FCE: A QUALITY METRIC FOR COTS BASED SOFTWARE DESIGN

M.V.VIJAYA SARADHI [1]
[1] Assoc.Prof, Dept. of CSE,
ASTRA, Hyderabad, India.

B.R.SASTRY [2]
[2] Director, ASTRA,
Bandlaguda, Hyderabad, India.

Abstract: **The software that is based on component is aimed at developing large software systems thorough combining the existing software components. Before integrate different components, first one need to identify whether functional and non functional properties of different components are feasible and required to be integrated to develop new system or software. Deriving a quality measure for reusable components has proven to be challenging task now a days. This paper proposes a quality metric that provides benefit at both project and process level, namely Fault Clearance Effectiveness (FCE). This paper identifies the different characteristics that component should have so that it can be used again and again. Component qualification is a system of finding out the fitness for use of existing components that will be used to develop a new system.**

Keywords: **Component; FCE Metric; Reusability**

I. INTRODUCTION

There are two phases in this system discovery and evaluation. In the first phase the properties are analyzed and relate to the functionality of the component and other qualities like adaptability, usability, portability, reliability and in the second phase components will be selected based its properties and their Fault clearance effectiveness.

High quality software is assured with component based approach and this approach was adopted by the total software industry. Due to budget and time limits, with the traditional approach it is not possible to deliver qualitative software. This position is considered as 'software crises' [1]. The reuse facility of components of existing software's in component based software development approach will save time and money. Component based approach manages complexities of the software especially those lead to software crises. In this approach even complex software need not be developed from basic, the existing software components can be used. Identifying the existing software components that are required to develop new software is enough. Identified components may be used directly without any modification or customize according to the requirements [3]. This will help to reduce the cost and time that will incur to develop and market the software and hence it results in increased productivity. As time is saved by using the existing components, more time can be devoted to enhance the quality of the software.

II. QUALITY OF COMPONENTS AND SUCCESS IN THEIR REUSABILITY

Usage of the existing components is dependent on the quality of the existing components and reuse will demand software testing and quality assurance. Especially while using these components to trade between organizations, software testing quality assurance should be given high priority. Quality of the component will decide the lifetime of the same. The components that have properties that can be reused will have long life the rest will die with in no time. More than any other software programs, components should undergo strenuous tests as there is possibility for errors in every application that uses the components. The developers' reputation will be at stake when the project has high reliability and availability requirements. A responsible developer can't blame any third party for the unreliable behavior of the project. All most all component explorers need to clarity about

- The potentiality of the component to provide needed functionality in the new application
- Its test performance
- Its implications on the new system regarding performance, reliability, robustness, maintainability, portability etc

As the component serves the needed behavior in a particular situation it is considered to be as a reusable component [1]. In order to reuse the component at high level one should consider varied situations to reuse the components. A software component may be victimized for more distinct applications, in diametric performing and study environments, by divergent developers using other methods and tools, for dissimilar users in divergent organizations.

III. QUALITY COMPONENT and its CHARACTERISTICS

Understandability, portability etc are the additional non-functional properties that a qualitative component provides apart from the functional properties. In terms of quality attributes only these properties are expressed. While developing system with available components, the important activity is characterization of component's qualities and their role in enclosing system [1]. The following are the qualities that a good quality component will have.

1) Comprehensibility - Based on the expected effort required to identify the concept behind a component and its applicability, the comprehensibility is defined [2]. Comprehensibility further can be attributed to documentation. Any component's documentation will explain how to use the component and how to configure the component. By providing the component's document the developer will communicate with the user and make it easy to the user to understand the utility and functionality of the component.

2) Compliance - The comfort with which component can be changed and utilized. The flexibility of a component to use it with other component by making required modifications in applications or environments apart from the intended use.

3) Communicability - Capacity to interact with the components of the software system in a reasonable manner or capacity to adapt the component with nominal modifications to accomplish the same goal. By analyzing the need for interface in-between different components the interdependency of the component can be estimated.

4) Portability - Component should fit on a varied range of computer platforms with miner modifications

5) Generality - The component should be a general one. Component should so flexible that it should have the entire feature that will enable the user to develop specific instances of the components to cater the application specific requirement. The level of the generality of the component will be decided by analyzing the efforts required to put the component operation like installation, n-installation and controlled features.

