Reusable Software Components Framework

Article · November 2010

CITATIONS
5

READS
838

5 authors, including:

Anas Bassam AL-Badareen
Aqaba University of Technology
16 PUBLICATIONS **97** CITATIONS
SEE PROFILE

Mohd Hasan Selamat
Universiti Putra Malaysia
97 PUBLICATIONS **647** CITATIONS
SEE PROFILE

Marzanah A. Jabar
Universiti Putra Malaysia
62 PUBLICATIONS **355** CITATIONS
SEE PROFILE

Jamilah Din
Universiti Putra Malaysia
14 PUBLICATIONS **73** CITATIONS
SEE PROFILE

Reusable Software Components Framework

Anas Bassam AL-Badareen, Mohd Hasan Selamat, Marzanah A. Jabar, Jamilah Din, Sherzod Turaev

***Abstract*—** The concept of reusability is widely used, in order to reduce cost, effort, and time of software development. Reusability also increases the productivity, maintainability, portability, and reliability of the software products, which it has been evaluated before in other software products. The problems faced in software engineering not a lack of reuse, but a lack of widespread, systematic reuse. They know how to do it, but they do it informally. Therefore, strong attention must be given to this concept. This study aims to propose a systematic way of software reuse: build-for-reuse and build-by-reuse. The evaluation criteria of the reusable components are then adopted. Finally, the proposed framework is applied on C-Registration System.

***Keywords*—** Software Reusability, Build for Reuse, Build by Reuse, Reusability Criteria.

I. Introduction

SOTWARE reusability represents the ability to use part or the whole system in other systems [1-4] which is related to the packaging and scope of the functions that programs perform [5]. Therefore, it is used to reduce the effort, cost, and time to develop a new system. According to Poulin [6], the US department of defense alone could save 300$ million annually by increasing its level of reuse as little as 1%. Moreover, software reusability aimed to improve productivity, maintainability, portability and therefore the overall quality of the end product [7].

Software reuse is an important and relatively new approach to software engineering [8]. According to Gomes [9] a new horizons are opened, since the idea of software reuse appeared in 1968.

According to Ouyang [10], software reusability can be considered from two view points: design-by-reuse and design-for-reuse. Software-by-reuse is the use (assemble) of the existing application or its components to build new application. Software-for-reuse is the ability of building applications that can be used at all or part of it in other applications.

According to Prieto-Diaz [11], the concept of the software reuse is not only applied to source code fragment, but also it is the all of the information are related to the product generating processes, including software requirements, analysis, design, and any information required by the developers to build a software. Hence, the problems faced in software engineering not a lack of reuse, but a lack of widespread, systematic reuse. Further, software engineers know how to adapt and reverse-engineer systems, but they do all of these processes informally.

Bitar [12] showed that, the reusability has to be considered as it is the most significant factor to improve the productivity and quality of the software development. Moreover, Ramamoorthy [13] mention that the reusability is not limited to the source code, but it has to include the requirements, design, documents, and test cases besides the source code.

II. The Reusable Components Extraction

Software product has to be designed and developed in certain way even it requires an extra effort in order to be able to use it later [13]. The process of extraction of reusable component starts from the project planning and match with all of the software development stages (see Figure 1). Thus it increases the quality of the reusable components and reduces the efforts required to build by reuse software.

In planning phase, the objective of the reusability is defined. The planning of component extraction identifies which component is needed to be extracted, how the process of extraction will be conducted, and when and where the extracted components can be used. Furthermore, the independency and generality of the system goals are considered.

The requirement phase is the most important phase in the build for reuse process, which makes the decision of build for reuse. According to the user requirements the software developers decide which subsystem can be built for reuse. Moreover, the reusable components library is checked to see whether these components exist. The decision is made when the required component does not exist in the library.

