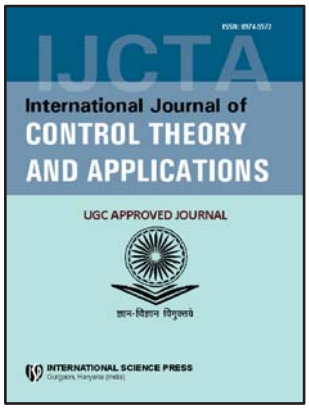

International Journal of Control Theory and Applications

ISSN : 0974-5572

Volume 10 • Number 31 • 2017

A Component-Based Model for Software Reusability

G. Manivasagama M. Anand Kumar[b] and B. Bharathi[c]

[a]*Research Scholar, Department of Computer Science, Karpagam University, Coimbatore, Tamilnadu, India*

E-mail: mani.mca.g@gmail.com

[b]*Associate Professor, Department of Information Technology, Karapagam University, Coimbatore, Tamilnadu, India*

E-mail: anand2kumarm@gmail.com

[c]*Assistant Professor, Department of Information Technology, Karapagam University, Coimbatore,Tamilnadu,*

E-mail: India drop2bharathi@gmail.com

Abstract: Component-based model used to develop and combine the product components which make easy for the software reusability with high quality and simple cost effective for testing . Component based software development approach makes use of already surviving software components to create new application developments. One of the most vital activities in the reuse based process is the selection of suitable components. The software development cost and time simultaneously reduced the paper investigates the existing model and proposes a new model to further improvement software quality with the cost reduction

Keywords: *Component-based model, Reusability, Software quality.*

1. INTRODUCTION

The main idea of the component-based model is building the systems from the already existing components. The model has several benefits: to improve efficiency, enhance the ability to reuse components and maintain the complexity, reducing the overall time and to find how much time need to develop software, decreasing of production costs through software reuse, enhancing the quality of system, reducing maintenance costs, increase of development so that the advantages for the Component-based development is reducing the development time and cost and improving the software quality and maintainability.

Component assessment is the core of the component selection process. Component quality models have put forward to decide upon a measure against the prospect components. It can be analyzed and then compare with other components. Component-based model reduces the development time and cost and improves the software quality and maintainability.

The component-based software engineering process consists of the following six phases:

1. Requirements Analysis,
2. Design

3. Component identification and Customization
4. System Integration
5. System Testing
6. Software Maintenance

1.1. Requirements Analysis

In Requirements Analysis phase all the component requirements like functional, and non-functional are collected and are analyzed and specified based on a well-defined methodology such as UML. The result of this phase is a component specification document.

1.2. Design

In Design phase engineers design components, based on the component requirements specification from the previous phase. The component design includes three tasks. The first task is to conduct component design for the functional logic and data objects and make trade-off decisions on technologies and operation environments. The second task is to follow a selected component model and to work on component realization by providing data exchange mechanisms for component communication and interactions. The final task is to define consistent approaches to support component packaging and to deployment. The outcome of the phase is the Design Specification Document.

1.3. Component identification and Customization (Coding)

It is mentioned in the earlier sections that reusability and reusable components form the backbone of the CBSD process. In the phase suitable components are identified and are customized. A specifying new component is done in the previous phases. Implementation of the components is performed using a specific technology and programming language. It is based on the design and targeted operating environments. The focus is to compose and assemble components that have to develop separately, and even independently. The Component identification, customization and integration are the crucial activities in the life cycle of Component Based systems. It includes two main parts:

1. Evaluation of each candidate components is based on the functional and quality requirements that will be used to assess the component.
2. Customization of the candidate components is to be modified before being integrated into a new Component Based software systems.

1.4. System- Integration

It is possible for a component to be implemented for more than one operating environment. Each implemented component depends on a specific technology set and targeted operating environment. Each component that is identified and customized are integrated together to meet the specifications. Integration is to makes key decisions and provide communication and coordination among various components of a target software system.

1.5. System-Testing

System testing is conducted by separate testing team, Follows Black Box testing techniques depends on Software Cost. Build level testing to It validate internal processing and depends on external interface processing.

2. RELATED WORK

Based on above mentioned model several process model are as follows

2.1. Waterfall model

The Waterfall Model was first created by Royce in 1970. Its linear model is different phase and are inter connected so that result of one phase becomes subsequently input values for the next phase. The Waterfall Model is mostly used by software engineers and becomes the important prevalent software life cycle model.

The model is initially developed to identify the phases inside the software development as a linear series of actions which must be existed before the next is commenced.

