Journal of Theoretical and Applied Information Technology
15th January 2021. Vol.99. No 1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

IN THE PROCESS OF SOFTWARE DEVELOPMENT: AVAILABLE RESOURCES AND APPLICABLE SCENARIOS

ANAS BASSAM AL-BADAREEN[1], ASHRAF MOUSA SALEH[2], HAYFA. Y. ABUADDOUS[3], ODAI ENAIZAN[4]

[1] Department of Software Engineering, Faculty of Information Technology, Aqaba University Of Technology, Aqaba, Jordan

[2]Department of Software Engineering, Faculty of Computer Science and Informatics, Amman Arab University, Jordan

[3] Department of Marketing, Faculty of Business, Jadara University, Irbid, Jordan

Email: [1]Anas_badareen@hotmail.com, [2]athamneh_2002@yahoo.com, [2]Haddose@aau.edu.jo, [3]Adi_momane@yahoo.com

ABSTRACT

For many decades, the cost, time and quality are the main concern of software engineering. The main objective of any software organization is to produce high quality software product within a shorter time and minimum cost. Software reuse is one of the main strategies concerns about using available resources to enhance the productivity of software development and the quality of software products. It aims at using existing software products and components in the development of new software systems. However, various types of software components available in different sources are used in the reuse strategy. This makes the reuse strategy confusable and its efficiency and effectiveness debatable. Selecting unsuitable component or scenario makes the reuse inefficient and ineffective. This study discusses the types of software components, their sources, characteristics and applicable scenarios for developing and reusing these components. A dataset from the literature is used to calculate and compare the cost of reuse processes. The results show that software reuse is an efficient strategy comparing with the normal development. Although, considering the reusability of software components required extra cost to the normal development, it could efficiently save the cost of the development of new software system. Moreover, using existing software components in the development of new reusable component is the most efficient strategy, which required even less than the cost of developing normal component.

Keywords: *Software Development, Software Reuse, Reuse Strategy, Software Component, CBSD, develop for Reuse, Develop by Reuse.*

1. INTRODUCTION

Software reuse is not a new concept [1]-[3], it has been recognized as a solution to enhance the efficiency of software development since several decades [4]. Yet, software reuse still recognized as an emerging discipline [5]. In most software companies, the development of large scale software systems is a complex, expensive, slow and unpredictable process [6]. Modern software systems become more large scale, complex and difficult to be managed, which increase the cost of software development, reduce the productivity, uncontrolled quality and the risky in the movement to new technology [7]-[9]. Software reuse is one of the main strategies used to solve these problems [10]. Instead of developing large and complex software applications from scratch, existing software components are used [11]-[13]. These components reduce the cost and time of software development, and to enhance the general quality of software products [14].

Nowadays, the main concern of the research is to reduce the cost and time of software development [15]. Therefore, the reuse strategy aims at using existing software system or part of it in the development of new software system [16], rather than developing it from scratch [5], [17], [18], which saves approximately the half of software development cost [19]. This strategy provides several economic benefits, includes reducing the development time and operating costs, enhancing the time to market and quality of software system,

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

utilizing the development knowledge and corporate expertise efficiently [20], [21].

Software reuse is not limited to specific type of or component of software system, it could be used in different levels of software development including the planning, requirements, analysis, design, testing and maintenance [4], [22], [23],[24]. Where, the reuse of source code is the most common [5], the idea, algorithm and any document produced during software development life cycle also reused [25].

The reuse strategy is used in two main contexts: developing similar software system and in releasing new version of the same software system [4]. Developing similar software system is known as Software Product Lines (SPL) or Domain Engineering [26], [27], [28], and releasing new version of the same software system is known as Software Maintenance (SM) [29],[30], [31].

Software reuse is a process of designing and developing software components and using them in the development of new software products in future. The formal reuse process consists of two main phases: develop-for-reuse and develop-by-reuse [7], [32]. Develop-for-reuse is the process of producing software components that could be used in the development of software system in the future [33], [34], [35]. Develop-by-reuse is the process of using existing software component in the development of software system [36], which includes three main activities: selection, adaptation and integration [37].

However, software reuse is not a simple process of producing software system based on existing software components. It is a decision based method with different alternatives. The reuse decision is made based on the type of software component, the resource of software component, the requirement of the new software system and mainly the cost of the alternative.

The paper is organized as follow: related works presented in section two, section three presents the elements of software reuse, section four presents the scenarios of software reuse, section five presents the costs of the reuse scenarios, the discussions of the results presented in section six and section seven conclude the work.

2. RELATED WORKS

Ravichandran & Rothenberger [38] identified three main reuse strategies: Black-Box Reuse (CBD) with component markets, Black-Box Reuse (CBD) with internal components, and White-Box Reuse with internal components. They provided an extensive comparison among these strategies, and developed decision tree from component reuse.

Ramachandran [39] analyzed the reuse guidelines and classified them into two categories: language-oriented and domain-oriented guidelines. Based on these guidelines, he proposed a tool support to provide advice and generate reusable components automatically.

Buccella et al. [40] proposed an approach for reusing geographical software systems. The proposed approach was based on the analysis of the main characteristics of the geographic domain the Software Product Line development standards.

Byun et al. [41] proposed dynamic reusability metrics (DRM), and apply code visualization to identify reusability maturity level of method/class/class-object with inheritance/associations on a software system.

[2] conducted a comparison of software reuse between software product line (SPL) and conventional software engineering (CSE) communities. They identified the differences and similarities of issues and concerns in software reuse in these two communities, and what each community can provide to overcome the issues identified in the other community. The results show that software reuse has not been addressed sufficiently and there is a lack of awareness of software reuse. Moreover, systematic software reuse process is an important for gaining more technical and economic benefits than other types of software reuse.