In the system analysis and design, after the decision of build reusable components for specific requirements, the generality, portability, and coherence and coupling are the main issues of this phase. These characteristics are considered in design

Manuscript received September 30, 2010. This work was supported in part by Fundamental Research Grant, Ministry of Higher Education Malaysia, Project number: 03-04-10-867FR

Anas Bassam AL-Badareen is with the University Putra Malaysia, 43400 UPM, Serdang, Selangor, Malaysia. Phone: 601-72301530; e-mail: anas_badareen@hotmail.com.

Mohd Hasan Selamat is a professor of software engineering with University Putra Malaysia. Phone: 603-8946 89471759; Fax: 603-8946 6577; e-mail: hasan@fsktm.upm.edu.my.

Marzanah A. Jabar PhD is with University Putra Malaysia, 43400 UPM Serdang Selangor, Malaysia; e-mail: marzanah@fsktm.upm.edu.my.

Jamilah Din PhD is with University Putra Malaysia, 43400 UPM Serdang Selangor, Malaysia; e-mail: jamilah@fsktm.upm.edu.my.

Sherzod Turaev is a Postdoctoral Researcher with University Putra Malaysia, 43400 UPM, Serdang Selangor, Malaysia; e-mail: sherzod@fsktm.upm.edu.my.

reusable components and integrate these components to work together, in order to perform the intended requirement.

In the implementation phase, the independent sub-systems are developed, and the relationships among these sub systems are considered. Moreover, in the test phase, the evaluation of each sub system is independently conducted, and the ability of the sub-systems to work together and produce the expected result is considered. Furthermore, in system documentation, more attention is given to the ability of the sub-system to work and its requirements to perform its goals.

As a result of the project, two software products are delivered instead of one. The first one is the required system, which is delivered to the market. The second product is the reusable components that are considered in the system. These components are sent to the reusable process.

The reusability phase is the process of evaluation and enhancement of the reusable components according to certain standard. Ramamoorthy [13] proposed a method of reusability-driven development. The concept of this method is adopted in the proposed framework (Figure 1), while the characteristics evaluated in this method are modified.

The reusability test consists of formal conditions that are required in the library. The process of the reusable components is conducted till it passes the reusability test. Finally, when the component passes the reusability test, it is saved in the library to be used in other systems.

III. The Reusable Components Adoption

The reusability concept is managed to use some components in the new system. According to the business objectives, the software developers identify components that are able to be adopted in new system.

In the requirement phase, the system analyst checks the library whether it contains any reusable component suitable for each requirement. Moreover, the evaluation of the selected components is conducted, in order to see whether any modification is required to adopt it to the system.

The reusability test in the build-by-reuse is different than the test in the build-for-reuse. The test for reuse considers the conditions which are required by the reusable components library, whereas the test of the reusability in the build-by-reuse process considers the conditions which are required by the new system. Moreover, the conditions required by the library consider only the nonfunctional requirements, while in the test for reuse functional and nonfunctional requirements are considered.

In the system analysis and design phase, the design of the reusable components first, in order to consider their need in the new components. In the implementation phase, the sub systems are seated as the base of the system, in order to consider their requirements in the other sub-systems. Finally, the evaluation of the sub-system and the system integration are conducted. Figure 2 shows the framework of build-by-reuse process.