Drawback

1. Main drawback of Waterfall Model is inflexible division of phase. Today requirements are changing quickly in all the phase and overlaps.
2. In the Waterfall Model output are verified in the last stage.

Therefore Waterfall Model is efficient for such a situation and are finely defined.

2.2. Incremental Model

To overcome the drawback of the Waterfall Model a cycle model called as Incremental Model was introduced by in the Incremental Model the characteristics of linear model and continuous characteristic of prototype life cycle model, are joined. In the incremental model partial implementation of systems construct build after improving the execution require functionality achieve.

Challenges : It is time consuming process. The user of the systems involve the whole process.

2.3. Evolutionary model

Evolutionary prototypes provide the incremental software development, so that software systems may be gradually developed and tested. It allows the major bugs to be exposed and corrected early, which means that they are often cheaper to plan , without any idea to manage sequences, this process cannot be generate into uncontrollable hacking.

2.4. Prototyping model

Normally customer of the software system initially contains broader requirement of the systems. In the study a immediate prototype can be build. Prototyping provides constructive feedback to the designers and the potential customers. The system needs can be solved and refined early during the software development. The prototype is corrected by the users and improves his needs.

2.5. Spiral Model

The Spiral Model was first introduced by Boehm, is an development software life cycle model. In the spiral life cycle model various kinds of phase are represented as a spiral rather than the series of behaviour. The Spiral Model makes the software development more flexible and propose, mainly to the accuracy of the software development through prototyping. In the Spiral Model the software is developed in a sequence of incremental release. The Spiral Model is separated into a number of regions; typically these are between three to six.

2.6. V. Model

In the V model the process begins with a usual way by the requirements engineering and specification, followed by system specification.

In a Non-Component-Based approach the process would continue with the unit design, implementation and test. Instead of performing the time and efforts consuming choose appropriate components and integrate them in the system.

2.7. Y. Model

Mainly top-down or bottom-up strategy of software production is not quite appropriate. The Y model preaches atop-down or bottom-up fashion for software development.Thesoftware engineer meet have the knowledge about the application domain. The knowledge naturally determines the prevailing strategy of the software development.

2.8. A Model

The A shape Model developed to support the parallelism and evolutionary design of the application and test plans.

2.9. Agile

Agile development model is a type of Incremental Model. The Software is developed in the incremental and rapid cycles. It results in the small incremental releases with each release building on the previous functionality. Each release is thoroughly tested to ensure the software quality and it is maintained. It is used for critical time applications. Extreme Programming (XP) is currently one of the most well known agile development life cycle model.

Pros:

1. Flexibility to make changes to the requirements
2. Testing is integrated from the beginning to the end
3. Speedup to market
4. Improved Risk Management

Cons:

1. Projects can run longer than the anticipated
2. Requires high level commitment of time and energy from the developers

Challenges : The model does not provide a notation or a set of constraints to increase reusability.

3. DISCUSSION

3.1. Is there any Benefits of Software Reusability?

88% of the article agrees that there are many benefits of software reusability. By reviewing them, it seems that the authors agreed that the major benefits are the following:

Increase productivity : Software reusability improves the productivity by using the existing software products the smaller ones are creating from scratch.

According to Singh [4], the concept of reusing the available software components consideras a key feature in developing productivity.

Minimize cost : The rate of developing software from scratch can be store by identifying and extracting. The reusable components from the already developed and existing systems or legacy systems [2].

Improve quality : A good software reuse assists the increase of reliability and quality [1].

According to [24] the improve software quality uses any software over time contains lot of errors which were not discernible when it was created. Therefore, a software product reused many times will contain low bugs and defects than the newly created software.

Increase dependability : Increase dependability will decrease the time of the software development since it reduces the development failures [16]. Reused software must be tried and tested in the working systems. It should be more reliable than new software [6].

Accelerate development : Reusing software can be an accurate system production because the both development and validation time will be reduced [16]. According to [17] Reusing the software to create a new software product can reach the market on the time for satisfying the customer requirements.

Generalized software components can reduce the time of product construction and delivery of the software.

Reduce process risk : If software exists, there is a low uncertainty in the costs of decreasing software the in the costs of development. It is an important factor for the project management as it reduces the margin of error in the project cost estimation. It is specifically true when relatively large software components such as sub-systems are reused [16].

According to [17] Risk is reduced in developing a new software when reusable components already encompass the desired functionality and have the standard of interfaces to facilitate integration.