6) Dependability - Component should be trustworthy in all dimensions as it will be justifiably placed on the services it offers. Component will ensure the qualities like reliability, availability, safety, security, usability and expendability.

7) Consistent - Component should be consistent as far as its utility is concerned and should address a specified need. It may cater different needs across domains like word processor or particular to a domain like system for airline especially in emergency conditions. In general domain specific components cater some internal business need. But while developing the component it should not be developed as specific as it will not be feasible to reuse the component again and again. It should have some provision to reuse as per requirement.

8) Independent - component should not be dependent on other components to the possible extent. This independence will enhance the comfort in using that component and also easy to incorporate in different applications with nominal changes. Basically complex components are highly dependent on other components. It may not be possible to make any component totally independent but it is very much required to make components independent from the components that may change with in no time.

9) Transparent Interface - Every component will provide or require pre defined services from other components. Interface of a component should be a qualitative one as it plays major role in connecting independently developed component.

Other than these technical issues that a reusable component is expected to have there are some other non technical issues that should be considered like earlier business performance of the developer.

A few of non-technical issues are discussed here

a) Proper documentation: Component should have proper documentation about its utility, functionality, reliability, validity etc. Documentation will reflect and explain the quality of the component. In order to create the confidence among the user community about the component every developer need to provide a qualitative documentation about the component. There will be four categories in document of component they are General Information, Information in detail, Acceptance test and Support and additional information.

b) Conventionality of Standards: To develop a common component market that best approach is standardization. Standards are defined by a component model and conventions will be sent to developers. Amicability of the component model is one of the major features that out rates the other forms of packaged software. Standardizing is one of the useful technique in developing common interfaces and infrastructure that helps in minimizing the chances of mismatch that disturbs the productivity of the component.

c) Availability of related information. Different needs and testing activities will minimize the probable problems of component reuse and enhances that quality of the software. qualitative component has associated with different artifacts that are updated as and when the

component is altered. These artifacts consist of the specifications of the software, test suite, link of the software specifications with the test cases in the test suite. It will be easy to find out when some parts of the specification changes and test suet that requires up gradation and rerun again.

d) Assistance of the Vendor: A component will have long life only when it is promoted by a economically stable vendor. There will not be any insecurity about the existence of the vendor. In situations like, when the supplier is not ready to continue in the business process and he will provide the source code to the user as per the agreement.

e) Qualified: long term survival of the software is dependent on the availability of the qualitative components that create trust among the consumers. Consumers will have trust on those that are qualified by a third party. These third parties will check weather components are functioning in the manner as advertised by the developer and issues a certificate. So components that carry good certificates will have long life in the market.

As quality is crucial in all systems, its role is more in product lines. A substandard product with side effects will not only affects the business but also force to spend lot of amounts for different reasons when the component is used only in a single system. That's the reason the quality of the component is crucial for both developer and component supplier.

The services that a component should render will be defined by the architecture of the software where these components are going to be used. The specified standard requirements will guide the functional as well as non-functional requirements. Defining requirements of a component is an architect responsibility.

IV. MEASURING THE REUSABILITY EFFICIENCY

Calculating software longevity is a critical job but very much required to build healthy software development atmosphere. All most all software projects will exceed its scheduled time limits as well as budget, in spite of this it will have problems related to quality. The objective of software calculation is to measure the schedule; man hours required size of the product, stage of the product development and assured quality. Work status will be analyzed by comparing actual status with planned status of work. The information gathered will help to take the right decisions and at the time of planning for next projects [7]. For development activities quantitative basis will be provided by the software metrics. These metrics are useful in improving productivity and quality of the software [1, 2].

For calculating the reusability of the software Poulin [5] proposes two methods Qualitative Method and Empirical Method. In Empirical method objective data will be used. Simple analyses will calculate them easily with no expenditure, and this is one of the most important properties of a metric. Comparing the reusable component attributes with the attributes of the components that are not reused is common in empirical study. The qualities of the reusable software will have influence on the level of reusability of the software. In Qualitative study, it defines the software reusable qualities and analyzer subjectively evaluates how the software to be analyzed with respect to these qualities. Qualitative methods are expensive in comparison with empirical methods due to the use of analysis.

