A Metric Based Approach to Extract, Store and Deploy Software Reusable Components Effectively

Muhammad Ilyas[1], Mubashir Abbas[2], Khansa Saleem[3]

[1]Department of Computer Science and Information Technology, University of Sargodha Sargodha, Punjab, Pakistan

[2]Department of Computer Science and Information Technology, University of Sargodha Sargodha, Punjab, Pakistan

[3]Department of Computer Science and Information Technology, University of Sargodha Sargodha, Punjab, Pakistan

Abstract

Software reusability is a valuable methodology for quality, economical and timely software development. The effective use of reusability gives benefits in the form of less time, efforts and cost for quality software development. Reusability also helps to diminish the risk associated with software development and success. Due to inevitable payback, reusability has grown up to be most accepted practice for software development. But reusability handling methods and techniques are not well organized, so need is to formalize the reusability process in order to get its actual benefits in form of time, cost and effort savings. Formal and structured approach is required in reusability practices because reusability observation, extraction, classification and deployment methods are not disciplined. In this paper a framework is suggested to make reusability process formal and organized. In this approach quality earning criteria are defined at each level of reusability process to observe the need of reusability, extracting reusable components, classifying them and then integrating with new systems efficiently. The aim is to formalize each activity of reusability process to get satisfactory and quality results.

KEYWORDS: *Software Component Assessment, Software Reusability, Software Risk, Reusability Process.*

1. Introduction

Reusability methodology has turn into a productive tool for software development as it reduces time, cost and work required for the software development [1, 2, 3].Reusability increases reliability, quality and productivity of software products by using already existing tested components [1, 6 , 5]. Developers consider it most favorable choice for economical development of business and technical projects [1].Reusability ensures within time delivery of software products and minimizes risk involved with its success [2, 4]. Reusability boosts up confidence of developing team as they have previously groundwork in the current domain [2]. Reusability gives benefits in the form of portability, maintainability and productivity improvements [1, 4]. Due to inevitable paybacks reusability is widely used for cheaper and timely software development purposes. But techniques for its observability, suitability, selection and deployments are not well organized, so need is to apply structured and formal approach in the implementation of reusability to acquire real benefits.

This study aims to formalize the all activities performed during reusability life cycle [1].It will help to obtain satisfactory and quality results. For this purpose the reusability life cycle is divided into three stages: Reusable Component's Extraction, Reusable Component's Storage and Reusable Component's Deployment. Certain metrics are defined at each stage aiming quality and productive output from each stage. Meticulous meditation is given to the extraction of quality software components that can be reused for productive software development purpose. Than these reusable components are stored and classified in the reuse repository on the basis of certain characteristics. These characteristics make searching and retrieval process efficient [7]. At the end Pre Adopt test is conducted to find the most suitable and trusted reusable components according to the new system requirements [3].

The next section constitutes related work followed by a proposed process and conclusion and future work.

2. Related Work

Al-badareen [1] proposed framework consists of extraction, adaptation, storage, pre–store and pre-use process. The extraction of reusable components is performed during develop for-use process, which is focused on that how to extract suitable information for a reusable component. The storage process is performed to store extracted reusable components in the library, so they can be accessed easily for reusable purposes. The adaptation process is how to pick a suitable component according to new system requirements. Pre-store is the process to evaluate and enhance reusable components according to certain standards to satisfy library requirements. Pre-store

process evaluates reusable components on the basis of co-existence, compliance, generality and adaptability characteristics. The pre-use process is the process to evaluate and modify the reusable components, retrieved from the library to satisfy the new system requirements. Three main characteristics: suitability, accuracy and compliance are considered and evaluated in pre-use process. Both pre-store and pre-use processes include a reusability test. Reusability test checks for certain library conditions to be fulfilled by the reusable component during pre-store process. In pre-use process, reusability test evaluates the reusable components according to certain requirements of new system.

G. Sindre [7] has introduced reboot approach to deal with reusability issues. Reboot qualification model is used to ensure existence of quality reusable components in the library. Portability, flexibility, understandability and confidence metrics are used for evaluation of reusable components. It ensures that quality components have qualified for storage in the library. With large libraries having large numbers of reusable components of different domains, a proper way is required for efficient retrieval of these reusable components. For this purpose reboot used a faceted classification scheme to classify these reusable components in the library. He made use of four facets abstraction, operations, operates on and dependencies for the classification of reusable components. Also some other attributes like who developed it, when it was developed and how big it is, are considered for storing and classifying the reusable components in the library.

