AN IMPROVED REPOSITORY STRUCTURE TO IDENTIFY, SELECT AND INTEGRATE COMPONENTS IN COMPONENT-BASED DEVELOPMENT

Muhammad Khamis Dauda, Reda M Salama and Rizwan Qureshi

Faculty of Computing and Information Technology,
King Abdulaziz University, Jeddah. Saudi Arabia

ABSTRACT

An ultimate goal of software development is to build high quality products. The customers of software industry always demand for high-quality products quickly and cost effectively. The component-based development (CBD) is the most suitable methodology for the software companies to meet the demands of target market. To opt CBD, the software development teams have to customize generic components that are available in the market and it is very difficult for the development teams to choose the suitable components from the millions of third party and commercial off the shelf (COTS) components. On the other hand, the development of in-house repository is tedious and time consuming. In this paper, we propose an easy and understandable repository structure to provide helpful information about stored components like how to identify, select, retrieve and integrate components. The proposed repository will also provide previous assessments of developers and end-users about the selected component. The proposed repository will help the software companies by reducing the customization effort, improving the quality of developed software and preventing integrating unfamiliar components.

KEYWORDS

CBD, COTS, Third Party Components, Repository, Quality Attributes.

1. INTRODUCTION

Every software organization aims to produce or to acquire qualitative software, to save time, cost and meet the market demand. During the analysis phase, suitable components are searched from both the in house repository and third party repository. A lot of time is spent in identification, selection, and analysis, which increases the customization and integration cost because the component's structure did not provide any insight to help the team with some basic information including quality constituents or attribute and the development team has no direction [1]. A lot of time is also spent during customization and integration of reusable components. It is highly likely that more suitable components might not be identified resulting in wastage of time, cost, effort and resources. There is a need to provide a structure to evaluate quality attributes of commercial off the shelf (COTS) or in house components during the selection phase. There is also a need to monitor the version control system, end-user feedback, quality of service (QoS) issues and maturity of the component vendor [2].

The paper is further arranged as follows. Section II covers the related work. The problem statement is described in section III. Section IV illustrates the proposed solution. The proposed solution is validated in section V. The recommendations and future work are covered in section VI.

2. RELATED WORK

The idea of component-based software engineering (CBSE) among the software community is not new and the software development companies are practicing it from more than two decades. Quality evaluation and metrics are required to know for every individual component before its reuse. However, a formal direct and an indirect component coupling metrics are proposed to measure the quality of components with respect to complexity and performance [3]. The research investigates the risk-management activities and their correlations with the occurrences of typical risks in the development of COTS systems [4]. It is achieved by exploring the occurrences of typical risks in COTS systems and furtherly the effectiveness of the risk reduction activities are compared.

The aim of the proposed research is to improve security and quality concerns of open-source software systems [5]. The major challenges of open source development are addressed. A model is proposed to assure quality [5]. The study explores the current state of tool building in the reverse engineering domain intending to improve upon the practice to a predictable format [6]. The symptoms of code smells are poor design and implementation choices [7]. The high cost of maintainability and customization is due to smelly code. A model is proposed to identify poor quality software, bug prediction, and bug classification using F-measures. It is important to understand the implication of choosing a suitable component from a third-party repository because of the trade-off in quality [8]. Software Engineering taught programs do not teach how to ensure that COTS components are not compromised from production to integration [9]. Data are generated during component execution, which can be distilled and mined [10]. The study in explores the challenges of DevOps including performance and quality [11]. A quality assurance model is proposed to implement during the phases of analysis, development, certification, customization, design, integration, testing, and maintenance [11].

Table 1. Limitations of Related Work.

Paper Title	Limitation
A Review of Component Coupling Metrics for Component-Based Development [3].	The idea is proposed to establish to measure the effectiveness of reusable components.
A State-of-the-Practice Survey of Risk Management in Development with Off-the-Shelf Software Components [4].	The results show that there are still unexplained risk-related factors in the proposed risk-reduction activities.
A Model for Quality Assurance of OSS Architecture [5].	The quality assurance model is very hard to use in software engineering.
Building Reverse Engineering Tools with Software Components: Ten Lessons Learned [6].	More case studies are needed to conclude the results
Toward a Smell-Aware Bug Prediction Model [7].	Only Apache data sets are used to validate the model.
Choosing Component Origins for Software Intensive Systems: In-House, COTS, OSS, or Outsourcing? —A Case Survey [8].	There is a need to conduct empirical studies to generalize the results.