NASA projects were studied by the Selby [6] and software reuse was successful. He identified the factors that made software reuse successful. Primary factor is, reusable software module's interface is simple and it is small in size. It is almost independent from other components. It is supported with detailed documentation. The reuse of utility functions of the component along with low level system is more often than reuse of human interface function. Selby used statistical methods to validate the results.

Four reusability factors were given by the ESPRIT-2 project REBOOT [5] (reuse based on object oriented technique). These are specified using a multiple number of criteria. At least one metric will be with each criterion. Reusability is a value varying in-between 0 to 1 and this is calculated by normalizing the metrics. The four factors of reusability are understandability, flexibility, probability and confidence of the reuses. The comfort of using the component in other environment is expressed in probability. Flexibility will define the generality and modularity of the component. Understandability comprises of complexity of the code, documented information and self explanatory and component complexity. Confidence is the knowledge of the reuses about the component and its utility and functionality.

It is recommended by IBM methods [5] that, along with code, development project's documentation should be accessible to the developer. Fonash [4] categorized the software component reuse metrics into five types as Cohesion, Parameterization, Coupling, General and Quality. Cohesion kind will calculate the functional cohesion along with data cohesion. Functional cohesion metrics validates the each and every part with respect to its requirement in performing a single function where as the data cohesion metrics analyses the level to which the module has a single data type associated with it. Parameterization kind will calculate the number of functional or data parameters in a module. Level of independence will be calculated by coupling kind of metrics. Size, type and understandability will be calculated by the general type of metrics. Flexibility,

formation of the code, comments etc will be calculated by the quality kind of metrics.

To calculate the reusability of the component Hironari Washizaki et al [2] have proposed five metrics. The proposed five metrics are

1) Existence of Meta Information
2) Half or one third of the component characteristics should be readable,
3) One third of the components characteristics must be writable.
4) Three fourth of the component's business methods should not have return value
5) Irrespective of method parameters presence components'' quality should be assured.

In every study the methods used to evaluate the quality of the component are almost same. Understandability, portability, flexibility, proper documentation, independence, confidence of the consumer is some of the characteristics a component should have to reuse it effectively. These characteristics' should be formalize and quantify by applying empirical method.

The above hence concluding that the metrics that discussed are measuring the reusability levels of a component based on their characteristics, metadata and the behavior of their business methods. There is lack of a quality metric that measure the reusability of a component based on their earlier usage. The following section defines a new quality metric for COTs based system.

V. FAULT CLEARANCE EFFECTIVENESS: A QUALITY METRIC IN THE REUSE CONTEXT

Fault clearance effectiveness is a metric proposed to measure the effectiveness in fault clearance at each life cycle of software that build by different cots integration. FCE helps to find the desired improvements to be done. Fault classification and measuring the volume of the faults will show their impact on the materialization of the root cause analysis efforts.

The log of faults can be used to find the total faults inserted and detect on each COT integration. This process will be at the early stages of project planning. The log even helps to find the fault accumulation rate.

The FCE in COST based software development can be formulated as

$$\mathbf{FCE=F_{ci}/(F_{ci}+F_{ci+1})}$$

F_{ci} represents faults count for i^{th} component "ci"

F_{ci+1} represent fault count detect while integrating components ci and ci+1

In this context FCE value 1 indicates efficacy of the product in terms of quality and efficiency. This concludes that fault detection rate at each component level defines the FCE

Input for FCE computation process

No of faults found in each component. This fault cont will be taken in the sequence of component integration.

Set of components c_1, c2, c_3....., c_n.

And Number of faults detected in each component is F_{c1}, $F_{c2,}$ F_{c3}, F_{c4} ..... , F_{cn}

Find FCE of i^{th} component FCE_i as fallow

$$\mathbf{FCE_i=F_{ci}/(F_{ci}+F_{ci+1})}$$

$$\mathbf{FCE=\sum FCE_i/n \text{ for } i=1\ldots n;}$$

A. Planning a fault Profile

The steps involved in each stage of the component based software development system are f0llowing