Fazl-e-amin [6] developed the reusability attribute model and metrics to measure the attributes of reusable components. This model defines those attributes of an aspect oriented component that contributes to its reusability. Flexibility, maintainability, portability, scope coverage, understandability and variability were identified as the attributes of the given model.

3. Proposed Framework

Although Reusability has grown up into a fruitful practice for software development but its implementation techniques are not systematic and formal. Formal approach is pretty required in reusability observation, extraction, classification and deployment processes. To overcome limitations in usage of reusability, a frame work is purposed in this paper. Aim is to formalize the reusability process at each step that will help to gain quality results from each stage of reusability life cycle [1].

According to this purposed approach reusability life cycle is divided in to three stages Reusable Component's Extraction, Reusable Component's Storage and Reusable Component's Deployment shown in fig. 1.

At each stage certain metrics are defined, which component has to be fulfilled to move on to the next stage during reusability life cycle. At Extraction stage components that can be reused are extracted. These reusable components must be capable of some particular characteristics to become an effective reusable component [1]. When a reusable component is selected then it needs to be store into the reusable repository from where it can be retrieved for reuse purpose [1, 7]. So searching and retrieval of reusable components can be easily made if components are properly stored. Third stage of this framework is deployment process in which a component of similar functionality is fetched from the Reuse Repository. Now this fetched component is assessed for suitability with new system's requirements. If it satisfies the new system requirements then it is adopted and put for Pre Adopt test for final selection [1]. After using metrics at each stage quality results are anticipated aiming to get true benefits of reusability in the form of less time, cost and work for the quality and productive software development.

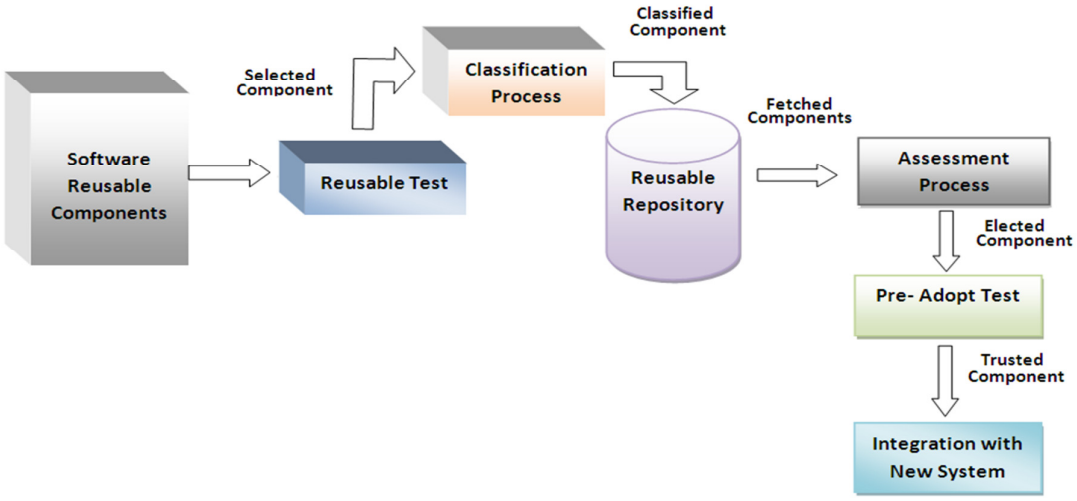

Fig.1 RESAD Framework

4. Extraction of Reusable Components

Reusability has become an ideal methodology for software development as it helps to develop quality software components with minimum cost and effort. It is possible when there is some repository of reliable, portable and qualitative components that can be reused into a new system. In this way time, cost and effort is decreasing and productivity of the new system is increasing [1, 3, 5, and 6]. So there must be a formal process to extract the components that can be reused. Extraction of reusable components is the process of finding components that can be reused. This selection is made on the basis of some particular characteristics shown in fig.2. These characteristics are defined to produce healthy reusable components. In this study following metrics are defined that a software component must fulfill in order to be select as a reusable component.