Shatnawi et al. [12] proposed an approach to identify reusable software components in object-oriented APIs, based on the interactions between client applications and the targeted API. The study deals with actual clients using the API, dynamic analysis allows to better capture the instances of API usage

Beibei et al. [42] discussed the relationships between object oriented software development and software reuse. The study shows that object oriented method produce general, maintainable and flexible software system, which are very important for software reuse. This method provides also basic technical support for reuse process.

Lucredio et al. [43] studied the impact of Model-Driven Engineering (MDE) on software reuse. The study was conducted through three exploratory studies and compared the software developed with and without MDE. The results show that although some costs associated with MDE, this approach enhances and/or improved software reuse in different situations.

Heinemann et al. [16] conducted an empirical study on software reuse using 20 open source java projects. It aimed to investigate whether the open

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

source software reuse third party code and black-box and white-box reuse occurred. The results of the study show that software reuse is commonly used in open source java software and black-box reuse is mainly used.

Kessel and Atkinson [37] discussed the main issues related to the improvement support for pragmatic reuse selection using test-driven search engines. Moreover, [9] proposed a set of metrics in order to address these issues and new approach for ranking components in search results.

Keswani et al. [44] discussed the major research contributions in software reuse and proposed an adoption model in software reuse. They concluded for appropriate results, a planned and systematic reuse is needed, although software reuse is a difficult task especially for legacy systems, but it provides significant improvement for quality and productivity.

Badampudi et al. [45] conducted a systematic review in order to identify the factors that might influence the decision of choosing component-based among different origins and solutions. The results of analyzing 24 primary studies show that the comparisons among the component origins mainly focused on in-house vs. COTS and COTS vs. OSS. Based on the analysis, 11 factors affect the decision of selecting a component origin are identified, where the cost, time and reliability are the main factors considered in the selection for in-house vs. COTS.

Sharma et al. [7] analyzed four common CBSD models and discussed their characteristics. They state that the process of developing component based software system consists of two main processes: component development and system development. In both processes, optimal selection of component based plays an important role, which is not considered in the presented development process. Therefore, a model for the optimal selection of software component for the development of component based software systems was proposed.

Nautiyal et al. [15] state that CBSE approach is the idea of selecting appropriate components and assembling them with well-defined software architecture. Component selection and integration play important role in component based software development. Since the CBSE has rapidly grown, there is a great need for a lifecycle model for component based software development [46]. Therefore, they proposed a life cycle model for component based software development called Elite Life Cycle Model (ELCM).

Khan et al. [47] state that time to market, cost and product's quality of software development are the main factors affect the software industry. For reducing the cost and time to market, and enhance the quality of product, several approaches, techniques and tools are proposed. CBSD techniques are example of these techniques, which aim to build software systems using existing reusable software components. They aimed at describing the characteristics of CBSD models presented in the literature and widely practiced in software industries. Based on the literature analysis, new model for component based software development was proposed. The proposed model covers both component based software development and component development phases. The strengths and weaknesses of the proposed model are compared with the current models.

Basha and Moiz [48] presented a state of art of radical change in component technology from component engineering to domain engineering. They concluded that the most promising branch of CBSD is the CBD-Arch-DE. That is because it has some desirable properties like design as orientation, extensibility and reusability; in addition, its focal models are good at revealing system essential. Where, modeling a software system in a way that can be reused is one of the software design goals. Component based software development is a key role in increasing the productivity of any organization [49]. Yet, the development of domain specific components and its impact on cost and time is still a challenging issue.

Tomer et al. [50] presented for applicable scenarios for software reuse, and proposed a model for evaluating these scenarios in terms of cost. Seven industrial assets were used in evaluating these scenarios and comparing them with the cost of the normal development.

AL-Badareen et al. [32], [33] proposed a framework for extracting, storing and retrieving normal and reusable components during software development lifecycle. The study presents new scenarios for software reuse, which are not considered in Tomer et al. [50]. Therefore, AL-Badareen et al. [19] proposed new model for evaluating the cost of software reuse taking into account the new scenarios. The proposed model presents the probable scenario of developing and reusing software components. According to Shirali-Shahreza and Shirali-Shahreza [51], software reuse reduces the time and cost of software development, whereas finding suitable software components is one of its main issues. They investigate the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

problems of retrieving software components from libraries and discuss their solutions.

3. REUSE ELEMENTS

3.1 Sources of Software Components

Since, software reuse is not limited to specific type of software component, they exist in different sources. These components could be acquired from five sources: new for reuse, legacy system, product during development, repository and external sources.

1. **New for Reuse:** is in-house development of new reusable component. This component is developed from scratch considering the reusability characteristics, such as generality, portability and interoperability. The main idea of the reusable component is to perform specific tasks with different data types in different environments and system architectures. This kind of software components could be used directly in the development of new software system, categorized in the library to be reused in the future (Black-Box or White-Box), and sold in the market as Commercial Off-The-Shelf (COTS).
2. **Legacy system (Exist SW Product):** is any Open Source Software (OSS) developed in the past, where its components are available for mining and retrieving. This kind of software systems could be developed in-house or acquired from external sources, which consists of a set of normal components. These components could be reused directly in the development of new software system (either Black-Box or White-Box), adapted to produce new reusable components or categorized in the library. However, this kind of software required to be analyzed and understood in order to be able to identify and extract the required components. Moreover, analyzing and understanding the extracted component is required for its categorization, adaptation and reusing.
3. **Product during Development:** presented in AL-Badareen et al. [32], [33]. The reuse concept is considered early in the planning phase of software development life cycle. At the planning phase, in addition to the requirements of the new software system, the developers made a decision about software components, to be categorized in the library or to develop reusable component instead of normal. Two type of components could be produced during the development of software system:

   a) Normal Component: during the development of software system, the normal components that are developed are categorized in the library. This process reduces the time and effort of mining the components and avoids the probability of missing these components.

   b) Reusable Component: instead of developing normal components to achieve the development objectives, reusable components are developed. At the same time, these components are adapted to the software system, categorized in the library and offered in the market (COTS).
4. **Library:** is a repository used to categorize and store software components or any information related to the software system and could be reused in the future, such as, planning, requirements, design, source code, etc. This information should be stored in a way that could be found and retrieved efficiently. Moreover, the library could enhance the understandability of the retrieved components in order to simplify the process of modifying and reusing them.
5. **External Acquisition:** Commercial off-the-shelf (COTS) is a ready-made software component available for sale in the market. COTS acquisition reduces the time and efforts of developing software components from scratch, while internal structure and content of the component are not available. Therefore, the quality of the component is unknown and any modification for adaptation is made on the system architecture instead of modifying the component.