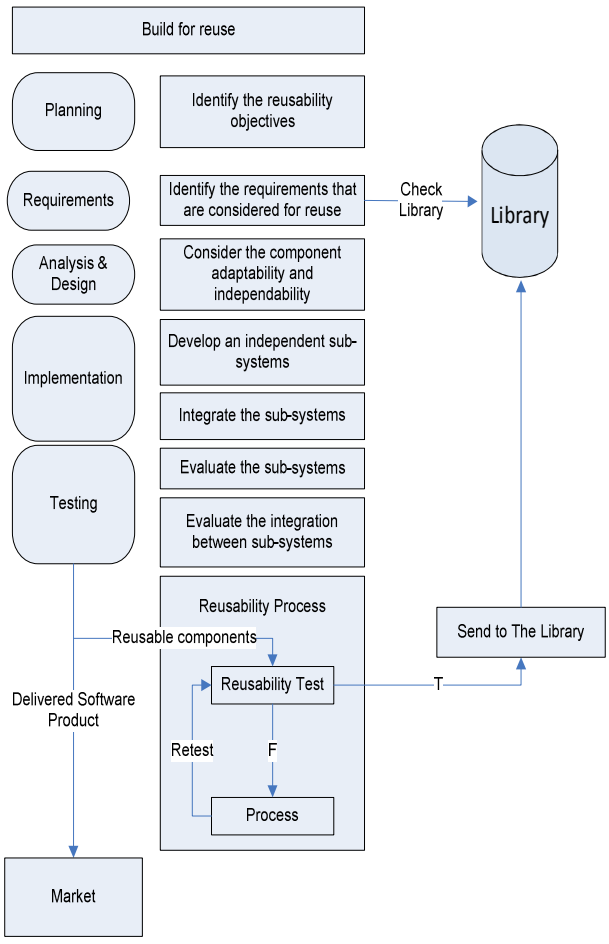

Figure 2: Build for Reuse Processes

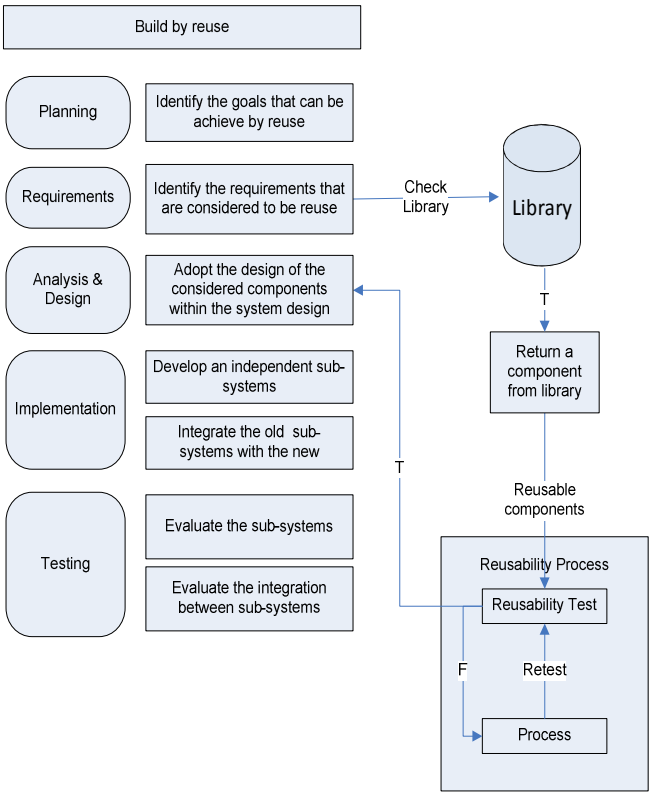

Figure 1: Build by Reuse Processes

ISBN: 978-960-474-250-9

IV. Transfer From For Reuse to By Reuse

The link between develop for reuse and develop by reuse is very important in the success of the new project development (see Figure 3). In order to decide whether the intended components are able to achieve their functions properly within other system, the project manager evaluates the ability of the components to work and communicate with other components within other systems in different platforms.

The reusability process, as mentioned above, considers the ability of the reusable components to achieve certain conditions that are required by the reusable component library or the new project.

The reusable component library is the repository of the components that are able to be used in different systems. The components existing in this storage have to be followed certain