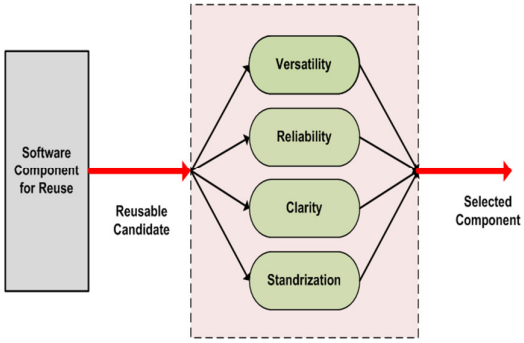

Fig. 2 Extraction Process

4.1. Versatility

Versatility characteristic of software component is the ability to deal with different type of platforms efficiently. It means that entire software component can perform its functionality independent of hardware and software constraints [1, 7]. Component is providing support to wide range of machines on which different operating systems are running.

Versatility of software component also defines that it contains support or compatible with different programming languages. Component with versatility feature is considered more flexible and portable, has the great chance to be reused [7].

Here Versatility Metric is being used as combination of Generality and Portability Metric of software components. To determine versatility of software components both generality and portability metrics will be analyzed. To measure versatility characteristic following metrics are proposed:

4.1.1 Generality Metric

According to W. J. Salamon generality of software can be measured by following metrics [8].

- Multiple Usage Metric

A module is more general if it is referenced by more than one module. Higher the value of entire metric denotes maximum generality of software component.

$$\frac{\text{Total no. of modules refferenced by more than one module}}{\text{Total no. of modules}}$$

This relation can be denoted mathematically as follows:

$$M_{US} = \frac{\sum_i^n Ref_i}{n}$$

$$\therefore Ref_i = \begin{cases} 1, when\ function\ is\ reffernced\ by\ more\ than\ one\ module \\ 0, otherwise \end{cases}$$

Where i=1, 2 …n & n= Total no. of modules (1)

- Mixed Function Metric

A function is considered to be Mixed Function if it performs more than one task like I/O as well as processing operation. Higher value of this metric provides maximum generality.

$$\frac{\text{Total no. of modules with mixed function property}}{\text{Total no. of modules}}$$

Mathematical form of this relation is:

$$M_{FN} = \frac{\sum_i^n Mix_i}{n}$$

$$\therefore Mix_i = \begin{cases} 1, when\ function\ performs\ more\ than\ one\ task \\ 0, otherwise \end{cases}$$

Where i=1, 2 …n & n= Total no. of modules (2)

- Data Volume Metric

A module that can process unlimited range of inputs is considered more general.

$$\frac{\text{Total no. of modules with data volume property}}{\text{Total no. of modules}}$$

Mathematically it can be represented as:

$$D_{VM} = \frac{\sum_i^n Vm\_Limit_i}{n}$$

$$\therefore Vm\_Limit_i = \begin{cases} 1, when\ function\ is\ data\ volume\ limited \\ 0, otherwise \end{cases}$$

Where i=1, 2 …n & n= Total no. of modules (3)

- Data Value Metric

A module is more general if it can process long range of data items.

$$\frac{\text{Total no. of modules with data value property}}{\text{Total no. of modules}}$$

Mathematical representation of this relation is:

$$D_{VL} = \frac{\sum_i^n Vl\_Limit_i}{n}$$

$$\therefore Vl\_Limit_i = \begin{cases} 1, when\ function\ is\ data\ value\ limited \\ 0, otherwise \end{cases}$$
Where i=1, 2 …n & n= Total no. of modules

(4)

- Redefinition of Constants Metric

To change the function of modules constants should not be redefined.

$$\frac{\text{No. of constants that are redefined}}{\text{Total no. of constants}}$$

Mathematical form of this relation is:

$$C_{RF} = \frac{\sum_i^C Rdf_i}{C}$$

$$\therefore Rdf_i = \begin{cases} 1, when\ constant\ is\ redefined \\ 0, otherwise \end{cases}$$
Where i=1, 2……. c & c= no. of constants

(5)

K.K Aggarwal presented the idea of General Programming using templates. According to him templates are more generic and can be used with different data types [9].
Templates can be classified in to two types Function Templates and Class Templates.