3.2 Components Types

In terms of reuse, software components can be classified into three types: normal component, reusable component with internal content and reusable component with market component (COTS).

1. Normal Component (NC): is a software component developed to perform particular task in specific software system. This component achieves the requirements of its intended software system, while the reusability characteristics are not considered. The normal components are developed from scratch, retrieved from legacy system or from system during development in order to be reused directly or categorized in the library. The

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

modification of the normal components can be done to fit the requirements of the new software system or to produce reusable components by achieving certain level of reusability.

2. Reusable Component with Internal Content-Open Source Component (OSC): is a software component satisfies the reusability requirements and available with its internal content, which allows a modification for adaptation. It is in-house software component, developed from scratch or by modifying normal software components. This type of components is available for reusing in new software systems, categorized in the library and also could be adapted to produce reusable component with market content in order to be sold in the market.
3. Reusable Components with Market Content-Commercial Off-The-Shelf (COTS): is a software component purchased from the market, which satisfies the reusability requirements. This type of components is closed and its internal structure and content are not available. Therefore, the quality of the component is unknown and any modification for adaptation is made on the system architecture instead of the components itself. Acquiring the market component saves the time and cost of software development and it can be categorized in the library for future reuse.

3.3 Reuse Operations

The reuse operations are elementary activities performs on the software components. These activities are classified into transition and transformation. Transition operations are related to the movement of software components, while transformation operations are related to producing, modifying and reusing the components.

3.3.1 Transition operation

- Cataloging (C): categorizing and storing software components in a library. This process includes storing the software component, and its details and characteristics in a way that can be retrieved efficiently.
- Mining (M): identifying and acquiring software components from certain software system (OSS). This process includes the process of analyzing the software system and identifying the required component.
- Catalog Acquisition (CA): identifying and acquiring software component from the reuse library. It is a process of searching for a certain software component to perform specific task.
- External Acquisition (XA): acquiring software component from external source (COTS component) and categorizing it in the library.
- Copy and Paste (CP): acquiring software component from software system, where the source of the component is known to the developer, which is based on the personnel knowledge of the component and its source.

3.3.2 Transformation operation

- New Development (ND): developing new software component from scratch. This component is developed to perform specific task with specific data type, and within certain system architecture and environment.
- New for Reuse (NR): developing new reusable software component from scratch. It is categorized in the library with its internal structure and content, and it could be sold in the market as COTS.
- Adaptation for Reuse (AR): modifying software component in the library in order to satisfy new requirements. Usually, normal components are modified to satisfy requirements of new software system or to produce new reusable software components.
- Black Box reuse with Modifying System Architecture (BBMSA): using the reusable component as is in the development of new software system, while the system architecture is modified for adaptation. This process is performed for reusing COTS components.
- Black Box as is (BBAI): using software component in the development of new software system as is without any modifications.
- White Box (WB): modifying and reusing software component in the development of new software system. The modification is for adapting the component to the new software system.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4. SOFTWARE REUSE SCENARIOS

The reuse process is divided into two phases: develop-for-reuse and develop-by-reuse. Develop-for-reuse is a process of producing software components that could be used in the development of software systems in the future. Develop-by-reuse is a process of using existing software component in the development of new software system.

4.1 Develop-for-Reuse

Develop-for-ruse concerns about preparing any software component in order to be reused. This includes, developing software component from scratch (normal or reusable), modifying software components categorized in the library, mining and extracting software components from existing software system (OSS), or acquiring software component from external source (COTS). Based on the type and source of software component, there are six applicable scenarios for preparing software components in order to be reused, see figure 1.

Figure 1:. Develop-for-reuse scenarios

Extract from existing software system (OSS): In this scenario, normal software components are extracted from open source software OSS system. This process includes analyzing the software system and extracting the intended components. The extracted components are normal components, which could be used in the development of new software system (Black Box or White Box), categorized in the library or modified to produce new software components.

New for Reuse (NR): is a development of new reusable software component from scratch. It is a process of producing software components that able to perform specific tasks with different data types, and within different environments and systems architectures. The extra cost required in this scenario is to consider the reusability characteristics, such as, generality, interoperability and portability. The reusable components produced in this scenario could be used in the development of new software system, categorized in the library and adapted to be sold in the market as COTS.

External Acquisition (XA): is a process of purchasing and acquiring Commercial Off-The-Shelf components from external sources. This kind of components is able to work in different environments with different systems architectures and data types, where a modification is not allowed.

Adaptation for Reuse (AR): is a process of modifying software component in order to satisfy new requirements. This modification is performed in order to use the component in the development of new software system or to produce new reusable component. Normally, this process is performed on the normal components categorized in the library.

Extract during Development: presented in [32], [33], it aims to extract and categorize software components during the development of software system. During the development of software system, each component developed to perform specific task in the new system is directly categorized in the library. This idea helps to avoid the probability of missing the developed components, and reduce the cost of mining and extracting the components. Extra effort might be added to the development efforts in order to produce reusable components instead of normal.

Extract Normal Component during Development (NC): is the process of copying and categorizing software components once they are developed during the development of software system.