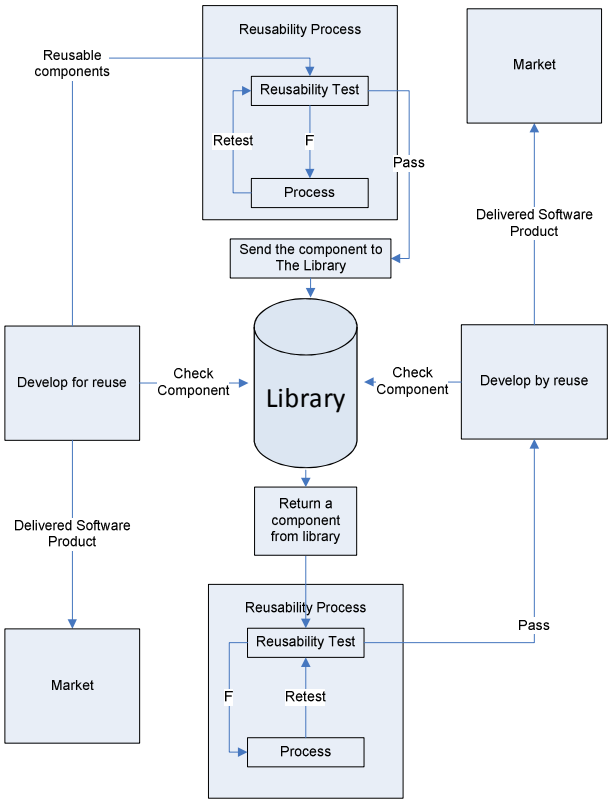

Figure 3: Transfer the System From For Reuse to By Reuse

standard or conditions. Hence, the reusability test evaluates whether the new component needed to store in the library achieves these conditions. If the component passes this test, it will be send directly to the library; otherwise, the process is required to modify this component in order to achieve the required conditions for the library.

In the component adoption, the required components returned from the library have to be sent to the reusability test. This test is different from the reusability test in the extraction process. The components which are returned from the library have been verified according to the library conditions. But in this test, it evaluates whether the component is suitable for the new system.

V. The Characteristics of the Reusable Components

Hence, the reusability concern about how to use a system or its part in other system, several characteristics have to be considered. In this section the characteristics required to make the system able to be used in different system are considered. The first test which is required to store the components in the library considered the general characteristics that are required for any system (Figure 4).

Software coexistence considers the ability of the system or the sub-system to work in different platforms. This characteristic includes software system independence and machine independence. Software system independence represents the degree to which the program is independent of nonstandard programming language features, operating system characteristics and other environment constraints. Machine

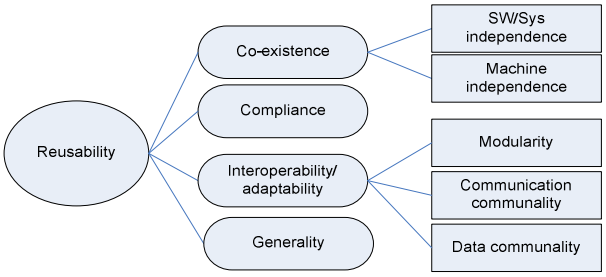

Figure 4: The Characteristics of the Software Reusability

independence is the degree to which the software is de-coupled from its operating hardware.

The adaptability (interoperability) of the software is the degree of the software to communicate with other systems. It includes modularity, communication communality, and data communality. System modularity is a functional independence of program components that represents the degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.

Communication commonality represents the degree to which standard interfaces, protocols and bandwidth are used. Data commonality is explicit use of standard data structures and types throughout the program.

Software generality represents the degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components. The compliance verifies whether the software follows any standard or international certificates in order to build reusable software.

The second test is required to retrieve the stored components from the library, in order to build new system. This test considers specific characteristics that help in the

ISBN: 978-960-474-250-9

system development. These characteristics are component suitability, documentation, and modifiability.

The suitability of the component is measured in order to see whether this component is able to perform the intended function properly. The documentation of the component simplifies the job of the software developer to understand it. Component modifiability is required to change the components as it is required in the new system.

VI. Results and Discussions

C-Registration System is documented in the Rational Unified Process (RUP Version 2003.06.00.65) document as an example of Web site project. This system enables the students to register for courses on-line, professors to select their teaching courses and to maintain student grades, and registrar to maintain professor and student information.

The proposed framework is applied on Maintain Professor Information function within this system. Table 1 shows the general information required in Reusable Software Components Library. In addition to this information, more detailed description is required such as use case specification, sequence diagram (Figure 5), and source code.