- Function Template Metric

To make behaviour of a function general, to be operate on each given data type function templates are used.

$$\frac{\text{No. of fuctions using function template}}{\text{Total no. of functions}}$$

Mathematically it can be represented as:

$$F_{TM} = \frac{\sum_i^n UFt_i}{n}$$

$$\therefore UFt_i = \begin{cases} 1, when\ function\ is\ using\ function\ template \\ 0, otherwise \end{cases}$$
Where i=1, 2…… n & n= Total no. of functions

(6)

- Class Template Metric

To make behaviour of a class general, to accept objects of particular data type class template are used.

$$\frac{\text{No. of classes using class template}}{\text{Total no. of classes}}$$

Mathematical form of this relation is:

$$C_{TM} = \frac{\sum_i^C UCt_i}{n}$$

$$\therefore UCt_i = \begin{cases} 1, when\ class\ is\ using\ class\ template \\ 0, otherwise \end{cases}$$
Where i=1, 2……. C & C= no. of Classes

(7)

Probable effect of all above described metrics about generality of software component is mentioned in following table.

Table 1: Effect of Generality Metrics

Metric Name	*Metric Symbol*	*Value*	*Reusability Probability*
Multiple Usage	M_{US}	High	High
Mixed Function	M_{FN}	Low	High
Data Volume	D_{VM}	Low	High
Data Value	D_{VL}	Low	High
Redefinition of Constants	C_{RF}	Low	High
Function Template	F_{TM}	High	High
Class Template	C_{TM}	High	High

4.1.2. Portability Metrics

W. J. Salamon made use of following metrics to measure portability characteristic of software components [8].

Software Independence Measurement

- Compatibility Metric

No. of operating systems with software is compatible

$$C_{MP} = \frac{\sum_i^n Os_i}{n}$$

$$\therefore Os_i = \begin{cases} 1, when\ operating\ system\ is\ compatible \\ 0, otherwise \end{cases}$$
Where i=1, 2……..n & n= no. of operating systems

(8)

- System Utilities Metric

Total no. of system utility utilized. (To measure S/W dependability)

$$S_{UY} = \frac{\sum_{i=0}^n Uty_i}{n}$$

$$\therefore \text{Uty}_i = \begin{cases} 1, when\ system\ utility\ is\ being\ utilized \\ 0, otherwise \end{cases}$$

Where i=1, 2……..n & n= no. of operating system utilities

(9)

- Standard Constructs Metric

Checks for Common, standard subsets of language used:

$$\frac{\text{Total no. of modules using non standard constructs}}{\text{Total no. of modules}}$$

Mathematical form of this relation is:

$$S_{CN} = \frac{\sum_i^n USing\ \_Nstd_i}{n}$$

$$\therefore Using\_Nstd_i = \begin{cases} 1, when\ module\ is\ using\ non\ \ standard\ constructs \\ 0, otherwise \end{cases}$$
Where i=1, 2…… n & n= Total no. of modules

(10)

Hardware Independence Measurement

- Open system Metric

Are the programming languages and tools (e.g., compilers, DBMS, and user interface) available on other machines? A value of 1 means yes, 0 means No. Mathematically this property can be represented as:

$$O_{SY} = \begin{cases} 1, when\ progrmming\ languages\ and\ tools\ are\ available \\ 0, otherwise \end{cases} \quad (11)$$

- I/O References Metric

$$\frac{\text{Total no. of modules making IO references}}{\text{Total no. of modules}}$$

$$R_{IO} = \frac{\sum_i^n Ref_i}{n}$$

$$\therefore Ref_i = \begin{cases} 1, when\ \ module\ is\ making\ I/O\ referrence \\ 0, otherwise \end{cases}$$

Where i=1, 2..... n & n= Total no. of modules (12)

- Word/Character Size Metric

$$\frac{\text{Total no. of modules not following convention}}{\text{Total no. of modules}}$$

$$W_{SZ} = \frac{\sum_i^n NT\_Conv_i}{n}$$

$$\therefore NT\_Conv_i = \begin{cases} 1, when\ module\ is\ not\ follwoing\ convention \\ 0, otherwise \end{cases}$$

Where i=1, 2..... n & n= Total no. of modules (13)

Anticipated outcome of above metrics measuring Generality of software component is mentioned below:

Table 2: Effect of Portability Metrics

Metric Name	*Metric Symbol*	*Value*	*Reusability Probability*
Compatibility	C_{MP}	High	High
System Utilities	U_{SY}	Low	High
Standard Construct	S_{CN}	Low	High
Open System	O_{SY}	Yes	High
I/O Reference	R_{IO}	Low	High
Word/ Character Size	W_{SZ}	High	High

4.2 Clarity

Clarity of the software components is the characteristic which states that component is clear and well understood in its vision, scope and functionality [6, 7]. A component is understandable if it is readable. Then it supports its understandability which enhances analyzability and changeability of software component. Bajeh has suggested following metric to measure readability of software component which is directly connected with it clarity and understandability [10].