Extract Reusable Component during Development (Open Source Component- OSC): is the process of producing reusable software components instead of normal, and copy and categorize these components in the library during the development of software system. At early stage of software development, the reusability is considered in the development of some components. The decision of developing reusable components instead of normal is based on a high probability of reusing them in the future or the feasibility of offering them in the market (COTS). Although, an extra effort required for considering the reusability, it saves the time and effort of software development in the future, enhance the quality of the components and it could be worthy to be offered in the market for selling as COTS components.

4.2 Develop-by- Reuse (DBR)

Develop-by-reuse or component based software (CBS) is the main idea of software reuse, which aims to use existing software components in the development of new software system. Two applicable scenarios for developing software system using existing software components: White-Box and Black-Box. The selection of the scenario is identified based on the type of software component

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

and the requirements of the new system, see figure 2.

Figure 2: Develop-by-reuse scenarios

Black-Box reuse (BB): using existing software component in the development of new software system as it is without any modifications. This scenario is applicable for all types of components if they fit the requirements of the new system. However, the black-box reuse is compulsory for the COTS components, where modifying these components is not allowed. Therefore, for adaptation, the system architecture is modified instead of the COTS component.

White-Box reuse (WB): is the process of modifying and using existing software component in the development of new software system. In this scenario, the normal and open source reusable components (OSC) are modified to satisfy the requirement of the new software system.

5. COST OF REUSE PROCESS

The decision of selecting the reuse scenario is based on the cost of the applicable scenarios. In each reuse process, the applicable scenarios are identified based on the type of the available components and their sources. In this section, the cost of the reuse processes is discussed based on the dataset published in [50]. The dataset presents the cost of the basic operations for developing seven industrial software components. The operations are: New Development (ND), New for Reuse (NR), Mining and Cataloging (MC), Copy and Paste (CP), Adaptation for Reuse (AR), Catalog Acquisition (CA), Black Box Reuse (BB) and White Box Reuse (WB).

As shows in figure 3, the average cost of developing new reusable component (NR) from scratch required approximately 34% extra cost to the development of normal software component (ND). This extra cost is required for considering the reusability characteristics, such as generality, adaptability and interoperability. However, developing reusable component and reuse it one time (NR+CP+CA+BB) saves around 12% of the cost of developing normal component two times (ND+ND), see figure 4, while the cost of acquiring software component from the library and reusing it (CA+BB) is 23% of the cost of developing normal component (ND), see figure 5.

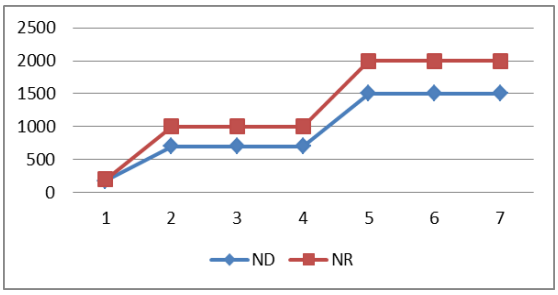

Figure 3: The cost of developing normal component vs. reusable component

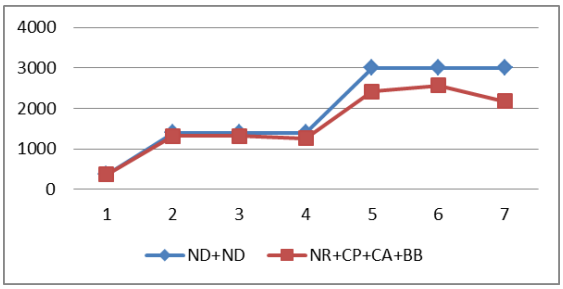

Figure 4: The cost of developing normal component twice vs. developing and reusing reusable component

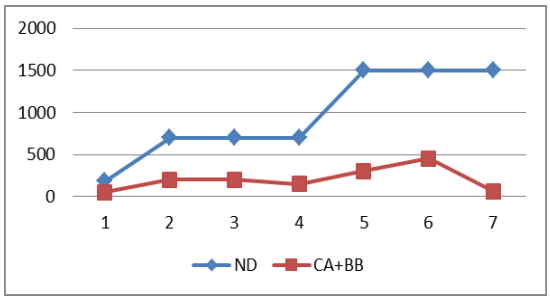

Figure 5: The cost of reusing existing component vs. developing normal component

As shows in figure 6, developing reusable component during the development of new software system and reuse it one time (NR+CP+CA+BB) saves around 7% of the cost of developing normal component, adapting it, and reusing it in the development of new software system (ND+CP+AR+CA+BB).

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

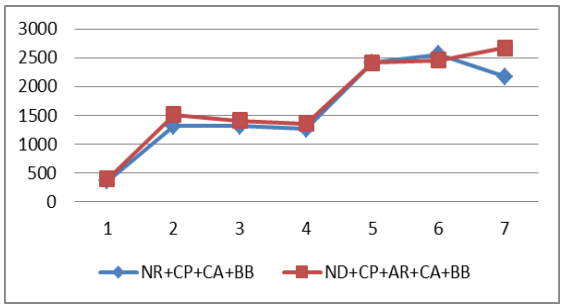

Figure 6: The cost of new for reuse vs. adapting for reuse

Figure 7 shows that retrieving software components from existing software system (OSS) and modifying it to produce reusable components (MC+AR) equals around 50% of the cost of developing reusable software component from scratch (NR) and around 32% of the cost of developing normal component from scratch (ND).