- Number of Faults 'n_f' to be inserted should be estimated.
- Clearance effectiveness c_y should be measured in percentage
- Approximate count of faults (n_f*c_y) that can be cleared
- Count the faults that are not cleared n_f-n_f*c_y
- Calculate number remaining (n_f-c_y*n_f)
- Add to estimate of the number likely to be added in next stage of component integration
- Find cumulative clearance effectiveness in percentage

		Search	trace	Build	Test	Integrate
1st phase	**Fault Insertion rate**	30	11	60	5	2
	Fault Clearance Effectiveness	70 %	65 %	60 %	57 %	47 %
	Cumulative FCE	70 %	83 %	76 %	88 %	93 %
2nd Phase	**Fault Insertion rate**	20	11	60	2	1
	Fault Clearance Effectiveness	70 %	65 %	65 %	60 %	50 %
	Cumulative FCE	70 %	80 %	78 %	91 %	95 %
3rd Phase	**Fault Insertion rate**	10	11	60	2	1
	Fault Clearance Effectiveness	70 %	65 %	68 %	63 %	52 %
	Cumulative FEC	70 %	76 %	80 %	94 %	97 %

Table 1 fault insertion rate and Clearance Effectiveness

The Table 1 shows the fault insertion rate, Clearance Effectiveness and cumulative effectiveness of FCE in stages like search, trace, build, test and integrate of the COTs based software development. The table 1 is sampling the details only for three phases of COTs integration. In production level model the collection of fault insertion rate, Clearance Effectiveness and cumulative effectiveness of FCE continues for all phases. By utilizing these details a fault reporting process occurs. This fault reporting process performed in the following steps.

B. FAULT LOGGING

Log of the faults (Where found, date found, type, stage injected, stage removed, consequences of removal, time to repair, etc) Fault report forms (Location, severity, inspection rates, yields, etc.)

VI. CONCLUSION

If an organization is considering existing software to reuse, it should analyze several issues before it decides that the existing software is qualified for reuse. Quality of the software is the foremost issue to be checked before considering software for reuse. Therefore quality of the components should be evaluated by the software developing organizations. The software development organization that are developing software that can be reused should evaluate whether these software cater the reusability criteria or not. Further the criteria of selecting a COT for reuse leads to success in cots based development, maintenance and extension of the systems.. In the COTs reuse orientation, the component quality will be considered more. The reusability frequency of a component is proportional to complexity of the component development process. Hence to ensure the quality of the developed COTs based product, more support required.

REFERENCES

[1] Software process and product improvement: an empirical assessment, **J. P. Kuilboer and N. Ashrafi,** University of Massachusetts, Management Science/Information Systems, 100 Morrissey Blvd., Boston, MA 02125, USA

[2] "A Metrics suite for measuring Reusability of software Components" - Hironari Washizaki et al, 9th IEEE International Symposium on Software Metrics, 2003.

[3] "Component Software: Beyond Object-Oriented Programming" - C. Szyperski, Addison Wesley, 1999.

[4] "Characteristics of reusable software code components" - Fonash P, Ph.D. Dissertation, George Mason University, 1993.

[5] "Measuring software reuse, principles, Practices and Economic Models" - Poulin J. S, Addison Wesley Publishing, 1997.

[6] "Quantitative studies of software reuse", Software reusability - Selby, R.W, vol 2, ed. Biggerstaff, T.J and Perlis, A.J., Addison Wesley, 1989

[7] "Measuring and Improving Component Based software Development" - Pentti Virtanen, Ph. D. thesis , department of Computer Science, university of Turku , Finland.

AUTHORS PROFILE

M.V.Vijaya Saradhi is Currently Associated Professor in the Department of Computer Science and Engineering (CSE) at Aurora's Scientific, Technological and Research Academy, (ASTRA), Bandlaguda, Hyderabad, India, where he teaches Several Courses in the area of Computer Science. He is Currently Pursuing the PhD degree in Computer Science at Osmania University, Faculty of Engineering, Hyderabad, India.. He is a life member of various professional bodies like MIETE, MCSI, MIE, MISTE.

Dr. B. R. Sastry is currently working as Director, Astra, Hyderabad, India. He earlier worked for 12 years in Industry that developed indigenous computer systems in India. His areas of research includes Computer Architecture, Network Security, Software Engineering, Data Mining and Natural Language Processing, He is currently concentrating on improving academic standards and imparting quality engineering