The study in [12] focuses on software refactoring and quality enhancement. A pilot survey on data analysis claimed that the construction of a new framework for healthcare COTS evaluation and selection is necessary [13]. Most scientific software offer performance and maintainability

quality attributes at the highest priority [14]. As soon as new requirements are required to implement in a system, the developers must identify their attributes and impact on the existing components of respective system [15]. The study uses 18 matrices and 6 NLP techniques to measure and identify the semantic similarities of a text. The experimental results show that the accuracy of predicting impacted classes are increased more than 60 percent. High software quality is mandatory in an organization to avoid costly patching [16]. There is a need of a standardized quality check to maintain consistency and reliability among recent technologies [17]. It is a common dilemma that increasing demand of working software and integration factors led to forget about the quality of selected components. Table 1 illustrates the limitations of related work [3-8].

3. PROBLEM DEFINITION

According to the proposed extended CBD in [18], 'analysis, selection, and risk management' phase collects detailed specifications of the system to be developed.

Software industries face difficulties in selecting suitable components that will match the requirements of the new system. Many research studies suggest the need to find a suitable way for component selection.

How to propose a repository structure to facilitate easy identification, selection and integration of a reusable component?

4. THE PROPOSED SOLUTION

We propose a structure for software industries regardless of the methodology that will contribute to solving the problem of identification, selection, and acquiring a qualitative component from repositories as shown in figure 1.

A component possesses specific quality attribute like its domain, type of design pattern, and the intended users. To make our structure flexible and easily adaptable to organizations, we limited our structure to the following basic information that provides the ultimate goal needed.

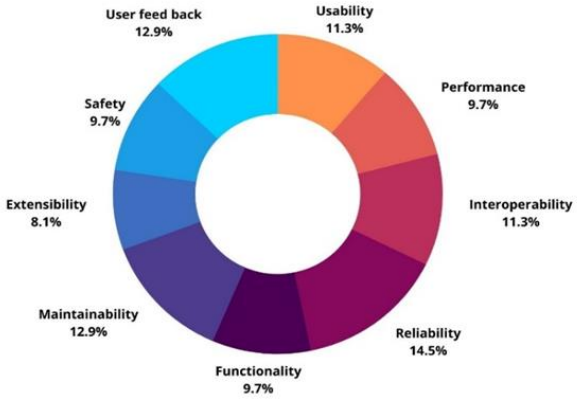

Figure 1. The Proposed Structure showing Quality attributes with team rating

Quality attributes of a component will serve two purposes:

- the maturity level;
- easy identification and retrieval from the repository.

A relative number will be assigned to each quality attribute by the development team during meetings from one to ten. Customer collaboration and feedback will help to rate a component. As a result, the proposed structure fulfills two functions related to quality:

- quality views based on the user;
- quality views based on the team.

A component always belongs to a specific design pattern and it is stored in the repository as per its type. It is extremely helpful for the development team to know that the selected component belongs to creational, structural or behavioral patterns. It will help to determine the architecture and degree of customization and integration risks. Estimate the domain of a component, the development team can reduce customization cost and time. A component can have different versions that can have different scores and ratings. The trade-off of the quality attribute will be visible by version control mechanism. It will help to determine, select and acquire the most feasible component and save time, cost and effort of a team as proposed in figure 2. According to figure 2, the proposed repository structure will save time, effort and integration cost during the component acquisition process.