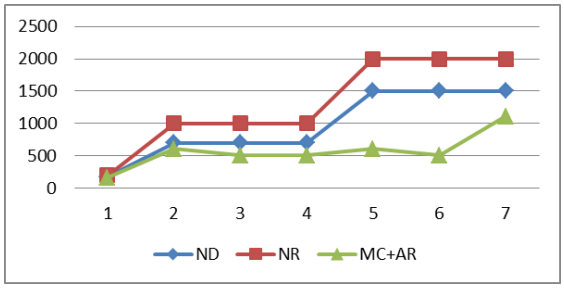

Figure 7: The cost of retrieving and adaptation for reuse vs. normal development vs. develop for reuse

6. DISCUSSION AND IMPLICATION

Software reuse is an efficient strategy for reducing the cost and time of software development and enhancing the quality of software products. The main idea of software reuse is to use any available material related to software product in the development process. However, the available components have different forms and exist in different type of sources. Therefore, the processes of extracting, adapting and reusing these components are different.

Software components could be retrieved from four types of sources: existing software system (OSS), software during development, library and external acquisition.

Existing software system are open source software systems (OSS) developed in the past and available for mining. This kind of software systems are developed in-house or acquired from external sources. The main issue related to this kind of sources is the process of analyzing and understanding the software system in order to be able to identify and extract the components. Especially, in the complex and large size software systems, the understandability is very difficult and time-consuming task. Although, several approaches and tools were proposed to overcome this issue, yet, studies claim that understandability is a challenging issue. Moreover, the understandability and modifiability of the extracted components are key factors for the success adaptation.

The main idea of extracting software components during the development of software systems is reduce the cost of mining and the probability of missing the components in the future. Once the software component is developed, it will be categorized in the library. In this process, the process of analyzing and understanding the software system and software component are avoided, since the component is categorized by its developer. Although, it adds extra efforts to the development process, it saves a lot of time and efforts in the reusing process. Also, this idea overcome the issue of missing and forgetting the components by the developers and helps to share these components with other developers and organizations, which increase the reuse probability.

Reuse library is a warehouse stores and categorizes software components in a way that could be shared, retrieved, understood and reused efficiently. The library design takes into account that components will be mined, retrieved, modified and reused by other developers and organization, and it will be compared with other components for similarity and suitability. Therefore, sufficient, suitable and accurate information stored in the library are essential in the success of any library.

External acquisition is a process of acquiring software components from the market, commercial-off-the-shelf (COTS). The main benefits of the COTS are the time and cost efficiency, while it suffers from the unknown internal quality and unable to modify it for adaptation. Since, the internal content of the components are not available, the quality of the component is unable to be measured and therefore, the total quality of the system is unable to be calculated. Moreover, the developers are unable to modify the component for adaptation, which requires modifying the system architecture instead. This makes the adaptation more complex, and needs more time and effort.

The reusable software components represent any software component that could be used in the development of new software system, which include: normal software component, reusable software component and COTS components. The

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

normal software component is any software component developed for specific purpose in certain software system and available for mining and modifying to be reused. The normal components are available in exist software systems or in a library. Reusable software component is any software component considered the reusability characteristics and able to be modified for adaptation. This component is developed in-house or it could be acquired from external sources. Normally, open source software components are available in a library for mining. The COTS components are normally acquired from external sources and available in a library for mining and reusing as a Black-Box.

Since, software components have different forms and exist in different sources, different processes are required to deal with them. The processes of dealing with the software components are distributed in two categories: transition and transformation. Transition is a process of moving software component from its source without any modification. Transformation is a process of creating and modifying software component for producing other form component in order to satisfy new requirements.

Transition operation processes include: catalog, mining, catalog acquisition, external acquisition, and copy and paste. Catalog is a process of classifying and storing software component in a library. Mining is a process of retrieving software component from existing open source software system (OSS). External acquisition is a process of acquiring software component from external source (COTS). Copy and paste is a process acquiring software components known for developers from existing software system (OSS).

The transformation operation processes include: New Development (ND), New for Reuse (NR), Adaptation for Reuse (AR), Black-Box reuse as is (BBAI), Black-Box reuse with modified system architecture (BBMSA) and White-Box reuse (WB).

New development is a process of producing software component from scratch. New for reuse is a process of producing new software component from scratch considering the reusability characteristics. Adaptation for reuse is a process of modifying software component in order to satisfy new requirements including reusability. Black-Box reuse with modifying system architecture is the process of using reusable component (COTS) with modification of system architecture. Black-Box reuse "as is" is a process of using software component without any modification on the component or the system architecture. White-Box reuse is a process of modifying software component and using it in the development of new software system.

Software reuse is a set of processes applied on specific type of software components exists in specific software system. A combination of these processes is a reuse scenario. In this study, software reuse is considered from two sides, developed-for-reuse and develop-by-reuse. Develop-for-reuse considers the applicable scenarios for developing software component that could be used in the development of new software system in the future. Develop-by-reuse considers the process of using existing software component in the development of new software system.

The cost analysis presented the cost of the main reuse processes comparing with the normal development. Although, the cost of considering the reuse early is higher than the normal scenario of software development, it saves a lot of time and effort in the future. The analysis showed that reusing software component for one-time payback the cost of considering the reusability characteristics efficiently.

Moreover, the analysis showed that all of the reuse scenarios are efficient in terms of cost and time comparing with the normal development. Using the reuse strategy in the development of reusable components is the most efficient scenario in the reuse strategy, which aims at adapting software components to produce reusable components. However, using existing software components in the development process suffers from many issues related to retrieving and modifying software components. Identifying and choosing software components is a complex and difficult task, especially in large scale software systems and distributed libraries. It required analyzing software system and identifying the components that are suitable to the intended requirements, and modifying the component for adaptation. This process required understanding the structure and the content of the software system. For retrieving software component from distributed libraries, more than one component could be retrieved from different sources for specific requirements. This required an accurate comparison among the retrieved components in order to find the most suitable one to work within the new system.