- Indentation Metric

$$\frac{\text{No. of lines of code properly indented}}{\text{No. of lines of code expected to be indented}}$$

Mathematical form of this relation is:

$$N_{LI} = \frac{\sum_i^L Pid_i}{\sum_i^L Eid_i}$$

$$\therefore Pid_i = \begin{cases} 1, when\ line\ of\ code\ is\ properly\ indented \\ 0, otherwise \end{cases}$$

$$\therefore Eid_i = \begin{cases} 1, when\ line\ of\ code\ is\ expected\ to\ be\ indented \\ 0, otherwise \end{cases}$$

Where i=1, 2..... L & L= Total no. of lines of codes (14)

- Comments Metric

$$\frac{\text{No. of lines of code commented}}{\text{Total no. of lines of code}}$$

Mathematically it can be represented as:

$$N_{LC} = \frac{\sum_i^L Cmt_i}{L}$$

$$\therefore Cmt_i = \begin{cases} 1, when\ line\ of\ code\ is\ commented \\ 0, otherwise \end{cases}$$

Where i=1, 2..... L & L= Total no. of lines of code (15)

Anticipated effect of clarity metrics is described in following table.

Table 3: Effect of Clarity Metrics

Metric Name	*Metric Symbol*	*Value*	*Reusability Probability*
Indentation Metric	N_{LI}	High	High
Comments Metric	N_{LC}	High	High

4.3 Reliability

Reliability characteristic of software component states that entire component is a confident component. It has performed its functionality satisfactory without any failure in different environments at different times [1, 7]. Reliability of software components can also be defined as the entire component is capable of excellent quality results in various circumstances without any exception that is associated with quality and performance decreasing.
Mathematically it can be represented as:

$$\frac{\text{No. of time used}}{\text{No. of time failure is reported}}$$

$$R_{LY} = \frac{\sum_i^T Flr_i}{T}$$

$$\therefore Flr_i = \begin{cases} 1, when\ failure\ is\ reported \\ 0, otherwise \end{cases}$$

Where i=1, 2..... T & T= Total no. of time used (16)

4.4 Standardization

Standardization characteristic of software component declares that component is developed in view of standard software engineering rules and practices [1]. These rules are necessary to be followed for standard and quality software products developments.
In other words it is the certificate held by the software products that they can be reused for satisfactory results.
Mathematical representation of this metric is as follows:

$$S_{TD} = \begin{cases} 1\,, when\ component\ is\ following\ software\ standards \\ 0\,, otherwise \end{cases} \quad (17)$$

5. Storage and Classification of Reusable Components

During extraction process, candidates that are selected as reusable components needs to be store somewhere from which they can be retrieved and accessed easily [1]. So there must be a proper way to store and classify the software components. Now it will be easy to access and retrieve them with minimum effort and time that are the focal benefits of the reusability methodology [7].
In this purposed approach components that are selected as reusable components are classified on the basis of some features shown in fig. 3.

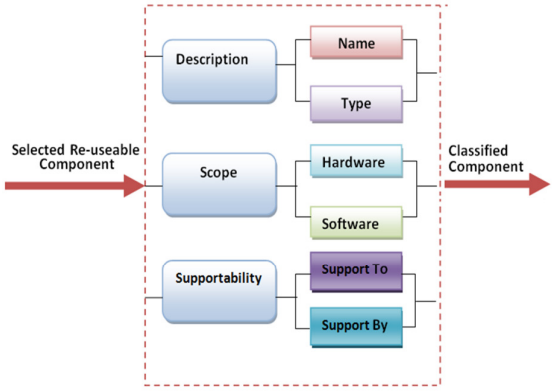

Fig. 3 Storage and Classification Process

5.1 Descriptions

Description attribute of classification process contains information about name and type of reusable component.