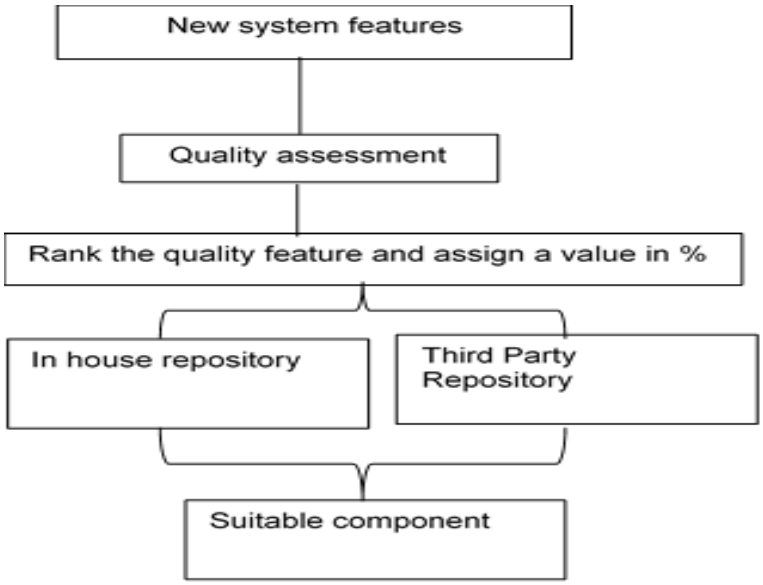

Figure 2. The Proposed Repository Structure to Perform Component Acquisition

5. VALIDATION

There are many techniques available to validate the proposed solution. We found that questionnaire technique is more suitable to validate our research proposal as per the time constraints. This paper is an outcome of one of master courses taught at King Abdul-Aziz University, KSA. The duration of the course is six months and one of the benefits of questionnaire technique is that it can examine the responses of a relatively large number of participants in less time. To validate our proposal, we targeted only those participants who are

practicing CBD through emails using embedded google sheets. The questionnaire is composed of twenty-one questions to validate the two goals of proposed solution.

Table 2. Likert scale used to evaluate the questionnaire

Strongly Agreed	5
Agreed	4
Neither Agreed Nor Disagreed	3
Disagreed	2
Strongly Disagreed	1

Goal 1 is to identify the organizational acceptability of the proposed repository structure and it is evaluated against eleven questions. The result of the acceptability will indicate the positive impact and usefulness of the structure because organizations (stakeholders, investors) cannot accept any structure without vividly seeing its positive results with respect to identify, select and acquire a component.

Goal 2 is to inquire that how much proposed repository structure facilitates to the development team and it is assessed using ten questions. Goal 2 will help the researchers to analyze the effectiveness of the proposed structure to adapt and integrate a reusable component.
The questions are evaluated using likert scale as shown in Table 2.

5.1. Goal 1: Determine the Organization Acceptability of the Proposed Structure

The questions in this goal represent the acceptability of the proposed repository structure to identify, select and retrieve the reusable components. Table 3 shows that 34% are strongly agreed whereas 28% are agreed. Furthermore, 21% are neither agreed nor disagreed. However, 12% are disagreed while other 5% are strongly disagreed. Figure 3 depicts the same result graphically.

Table 3. Analysis Result of Goal 1

Q. No.	**Strongly Disagreed**	**Disagreed**	**Neutral**	**Agreed**	**Strongly Agreed**
Q1.	0	1	3	7	11
Q2.	0	1	2	5	14
Q3.	2	7	7	3	3
Q4.	1	10	6	1	4
Q5	6	6	4	4	2
Q6	2	7	5	5	3
Q7	1	3	8	5	5
Q8	1	1	4	8	8
Q9	0	1	3	8	10
Q10	0	2	3	12	5
Q11	0	1	3	6	12
Total	13	30	48	64	77
Avg. in %	5	12	21	28	34

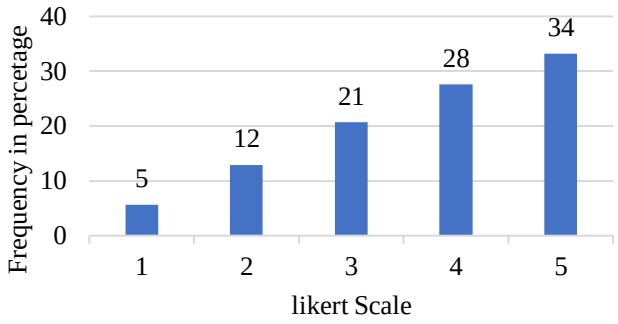

Figure 3. Cumulative Frequency Results of Goal 1

5.2. Goal 2: Adaption of Proposed Structure by Development Team

According to Table 4, the analysis of the results show that 35% are agreed and 31% are strongly agreed that the proposed structure facilitates to development teams to adapt and integrate the reusable components. Whereas 13% are disagreed and 1% are strongly disagree with the goal 2. Also, 20% are remained neutral. The results of Table 4 are presented graphically in figure 4.