The main form consists of general information about the software component. The name of the component is modified to be more general. The sub-system maintaining professor information is changed to maintain user information, which is more general and suitable for many types of functions. In addition, to create forms, the form of the component modification is also considered.

The component characteristics represent the ability of the software component to achieve certain level of quality criteria. The values of these criteria are presented in details instead of specific value that considered in order calculating the re-

Table 1: Reusable Software Component Library Form

Component Name: *Maintain User Information*		**ID:** *1*	**Created Date:** *23-9-10*	**Responsible:** *Anas*
User Objective: *Allow the administrator to maintain the users' information (Add, Delete, and Update)*				
Actor: *Administrator*	**Trigger:** *the admin selects " Maintain User Information" activity from Admin menu*			
Component Characteristics				
Co-existence	SW/System Independence	The component passed the test of work with MS Windows XP, 2007, and Unix OS.		
	Machine Independence	The components passed the test of 32 and 64 Bit PC		
Compliance				
Interoperability	Modularity	This component is developed as a one independent part		
	Communication Communality	The ports of communication are defined in Process flow Part		
	Data Communality	The ports of communication and data type are defined in Process flow and attributes Parts		
Generality		The component is able to work with data described in Attribute Part		
Process Flow				
Inputs		**Output**		
Description	**Source**	**Description**	**Destination**	
User Info. Request	*Admin*	*User info*	*Admin*	
New user info.	*Admin*	*UserID*	*User Info. DB*	
Modified User Info.	*Admin*	*New User Info.*	*User Info. DB*	
Deleted User Info.	*Admin*	*Updated User Info.*	*User Info. DB*	
User Details	*User Info. DB*	*Deleted User Info.*	*User Info. DB*	
Data Store				
Description: *this entity is used to store a general user information*				
Accessibility				
Source	**Type**	**Description**		
Admin	*Create, Modify, Deleted, View*			
User	*View*			
Attributes		**Primary Key:** *UserID*		
Attribute	**Description**	**Acceptable Value**	**Format**	**Constraint**
UserID	*Identifier attribute*	*Number only*	*9 digits, no space*	*Unique, Not null*
Affiliation		*Text*	*30 characters*	*Not null*
Department		*Text*	*30 characters*	*Not null*
Occupation		*Text*	*30 characters*	*Not null*
Address		*Text*	*50 characters*	*Not null*
Phone		*Number*	*20 characters*	*Not null*
Mobile		*Number*	*20 characters*	*Not null*

ISBN: 978-960-474-250-9

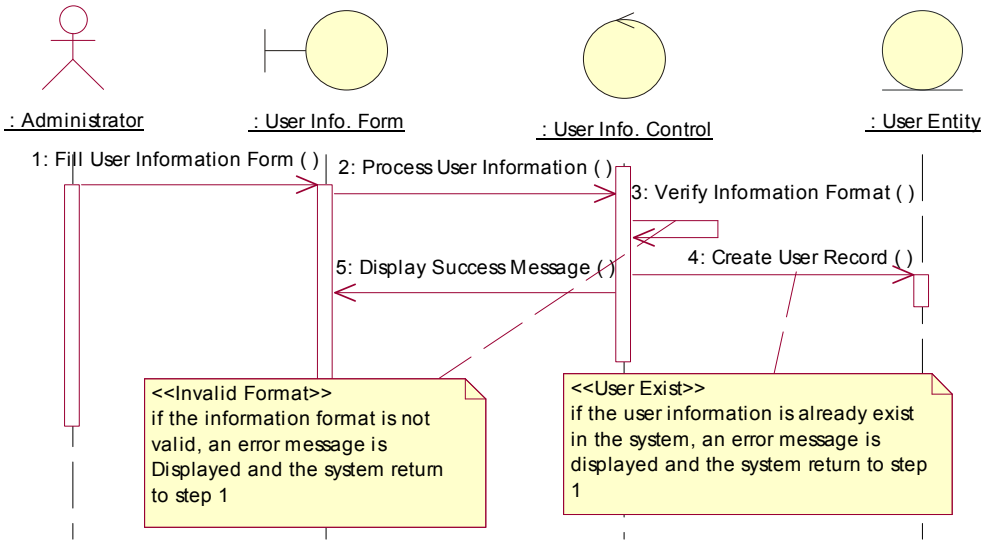

Figure 5: The Characteristics of the Software Reusability

usability of the component. This description is clearer, usable, and specific in order to understand the real situation of the component, which is more helpful for developing by reuse processes.