- *Name* attribute specifies the name of reusable component.
- *Type* attribute defines whether reusable components is requirement document, design template , source code, test case or else one belongs to SDLC phase[7].

5.2 Scope

Scope attribute of classification process provides details about range of hardware and software.

- *Hardware* attribute specifies the series of machines where this component can be run to perform the prescribed task.
- *Software* attribute specifies the range of operating systems that provide support to entire component to perform its functionality.

5.3 Supportability

Supportability attribute of classification process contains two attributes, Support To and Support By.

- *Support To* attribute defines the software category to which it belongs. Software categories can be system software, application software, embedded software etc.
- *Support By* attribute defines the set of compatible programming languages.

6. Assessment Process

Assessment process is the activity during reusability life cycle in which reusable components are evaluated and analyzed according to the new system requirements [1, 7].
The aim is to find the most suitable component for the given requirement. There is possibility that more than one reusable component can be in the repository for the given requirements. In this situation the task is to select the best component that is most suitable to the given requirements [7].

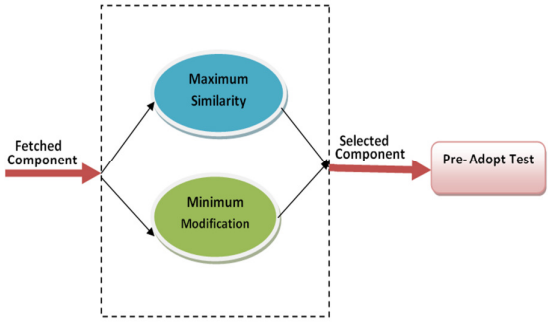

Fig.4 Assessment Process

In this purposed approach this task is performed by Assessment Process. It selects most suitable component from the repository for the given requirements. Assessment is performed on the basis of metrics described in fig. 4:

6.1 Maximum Similarities

The reusable component, having maximum similarity with given requirements is fetched from the repository. The selected component can be the most suitable

choice to carry out the given task efficiently due to the maximum similarities with given requirement. Mathematical form of this relation is as follows:

$$\frac{\text{No. of features available}}{\text{No. of features required}}$$

$$M_{SM} = \frac{\sum_i^F Ftr_i}{F}$$

$$\therefore Ftr_i = \begin{cases} 1, when\ required\ feature\ is\ available \\ 0, otherwise \end{cases}$$

Where i=1, 2..... F & F = Total no. of features

(18)

6.2 Minimum Modifications

Assessment Process fetches reusable component from repository which is requiring minimum modifications to meet new system requirements. The selected component with minimum modifications is demanding less efforts, time and cost to perform required functionality.

Minimum Modification metric is compliment of Maximum Similarities metric. Mathematically this relation can be shown as:

$$M_{DN} = \sim M_{SM}$$

(19)

7. Adaptation Process

The assessment process provides components having maximum similarities and minimum modifications to the given requirements. Now this component is most suitable for integrating with new system. But still this component has to cross Pre Adopt test. Pre Adopt test consists of characteristics specifically belongs to new system requirements. These characteristics evaluate that whether the selected component is suitable or not to use in the new system [1].

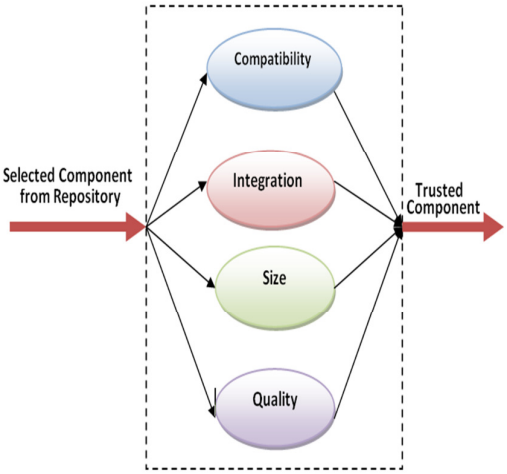

Fig. 5 Pre Adopt Test

7.1 Compatibility

Compatibility metric of Pre Adopt Test evaluates that selected component is compatible with new system. It is and easier job to adjust it into the new system and it can perform required functionality without any complication.