7. CONCLUSION

Software reuse is one of the main strategies of reducing the time and cost of software development and enhancing the quality of software systems. Since, the reuse is not limited to specific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

type of software components or acquired from specific source, which it could be anything that can be used in the development of software system in the future retrieved from any source, different scenarios are applicable. The reuse scenario is identified based on the type and source of software component. In this study, the types of software components that could be reused in the development of software system were identified. Based on this, the processes that are required for acquiring, modifying and reusing these components were discussed and the applicable scenarios are presented. A dataset from the literature was used to calculate and compare the cost of reuse processes comparing with the normal development. The results showed that software reuse is an efficient development strategy comparing with the normal development. Although, considering the reusability of the software components required extra cost, it could save more than this cost in the development of new software system. Moreover, using existing software components in the development of new reusable component is the most efficient strategy, which required even less than the cost of developing normal component.

REFERENCES:

[1] Incorvaia, A. J., Davis, A. M., & Fairley, R. E, "Case Studies In Software Reuse", *InProceedings Fourteenth Annual International Computer Software and Applications Conference*, IEEE Computer Society, Jan 1, 1990, pp. 301-302,

[2] Jha M, O'Brien L, "A Comparison Of Software Reuse In Software Development Communities", *In 2011 Malaysian Conference in Software Engineering*, IEEE, Dec 13, 2011, pp. 313-318.

[3] Trauter R, " Design-Related Reuse Problems-An Experience Report", *In Proceedings. Fifth International Conference on Software Reuse (Cat. No. 98TB100203)*, IEEE, Jun 5, 1998, pp. 176-183.

[4] Chernak Y, "Requirements Reuse: The State Of The Practice", *In 2012 IEEE International Conference on Software Science, Technology and Engineering (SWSTE)*, IEEE, Jun 12, 2012, pp. 46-53.

[5] Jalender B, Govardhan A, Premchand P, "Breaking The Boundaries For Software Component Reuse Technology", *International Journal of Computer Applications*, Vol. 13, No. 6, Jan, 2011, pp. 37-41.

[6] Santos R, Barbosa O, Alves C, "Software Ecosystems: Trends And Impacts On Software Engineering", *In 2012 26th Brazilian Symposium on Software Engineering*, IEEE, Sep 23, 2012, pp. 206-210.

[7] Sharma Ms J, Kumar A, Ms K, "A Design Based New Reusable Software Process Model for Component Based Development Environment", *Procedia Computer Science*, Vol. 85, Jan 1, 2016, pp. 922-8.

[8] Yu L, Chen K, Ramaswamy S, "Multiple-Parameter Coupling Metrics For Layered Component-Based Software", *Software Quality Journal*, Vol. 17, No. 1, Mar 1, 2009, pp. 5-24.

[9] Padhy N, Singh RP, Satapathy SC. "Software reusability metrics estimation: algorithms, models and optimization techniques". *Computers & Electrical Engineering*. 2018 Jul 1;69:653-68.

[10] Boehm, B, "Managing software productivity and reuse", *Computer*, Vol. 32, No. 9, 1999, pp. 111-113.

[11] Frakes WB, Kang K, "Software Reuse Research: Status And Future", *IEEE transactions on Software Engineering*, Vol. 31, No. 7, Aug 8, 2005, pp. 529-36.

[12] Shatnawi A, Seriai AD, Sahraoui H, Alshara Z, "Reverse engineering reusable software components from object-oriented APIs", *Journal of Systems and Software*, Vol. 131, Sep 1, 2017, pp.442-60.

[13] Zibran MF, Eishita FZ, Roy CK, "Useful, But Usable? Factors Affecting The Usability Of APIs", *In 2011 18th Working Conference on Reverse Engineering,* IEEE, Oct 17, 2011, pp. 151-155.

[14] García-León RA, Flórez-Solano E, Rodríguez-Castilla M. "Application of the procedure of the ISO 50001: 2011 standard for energy planning in a company ceramic sector". *DYNA*. 2019 Jun;86(209):113-9.

[15] Nautiyal L, Gupta N, "Elicit-A New Component based Software Development Model", *International Journal of Computer Applications,* Vol. 63, No. 21, Jan 1, 2013,

[16] Heinemann L, Deissenboeck F, Gleirscher M, Hummel B, Irlbeck M, "On the extent and nature of software reuse in open source java projects", *In International Conference on Software Reuse,* Springer (Berlin, Heidelberg), Jun 13, 2011, pp. 207-222.

[17] Jalender B, Govardhan A, Premchand P. A, "Pragmatic Approach To Software Reuse", *Journal of Theoretical & Applied Information Technology*, Vol. 14, Apr 1, 2010.

[18] Rawwash H, Masad F, Enaizan O, Eneizan B, Adaileh M, Saleh A, Almestarihi R, "Factors

Affecting Jordanian Electronic Banking Services", *Management Science Letters*, Vol. 10, No. 4, 2020, pp. 915-22.

[19] Al-Badareen AB, Selamat MH, Jabar MA, Din J, Turaev S, "An Evaluation Model For Software Reuse Processes". *In International Conference on Software Engineering and Computer Systems,* Springer (Berlin, Heidelberg), Jun 27, 2011, pp. 586-599.

[20] Tung YH, Chuang CJ, Shan HL, "A Framework Of Code Reuse In Open Source Software", *In The 16th Asia-Pacific Network Operations and Management Symposium (APNOMS),* IEEE, Sep 17, 2014, pp. 1-6.

[21] Saleh AM, Ismail R, Fabil N, Norwawi NM, Wahid FA, "Measuring Usability: Importance Attributes for Mobile Applications", *Advanced Science Letters*, Vol. 23, No. 5, May 1, 2017, pp.4738-41.

[22] Hewett R, "Learning From Software Reuse Experience", *In 2005 International Symposium on Empirical Software Engineering*, IEEE, Nov 17, 2005, pp. 10-pp.

[23] Saleh A, Ismail R, Fabil N, "Evaluating Usability For Mobile Application: A MAUEM Approach", *In Proceedings of the 2017 International Conference on Software and e-Business*, Dec 28, 2017, pp. 71-77.