The process flow represents the ports of the components used to communicate with other components or external systems. Moreover, the description of these ports shows the type of action and data that are allowed to go through it.

The data store shows the storage required by the component in order to perform its tasks. The characteristics of the required storage and the information shared with other components or systems are included. That is used to have a clear picture of the information status, especially for security purposes.

The attributes of the storage are included, which shows the information formats that the component can deal with, and the special roles are considered in the component. Moreover, this information shows the coupling between the component and the other components or systems, such as the primary keys and foreign keys.

The component interoperability is detailed clears in the process flow and data store. The process flow shows the functions required from other components. The attributes in data store shows the data that required from other components or external systems.

VII. Conclusion

The problems faced in software engineering not a lack of reuse, but a lack of systematic reuse. They know how deal with, but they do it informally. This study proposed a reusable software component framework. The framework consists of process of extraction, evaluation, processing, adoption, and store reusable software component. The experimental has been done through apply the framework on C-Registration system. The result shows that the proposed framework simplifies the job of software developers, which required a little effort in order to build for reuse or build by reuse.

For future work, the proposed framework is managed to be implemented, in addition to be applied in different type of systems.

References

[1] J. A. McCall, *et al.*, "Factors in Software Quality," *Griffiths Air Force Base, N.Y. Rome Air Development Center Air Force Systems Command,* 1977.

[2] N. S. Gill, "Reusability issues in component-based development," *SIGSOFT Softw. Eng. Notes,* vol. 28, pp. 4-4, 2003.

[3] C. Luer, "Assessing Module Reusability," in *Assessment of Contemporary Modularization Techniques, 2007. ICSE Workshops ACoM '07. First International Workshop on*, 2007, pp. 7-7.

[4] F. Haiguang, "Modeling and Analysis for Educational Software Quality Hierarchy Triangle," in *Web-based Learning, 2008. ICWL 2008. Seventh International Conference on*, 2008, pp. 14-18.

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

[6] J. S. Poulin, "Measuring software reusability," in *Software Reuse: Advances in Software Reusability, 1994. Proceedings., Third International Conference on*, 1994, pp. 126-138.

[7] A. Sharma, *et al.*, "Reusability assessment for software components," *SIGSOFT Softw. Eng. Notes,* vol. 34, pp. 1-6, 2009.

[8] M. Burgin, *et al.*, "Software technological roles, usability, and reusability," in *Information Reuse and Integration, 2004. IRI 2004. Proceedings of the 2004 IEEE International Conference on*, 2004, pp. 210-214.

[9] P. Gomes and C. Bento, "A case similarity metric for software reuse and design," *Artif. Intell. Eng. Des. Anal. Manuf.,* vol. 15, pp. 21-35, 2001.

[10] Y. Ouyang and D. L. Carver, "Enhancing design reusability by clustering specifications," presented at the Proceedings of the 1996 ACM symposium on Applied Computing, Philadelphia, Pennsylvania, United States, 1996.

[11] R. Prieto-Diaz, "Status report: software reusability," *Software, IEEE,* vol. 10, pp. 61-66, 1993.

[12] I. Bitar, *et al.*, "Lessons learned in building the TRW software productivity system," 1985.

[13] C. V. Ramamoorthy, *et al.*, "Support for reusability in Genesis," *Software Engineering, IEEE Transactions on,* vol. 14, pp. 1145-1154, 1988.

ISBN: 978-960-474-250-9