," *Griffiths Air Force Base, N.Y. Rome Air Development Center Air Force Systems Command*, 1977.

[5] N. S. Gill, "Reusability issues in component-based development," *SIGSOFT Softw. Eng. Notes*, vol. 28, no. 4, pp. 4-4, 2003, doi: http://doi.acm.org/10.1145/882240.882255.

[6] C. Luer, "Assessing Module Reusability," in *Assessment of Contemporary Modularization Techniques, 2007. ICSE Workshops ACoM '07. First International Workshop on*, 20-26 May 2007 2007, pp. 7-7. [Online]. Available: 10.1109/ACOM.2007.2. [Online]. Available: 10.1109/ACOM.2007.2

[7] F. Haiguang, "Modeling and Analysis for Educational Software Quality Hierarchy Triangle," in *Web-based Learning, 2008. ICWL 2008. Seventh International Conference on*, 20-22 Aug. 2008 2008, pp. 14-18. [Online]. Available: 10.1109/ICWL.2008.19. [Online]. Available: 10.1109/ICWL.2008.19

[8] J. J. E. Gaffney, "Metrics in software quality assurance," presented at the Proceedings of the ACM '81 conference, 1981.

[9] A. Sharma, P. S. Grover, and R. Kumar, "Reusability assessment for software components," *SIGSOFT Softw. Eng. Notes*, vol. 34, no. 2, pp. 1-6, 2009, doi: http://doi.acm.org/10.1145/1507195.1507215.

[10] I. Bitar, M. H. Penedo, and E. D. Stuckle, "Lessons learned in building the TRW software productivity system," 1985.

[11] J. S. Poulin, "Measuring software reusability," in *Software Reuse: Advances in Software Reusability, 1994. Proceedings., Third International Conference on*, 1-4 Nov 1994 1994, pp. 126-138. [Online]. Available: 10.1109/ICSR.1994.365803. [Online]. Available: 10.1109/ICSR.1994.365803

[12] R. Prieto-Diaz, "Status report: software reusability," *Software, IEEE*, vol. 10, no. 3, pp. 61-66, 1993. [Online]. Available: 10.1109/52.210605.

[13] C. V. Ramamoorthy, V. Garg, and A. Prakash, "Support for reusability in Genesis," *Software Engineering, IEEE Transactions on*, vol. 14, no. 8, pp. 1145-1154, 1988. [Online]. Available: 10.1109/32.7625.

[14] B. Jalender, A. Govardhan, and P. Premchand, "Breaking the boundaries for software component reuse technology," *International Journal of Computer Applications*, vol. 13, no. 6, pp. 37-41, 2011.

[15] J. Sametinger, *Software engineering with reusable components*. Springer Science & Business Media, 1997.

[16] A. B. AL-Badareen, M. H. Selamat, M. A. Jabar, J. Din, and S. Turaev, "Reusable Software Components Framework," presented at the European Conference of COMPUTER SCIENCE (ECCS'11), Puerto De La Cruz, Tenerife, December 10-12, 2011, 2010.

[17] J. sharma Ms, A. Kumar, and M. Kavita, "A Design Based New Reusable Software Process Model for Component Based Development Environment," *Procedia Computer Science*, vol. 85, pp. 922-928, 2016, doi: http://dx.doi.org/10.1016/j.procs.2016.05.283.

[18] A. B. AL-Badareen, M. H. Selamat, M. A. Jabar, J. Din, and S. Turaev, "Reusable Software Component Life Cycle," *International Journal of Computers*, vol. 5, no. 2, pp. 191-199, 2011.

[19] H. Nakano, Z. Mao, K. Periyasamy, and W. Zhe, "An Empirical Study on Software Reuse," presented at the International Conference on Computer Science and Software Engineering., 12-14 Dec. 2008, 2008.

[20] Z. Zhu, "Study and Application of Patterns in Software Reuse," presented at the International Conference on Control, Automation and Systems Engineering., 11-12 July 2009, 2009.

[21] M. Kessel and C. Atkinson, "Ranking software components for pragmatic reuse," in *Proceedings of the Sixth International Workshop on Emerging Trends in Software Metrics*, 2015: IEEE Press, pp. 63-66.

[22] A. Tomer, L. Goldin, T. Kuflik, E. Kimchi, and S. R. Schach, "Evaluating software reuse alternatives: a model and its application to an industrial case study," *Software Engineering, IEEE Transactions on*, vol. 30, no. 9, pp. 601-612, 2004.

[23] A. B. AL-Badareen, M. H. Selamat, M. A. Jabar, J. Din, and S. Turaev, "An Evaluation Model for Software Reuse Processes," in *Software Engineering and Computer Systems: Second International Conference, ICSECS 2011, Kuantan, Pahang, Malaysia, June 27-29, 2011, Proceedings, Part III*, J. M. Zain, W. M. b. Wan Mohd, and E. El-Qawasmeh Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 586-599.