[24] Enaizan O, Eneizan B, Almaaitah M, Al-Radaideh AT, Saleh AM, "Effects of Privacy And Security On The Acceptance And Usage Of EMR: The Mediating Role Of Trust On The Basis Of Multiple Perspectives", *Informatics in Medicine Unlocked*, Vol. 21,Oct 13, 2020,100450.

[25] Sametinger J, "Software Engineering With Reusable Components", Springer Science & Business Media, Jun 19, 1997.

[26] Clements, P. and Northrop, L., "Software Product Lines", Addison-Wesley (Boston), 2002, pp. 226-229.

[27] Kang, K. C., Sugumaran, V., & Park, S, "Applied Software Product Line Engineering", CRC press, 2009.

[28] Van der Linden FJ, Schmid K, Rommes E, "Software Product Lines in Action: The Best Industrial Practice In Product Line Engineering", Springer Science & Business Media, Jun 10, 2007.

[29] Grubb P, Takang AA, "Software Maintenance: Concepts And Practice", World Scientific, Jul 7, 2003.

[30] Kang KC, Lee J, Donohoe P, "Feature-Oriented Product Line Engineering*", IEEE software*, Vol. 19, No. 4, Aug 7, 2002, pp. 58-65.

[31] Pigoski, T. M. "Practical Software Maintenance: Best Practices For Managing Your Software Investment", Wiley Publishing, 1996.

[32] Al-Badareen AB, Selamat MH, Jabar MA, Din J, Turaev S, "Reusable Software Components Framework", *InEuropean Conference of Computer Science (ECCS 2011),* Nov 30 ,2010, pp. 126-130.

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

[34] Nakano, H., Mao, Z., Periyasamy, K., & Zhe, W., "An Empirical Study on Software Reuse", In *2008 International Conference on Computer Science and Software Engineering,* IEEE, Vol. 6, December, 2008, pp. 509-512

[35] Enaizan, O. M., Alwi, N. H., & Zaizi, N. J., "Privacy and Security Concern for Electronic Medical Record Acceptance and Use: State of the Art", *Journal of Advanced Science and Engineering Research*, Vol. 7, No. 2, 2017, pp. 23-34.

[36] Zhu, Z., "Study And Application Of Patterns In Software Reuse", In *2009 IITA International Conference on Control, Automation and Systems Engineering (case 2009),* IEEE, July, 2009, pp. 550-553.

[37] Kessel, M., & Atkinson, C., "Ranking Software Components For Pragmatic Reuse", *In 2015 IEEE/ACM 6th International Workshop on Emerging Trends in Software Metrics,* IEEE, 2015, pp. 63-66.

[38] Ravichandran, T. and Rothenberger, M.A., "Software Reuse Strategies and Component Markets", *Communications of the ACM*, Vol.46, No. 8, 2003, pp.109-114.

[39] Ramachandran, M., "Software Reuse Guidelines", *ACM SIGSOFT Software Engineering Notes*, Vol. 30, No. 3, 2005, pp.1-8.

[40] Buccella, A., Cechich, A., Arias, M., Pol'La, M., del Socorro Doldan, M. and Morsan, E., "Towards Systematic Software Reuse Of Gis: Insights From A Case Study", *Computers & Geosciences*, 2013. Vol. 54, 2013, pp.9-20.

[41] Byun EY, Son HS, Jeon B, Kim RY. "Reusability Strategy Based on Dynamic Reusability Object Oriented Metrics". *Journal of Engineering Technology*. 2018 Jan;6(1):365-77.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[42] Beibei, X., Haitao, W. and Fengwang, Z., "Research On Software Reuse Methods Based On The Object-Oriented Components", *In Proceedings of 2012 2nd International Conference on Computer Science and Network Technology(ICCSNT),* IEEE, December, 2012, pp. 1857-1860.

[43] Lucredio, D., de Almeida, E.S. and Fortes, R.P., "An Investigation On The Impact Of MDE On Software Reuse", *In 2012 Sixth Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS),* IEEE, September 2012, pp. 101-110.

[44] Keswani, R., Joshi, S. and Jatain, A., "Software Reuse In Practice", *In 2014 Fourth International Conference on Advanced Computing & Communication Technologies,* IEEE, February, 2014, pp. 159-162.

[45] Badampudi, D., Wohlin, C. and Petersen, K., "Software Component Decision-Making: In-House, OSS, COTS Or Outsourcing-A Systematic Literature Review", *Journal of Systems and Software*, Vol. 121, 2016, pp.105-124.

[46] Gusev A, Ilin D, Kolyasnikov P, Nikulchev E. "Effective Selection of Software Components Based on Experimental Evaluations of Quality of Operation". *Engineering Letters*. 2020 Jun 1;28(2).

[47] Khan, A. I., & Khan, U. A, "An Improved Model for Component Based Software Development", *Software Engineering*, Vol. 2, No. 4, 2012, pp. 138-146.

[48] Basha NM, Moiz SA. "Component based software development: A state of art",*In IEEE-international conference on advances in engineering, science and management (ICAESM-2012),* Mar 30, 2012, pp. 599-604.

[49] Padhy, Neelamadhab, Rasmita Panigrahi, and K.Neeraja."Threshold estimation from software metrics by using evolutionary techniques and its proposed algorithms ,models". *Evolutionary Intelligence* .2019:1-15.

[50] Tomer A, Goldin L, Kuflik T, Kimchi E, Schach SR, "Evaluating Software Reuse Alternatives: A Model And Its Application To An Industrial Case Study", *IEEE Transactions on Software Engineering*, Vol. 30, No. 9, Aug 24, 2004, , pp. 601-12.

[51] Shirali-Shahreza S, Shirali-Shahreza M, "Using Formal Methods in Component Based Software Development", *In Innovations and Advances in Computer Sciences and Engineering*, Springer (Dordrecht), 2010, pp. 429-432.