3.2. Is the Study List the Different Levels of Software Reusability?

From the literature survey it was identified that 71% of the articles proposed the explicitly or implicitly of different reusability levels in the software life cycle. Reuse is *separated* into following levels

Specification reuse : Understanding the building of the software development is one of the most tedious aspects of software development, because sometimes customers do not really know their needs, so capitalizing the previously used abstract artefacts like requirement specification document may open the mind of software customers to more functionality that could have been overlooked. The reuse of specification is considered as a higher level of reuse [6]

Design reuse: The design processes is the most engineering disciplines based on the reuse of the existing systems or components. Software reusability is more specifically refers to the design features of a software element (or collection of software elements) that enhances its suitability for reuse [16].

This type of reuse is required when a system needs to be reported in an entirely different software or hardware environment [6].

Code reuse: In Computer Science and Software Engineering the reusability is the likelihood a segment of source code that can be used again to add new functionalities with a slight or no modification [16]. The reusability of a piece of code does not mean that it should be able to copy and paste the same code in several places within the application. In fact, it exactly means the opposite meaning.

A piece of reusable code means the same code can be reused in different places without overwriting [17]. The reusable code can be an object code, data objects, source code, or standard subroutines [6].

Application system reuse : An increasing number of organizations are using software just as all inclusive applications, as in the past, but also a component parts of larger applications. In the fresh role, acquired software must integrate with the other software functionality [16].

The Reuse of Application system is considered as a special case of software reuse. The complete system is reused by implementing through various range of operating systems and computers [6].

Test reuse: Reusable components are normally accompanied by the high quality documentation and previously developed tests plans and cases.

4. STUDY OF QUALITY REVIEW ASSESSMENT

The quality assessments are based on a checklist of factors/questions that needs to be evaluated in each study. For assessing studies, the following questions are defined:

RE1: Does the study mention the software reusability approaches?

RE2: Does the study present any benefits to the software reusability?

RE3: Is the study list out the different levels of the software reusability?

RE4: Does the study report any barriers to software the reusability?

RE5: Does study the propose any maturity model for the reuse?

RE6: Does study propose any attributes which affect the reuse?

It is evaluated bellow:

RE1: Y (Yes) the study proposed some software reusability approaches. P (Partially) the study mentioned one or more approaches, but did not describe it. N (No) the study did not propose any approaches.

RE2: Y, study mentioned more than one benefits of software reusability clearly. P, benefits are implicit. N, study does not mention any benefit.

RE3: Y, study defined some levels of software reusability. P, reusability levels are implicit. N, study did not present any levels.

RE4: Y, study mentioned some barriers of reusability explicitly. P, reusability barriers are implicitly reported. N, study did not report any barriers.

RE5: Y, study proposed some maturity models for reuse. P, study mentioned one or more maturity model, but did not describe it. N, study did not propose any maturity model.

RE6: Y, study mentioned attributes which affect the reuse explicitly. P, attributes are implicit. N, study did not mention any attribute These data were extracted from each paper:

1. Title and year of publication
2. Author(s) information
3. Research issues
4. Main topic
5. The full source and references.

5. SELECTED STUDIES FOR REVIEW

Table 1

S1	Minimal information for reusable scientific software	C. Hong	Looks at the concept of software reusability from the perspective of the software engineer and the researcher	2014
S2	Reusability in Component Based Software Development - A Review	S. Thakral, S. Sagar and Vinay	A literature review of various software reusability concepts is presented	2014
S3	Software Reuse in Practice	R. Keswani, S. Joshi, A. Jatain	Summarized software reuse research and discussed major research contributions.	2014
S4	Impact of Quality Attributes on Software Reusability and Metrics to assess these Attributes	C. Monga, A. Jatain, D. Gaur	Studied various attributes or factors that affect the reusability of software. The most common factors are identified and their impact is analyzed.	2014
S5	Taxonomy, Definition, Approaches, Benefits, Reusability Levels, Factors and adaptation of Software Reusability: A Review of the Research Literature	Y. Y. Ibraheem, A. M. Abualkishik and M. Z. Mohd Yussof,	Provided a systematic review of the concept of reusability, identifying the definition, Approaches, Benefits, Reusability Levels, Factors and adaptation of Software Reusability	2014
S6	Feature Prioritization for Analyzing and Enhancing Software Reusability	Md. Iftekharul A. Efat, Md. S. Siddik, M.Shoyaib, S. M. Khaled	An analysis of the various attributes from the organization, development and complexity perspective, an optimized group of properties are proposed	2014
S7	A Framework for Assessing the Software Reusability using Fuzzy Logic Approach for Aspect Oriented Software	P. K. Singh, O. P Sangwan, A. P. Singh, A. Pratap	Explored the various metric that affects the reusability of aspect oriented software and Estimate it using fuzzy logic approach.	2015