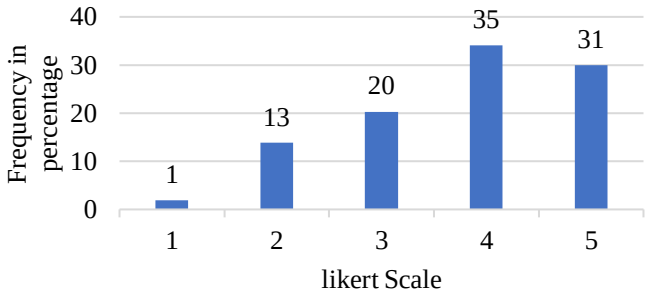

Figure 4. Cumulative Frequency Results of Goal 2

Table 4. Analysis result of goal 2

Q. No.	Strongly Disagreed	Disagreed	Neutral	Agreed	Strongly Agreed
Q1.	0	3	5	4	10
Q2.	0	3	4	9	6
Q3.	1	3	7	7	4
Q4.	1	4	8	4	6
Q5	1	4	3	8	6
Q6	0	2	7	9	4
Q7	1	1	1	10	9
Q8	0	3	3	7	9
Q9	0	3	4	9	6
Q10	0	4	6	7	5
Total	4	30	44	74	65
Avg. in %	1	13	20	35	31

5.3. Cumulative Analysis of Two Goals

Table 5 shows the cumulative analysis of two goals to evaluate the proposed research.

Table 5. Cumulative Analysis of Two Goals

Goal No.	Strongly Disagreed	Disagreed	Neutral	Agreed	Strongly Agreed
Goal 1	5	12	21	28	34
Goal 2	1	13	20	35	31
Total	6	25	41	63	65
Avg. in %	3	12	20	32	33

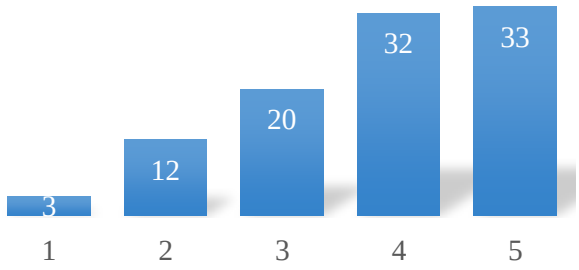

Figure 5. Cumulative Frequency Results of Two Goals

Figure 5 demonstrates the outcomes of cumulative statistical analysis of two goals. Thirty-two percentage of the participants are agreed with the proposed repository structure will help the software development teams during the selection and retrieval of suitable reusable components. Thirty-three percentage of the respondents are strongly agreed and twenty percentage of the respondents are neither agreed nor disagreed. Twelve percentage of the respondents are disagreed and only three percentages are strongly disagreed as shown in Table 5.

6. RECOMMENDATION AND FUTURE WORK

There are many version control libraries but those repositories do not cover all the necessary or required information to identify, select and integrate components using CBD. An improved repository structure is proposed to solve the problem in hand. BitBucket concentrates more on integrating and coordinating team while GitHub focuses more on component maturity and performance. GitLab provides hybrid features by concentering more on managing, organizing team, security and compliance but still it lacks in primary features mentioned in our proposed repository structure. The future work is to propose an extension of improve repository to evaluate the maturity level of each stored component. It will mitigate high cost of maintainability and decrease time to identify, select and retrieve a component.

7. CONCLUSION

To develop a high quality software, there is a need to accomplish a successful project coping the requirements of a customer in cost effective way and meeting the timelines. The problem is that customers are always pushing the software development companies to deliver products fast to take the competitive benefits. On the other hand, the software development companies want to reuse

the previously development components to avoid the scratch based development to meetneeds of demanding market. There is a need of software companies to store the reusable components in a repository. The repository contains several versions of each reusable component and it is extremely difficult for the software development team to select the most suitable version of a component for the project in hand to customize it. This research proposes a repository structure to facilitate the software development teams to choose the most appropriate component as per the requirements of a customer. The proposed repository structure will provide ample information about each version of reusable components so that the software development teams can easily analyze, select and retrieve reusable components. The core focus of this research is to propose such a repository structure that will save time, cost, efforts and resources of the software development companies by easing the components' identification, selection and retrieval processes. A questionnaire is used to validate the proposed repository structure and overall sixty-five percentage of the respondents are agreed with the effectiveness of the proposal.

REFERENCES

[1] Sommerville I., (2016) *Software Engineering*, Pearson Publisher.