Journal of Theoretical and Applied Information Technology
15th January 2021. Vol.99. No 1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

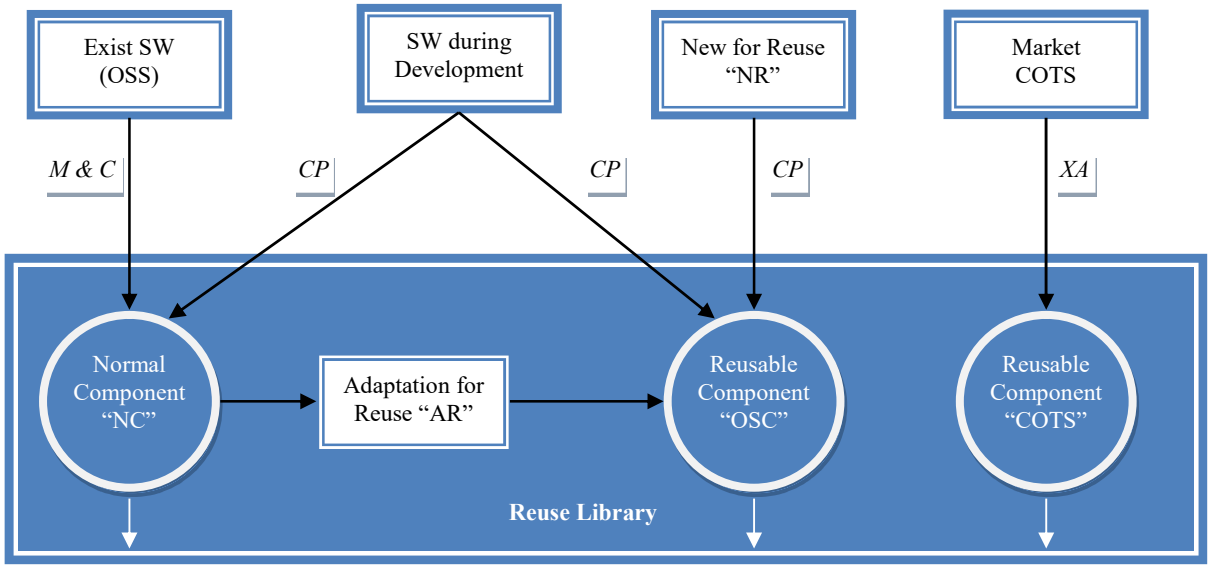

Figure 1:. Develop-for-reuse scenarios

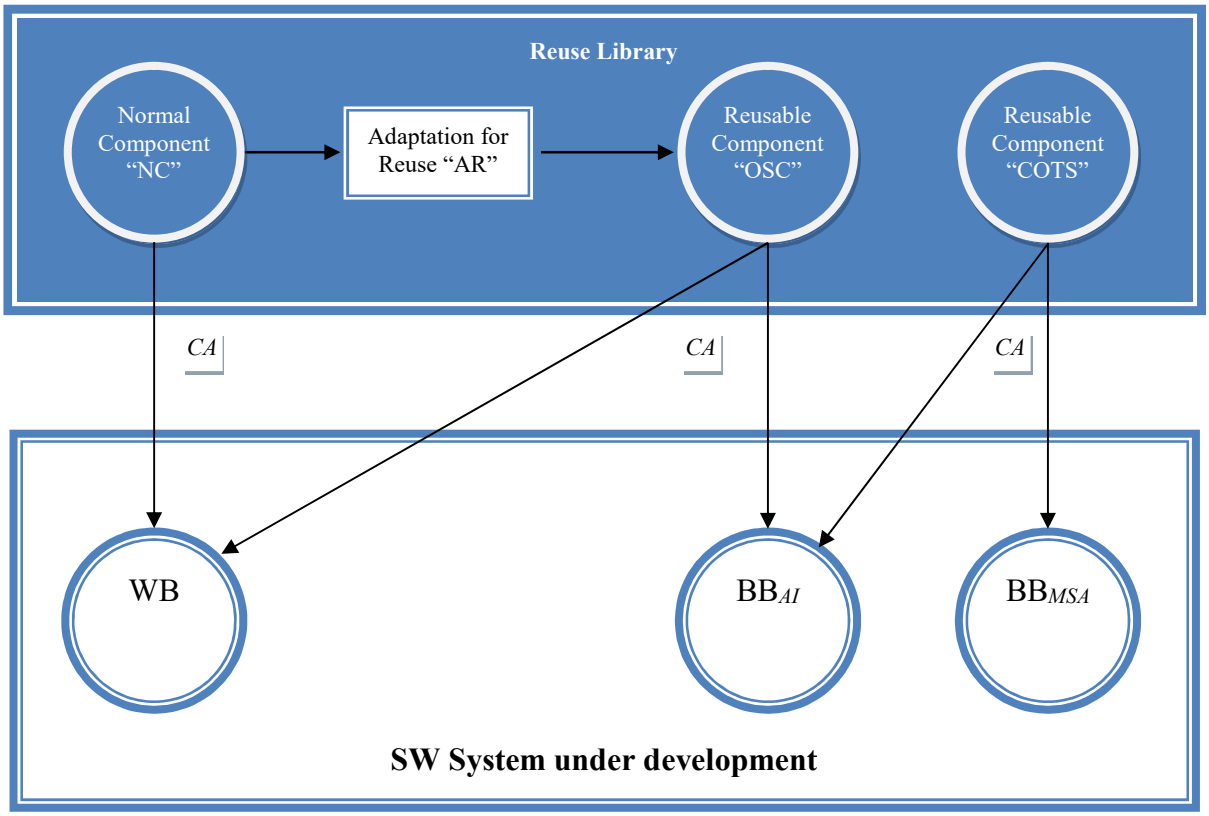

Figure 2: Develop-by-reuse scenarios