Study and Quality Review Evaluation of the Study

Table 2

SOURCE	*RE 1*	*RE 2*	*RE 3*	*RE 4*	*RE 5*	*RE 6*
S1	N	Y	N	N	Y	P
S2	P	Y	Y	Y	N	Y
S3	N	Y	P	Y	N	N
S4	P	N	N	N	N	Y
S5	Y	Y	Y	P	N	Y
S6	Y	Y	P	N	N	Y
S7	Y	Y	N	N	N	Y

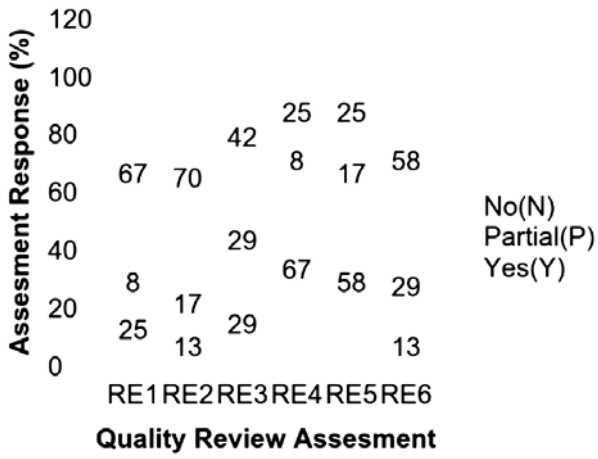

Figure 1: Study of Quality assessment review results per question and type of assessment response

Fig. 1 shows the coverage of every quality review assessment (RE) in the included studies. It shows that RE1, RE2, RE3 and RE6 were covered in a rate higher than 80% by Yes and partially answered. That means that 80% of the studies cover the approaches, benefits, levels and attributes of software reusability. On the contrary, RE4 and RE5 were covered in a rate higher than 50% by No. Which means the few works examined the barriers of reusability (RE4), which can motivate the organizations to adopt software reusability approaches? Moreover, the studies about maturity models to software reusability are limited, which highlights the need to explore this domain in order to help organizations auditing his maturity reuse levels.

6. PROPOSED MODEL

The life cycle of components based software consists of three stages

1. The Design phase, the components are selected by the repository or designed, defined and constructed
2. The Integration phase, the component are integrate with others component
3. The run-time phase, the component binaries are instantiated and executed in the running system.

A software component life cycle model should define the following questions

4. What are sequence components composition followed?
5. Why these sequences are needed?
6. How to compose the components?

Over the past three decades, several component based software development methodologies have appeared. Such methodologies address some or all phases of the software life cycle ranging from requirements to maintenance.

These methodologies have often been developed in responses to the new ideas about how to cope with the inherent complexity of software systems.

7. CONCLUSION

Although software reusability can significantly improve the productivity and quality of a software product, it is considered as a difficult task especially for legacy software. In this study, we presented a literature review of the most up-to-date research work published on software reusability. This review of various software reusability concepts offers a good understanding of reusability for accelerating the adoption of reusability in software development.

The Study found study that few works examined as a barriers of reusability, which can motivate organizations to adapt software reusability approaches. Also the studies about maturity models of software reuse are limited, so exploring this domain for helping organizations to audit his maturity reuse levels, can be a subject of a future work.

The Problem with the existing models is based on the requirement to prepare a Prototype and to get the acceptance from the user and implement it But in the proposed system there are a lot of existing prototypes for all software's while the requirement is collected these prototypes are shown to the user and get the acceptance and implemented Different modules have different design and implementations. So concurrent design for various models leads to the effective finish project in time.

REFRENCES

[1] N. S. Gill, S. Sikka, "Inheritance Hierarchy Based Reuse & Reusability Metrics in OOSD", International Journal on Computer Science and Engineering (IJCSE), Vol. 3, n. 6, 201, pp. 2300-2309.

[2] B. William, Frakes and Kyo Kang, "Software Reuse Research: Status and Future", IEEE T ransactions on software engineering, Vol. 31, n. 7, 2005.

[3] B. Jalendar, A. Govardhan and R. Emchand, "Desiging code level reusable software components", International Journal of Software Engineering & Applications, Vol. 3, n. 1, January 2012, pp. 219-229.

[4] Y. Singh, P. K. Bhatia, O. Sangwan1, "software reusability assessment using soft computing techniques", ACM SIGSOFT Software Engineering Notes, Vol. 36, n. 1, January 2011, pp. 1-7.

[5] K. Kaur, N. Mohan and Dr. P. S. Sandhu, "Reusability of Software Components using J48 Decision Tree", Proceedings of the International Conference on Artificial Intelligence and Embedded Systems, 2012, pp.69-71.