[2] Maximus B. and Pressman R. S., (2019) *Software Engineering*, McGraw Hill Publisher.

[3] Chen, J., Yeap, W. K. and Bruda S. D. (2009) "A Review of Component Coupling Metrics for Component Based Development," WRI World Congress on Software Engineering, pp65-69.

[4] Li, J., Conradi, R., Slyngstad, O. P., Torchiano, M., Morisio, M. and Bunse, C. (2008) "A State-of-the-Practice Survey of Risk Management in Development with Off-the-Shelf Software Components," IEEE Trans. Softw. Eng., Vol. 34, pp271-286.

[5] Kumar, R. and Singh, H. (2012) "A model for quality assurance of OSS architecture," Sixth Int. Conf. Software Engineering (CONSEG), pp1-6.

[6] Kienle, H. M. (2007) "Building Reverse Engineering Tools with Software Components: Ten Lessons Learned," 14th Working Conference on Reverse Engineering (WCRE 2007), pp289-292.

[7] Palomba, F., Zanoni, M., Fontana, F. A., De Lucia, A. and Oliveto, R. (2019) "Toward a Smell-Aware Bug Prediction Model," IEEE Trans. Softw. Eng., Vol. 45, pp194-218.

[8] Petersen, K. et al. (2018) "Choosing Component Origins for Software Intensive Systems: In-House, COTS, OSS or Outsourcing?—A Case Survey," IEEE Trans. Soft. Eng., Vol. 44, pp237-261.

[9] Mead, N. R., Kohnke, A. and Shoemaker, D. (2020) "Secure Sourcing of COTS Products: A Critical Missing Element in Software Engineering Education," 32nd Conf. Software Engineering Education and Training (CSEE T), pp1-5.

[10] Liu, C. (2020) "Discovery and Quality Evaluation of Software Component Behavioral Models," IEEE Trans. Autom. Sci. Eng., Vol. 18, pp1538-1549.

[11] Mishra, A. and Otaiwi, Z. (2020) "DevOps and software quality: A systematic mapping," Comput. Sci. Rev., Vol. 38, pp100308.

[12] Al Dalla,l J. and Abdin, A. (2018) "Empirical Evaluation of the Impact of Object-Oriented Code Refactoring on Quality Attributes: A Systematic Literature Review," IEEE Trans. Soft. Eng., Vol. 44, pp44-69.

[13] Al-Tarawneh, F. H. and Althunibat, A. (2019) "Pilot Study of Healthcare COTS Software Evaluation and Selection," Int. Joint Conf. Electrical Engineering and Information Technology (JEEIT), pp311-314.

[14] Arvanitou, A., Ampatzoglou, A. C. and Carver, J. C. (2021) "Software engineering practices for scientific software development: A systematic mapping study," J. Syst. Soft., Vol. 172, pp110848.

[15] Falessi, D., Roll, J., Guo, J. L. C., and Cleland-Huang, J. (2020) "Leveraging Historical Associations between Requirements and Source Code to Identify Impacted Classes," IEEE Trans. Soft. Eng., Vol. 46, pp420-441.

[16] Poth, A., Meyer, B., Schlicht, P. and Riel. A. (2020) "Quality Assurance for Machine Learning – an approach to function and system safeguarding," 20th Int. Conf. Software Quality, Reliability and Security (QRS), pp22-29.

[17] Chakraborty, A., Bagavathi, R. and Tomer, U. (2020) "A Comprehensive Decomposition towards the Facets of Quality in IoT," Int. Conf. Smart Electronics and Communication (ICOSEC), pp759-764.

[18] Qureshi, M. R. J., Barnawi, A., and Talhi, A. A. (2013) "Component based development model for new and customized software products," 3rd World Conf. Information Technology,pp1384-1389.

AUTHORS

Muhammad Khamis Dauda is a certified Scrum Master and M.Sc. student at Faculty of Computing and Information Technology, King Abdul-Aziz University, Saudi Arabia.

Dr. Reda M Salama is working as an associate professor in King Abdul-Aziz University, Jeddah, Saudi Arabia.

Dr. Rizwan Qureshi is currently working as a full professor in the Department of IT, King Abdul-Aziz University, Jeddah, Saudi Arabia. This author is the best researcher awardees from the Department of Information Technology, King Abdul-Aziz University in 2013 and 2016.