[6] Y. Y. Ibraheem, A. M. Abualkishik and M. Z. Mohd Yussof, Taxonomy, "Definition, Approaches, Benefits, Reusability Levels, Factors and adaptation of Software Reusability: A Review of the Research Literature", Journal of Applied Sciences, Vol. 14, n. 20, 2014, pp. 2396-2421.

[7] B. Kitchenham, S. Charters, "Guidelines for performing systematic literature reviews in software engineering", EBSE, 2007.

[8] E. S. Almeida, A. Alvaro, and D. Lucrédio, V. C. Garcia, and, S. R. L. Meira, "RiSE Project: Towards a Robust Framework for Software Reus", Proceedings of the In IEEE International Conference on Information Reuse and Integration (IRI), 2004, pp. 48–53.

[9] O. P. Rotaru, M. Dobre, "Reusability Metrics for Software Components", Proceedings of the the 3rd ACS/IEEE International Conference, 2005, pp. 24.

[10] J. Finnigan, J.Blanchette, "A Forward-Looking Software Reuse Strategy", Proceedings of the IEEE Aerospace Conference, 2007, pp. 1-9.

[11] V. Garcia, D. Lucrédio, A. Alvaro, "Towards a Maturity Model for a Reuse Incremental Adoption", Proceedings of Simpósio Brasileiro de Componentes, Arquitetura e Reutilização de Software (SBCARS), 2007, pp. 61-74.

[12] F. McCarey, M.O. Cinneide and N. Kushmerick, "Knowledge reuse for software reuse", Web Intelligence and Agent Systems, Vol. 6, n. 1, 2008, pp. 59-81.

[13] R. Kamalraj, B.G Geetha, G. Singaravel, "Reducing efforts on software project management using software package reusability", Proceedings of the IEE Advance Computing Conference, 2009, pp. 1624-1627.

[14] A.Sharma, P.S. Grover and R. Kumar, "Reusability assessment for software components", ACM SIGSOFT Software Engineering Notes, Vol. 34, n. 2, 2009, pp. 1-6.

[15] M. Dinsoreanu, I. Ignat, "A Value Analysis Model for Measuring Software Reuse", Proceedings of the Second International Conference IEE of Applications of Digital Information and Web Technologies(ICADIWT'09), 2009, pp. 846- 848.

[16] P.S. Sandhu, P. Kakkar, S. Sharma, "A survey on software reusability", Proceedings of the Second International Conference IEE of Applications of Mechanical and Electrical Technology (ICMET), 2010, pp. 769-773

[17] G. Singaravel, V. Palanisamy, A. Krishnan, "Overview analysis of reusability metrics in software development for risk reduction", Proceedings of the International Conference IEE of Innovative Computing Technologies (ICICT), 2010, pp. 1-5.

[18] K.S Jasmine, R. Vasantha, "A New Capability Maturity Model For Reuse Based Software Development process", IACSIT International Journal of Engineering and Technology, Vol. 2, n. 1, February 2010, pp. 112-116.

[19] W. Spoelstra, M. Iacob, M. Sinderen, "Software Reuse in Agile Development Organizations - A Conceptual Management Tool", Proceedings of the 2011 ACM Symposium on Applied Computing, 2011, pp. 315-322.

[20] B.Jalender, A. Govardhan,R. Emchand, "Designing code level reusable software components", International Journal of Software Engineering & Applications (IJSEA), Vol. 3, n. 1, January 2012, pp. 219-229.

[21] B. Koteska, G. Velinov, "Component-Based Development: A Unified Model Of Reusability Metrics", Proceedings of ICT Innovations 2012: Secure and Intelligent Systems, 2013, pp. 335.

[22] C. Hong, "Minimal information for reusable scientific software", Proceedings of the 2nd Workshop on Working towards Sustainable Scientific Software: Practice and Experience, 2014.

[23] S. Thakral, S. Sagar and Vinay, "Reusability in Component Based Software Development – A Review", World Applied Sciences Journal, Vol. 31, n. 12, 2014, pp. 2068-2072.

[24] R. Keswani, S. Joshi, A. Jatain, "Software Reuse in Practice", Proceedings of the IEEE International Conference on Advanced Computing & Communication Technologies (ACCT), 2014,pp. 159-162.

[25] C. Monga, A. Jatain, D. Gaur, "Impact of Quality Attributes on Software Reusability and Metrics to assess these Attributes", Proceedings of the IEEE International on Advance Computing Conference (IACC), 2014, pp. 1430-1434.