$$C_{MP} = \begin{cases} 1, when\ component\ is\ compatible\ with\ new\ system \\ 0, otherwise \end{cases}$$

(20)

7.2 Integration

Integration metric of Pre Adopt test analysis that selected component from the reusable repository is easy to integrate with new system. Proper interfaces are available for integrating with other systems. This attribute focuses that overall reusing cost and effort is less than development cost and effort of newly developed component.

$$I_{NT} = \begin{cases} 1, when\ \ interface\ for\ integration\ is\ available \\ 0, otherwise \end{cases}$$

(21)

7.3 Size

Size metric of Pre Adopt test observes that the size of selected reusable component. It will observe that whole component or some part of it is needed to fulfill given requirement.

$$S_{ZE} = \begin{cases} 1, whem\ full\ component\ is\ required \\ 0, when\ part\ of\ component\ is\ required \end{cases}$$

(22)

7.4 Quality

Quality metric of the Pre Adopt Test determines the quality of selected component. Quality metric depends upon Maximum Similarities Metric, Compatibility Metric and Integration Metric. Positive values of these metrics ensure that selected component is economical and trusted component for subsequent usage. Mathematical relation between Quality and above mentioned metrics can be shown as follows:

$$Q_{TY} = M_{SM} \wedge C_{MP} \wedge I_{NT}$$

(23)

8. Conclusions and Future Work

Reusability methodology has grown up into a productive tool for economical, quality and timely software development. Effective use of reusability improves quality, productivity and maintainability of the software products. But there is need of formal and systematic approach in the use of reusability methodology. This formalism is aiming earning of

actual benefits of reusability in the form of less time, cost and effort.

In this context, a framework is purposed to adopt formalism in the reusable component life cycle. This framework consists of three stages: Extraction, Storage and Deployment of reusable components. Extraction stage extracts the quality based reusable components during software development life cycle. Now this reusable component can be reused for similar kind of problem. Storage Stage classifies and stores these reusable components. After this they can be accessed and retrieved easily. Assessment Process evaluates the reusable component according to new system requirements and finds the most suitable component. In deployment stage Pre Adopt test is conducted for the final selection. After this selected reusable component is integrated with new system. All these tasks are performed on the basis of particular metrics at each level of reusability life cycle.

Our future work is aimed at validation of metrics used in the purposed framework to measure these attributes in different conditions. Prototype of this framework is also under construction, which is the foremost goal of future work.

9. References

[1] A. B. AL-Badareen, M. H. Selamat and M. A. Jabar "Reusable Software Component Life Cycle" International Journal of Computers, Vol.5, Issue 2 (2011), pp. 191-199.

[2] G. Singaravel, V. Palanisamy and A. Krishnan "Overview Analysis of Reusability Metrics in Software Development for Risk Reduction" International Conference on Innovative Computing Technologies (ICICTI), February, 12-13 (2010), Tamil Nadu, India.

[3] R. Kamalraj, B. G. Geetha, and G.Singaravel "Reducing Efforts on Software Project Management using Software Package Reusability" IEEE International, Advance Computing Conference, March, 6-7 (2009), pp.1624-1627, Patiala, India,

[4] P. S. Sandhu, Aashima, and P.Kakkar "A Survey on Software Reusability"2nd International Conference on Mechanical and Electrical Technology (ICMET), September, 10-12 (2010)

[5] S. Singh, M. Thapa, and S. Singh, et al. "Software Engineering, Survey of Reusability Based on Software Component" International Journal of Computer Applications (0975 – 8887) Vol. 8– No.12, October (2010), pp.39-42.

[6] F. Amin, A. K. Mahmood, and A. Oxley "A Proposed Reusability Attribute Model for Aspect Oriented Software Product Line Components" Information Technology (ITSim), International Symposium, June, (2010).

[7] G. Sindre E, A. Karlsson, and T. Staalhane "A Method for Software Reuse through Large Component Libraries" Fifth International Conference on Computing and Information. Proceedings ICCI 'May (1993).

[8] W. J. Salamon and D. R. Wallace "Quality Characteristics and Metrics for Reusable Software" Preliminary Report, U.S. Department of Commerce, (1994).

[9] K. K. Aggarwal, Y. Singh and A. Kaur, et al. "Software Reuse Metrics for Object-Oriented Systems" Proceedings of the 2005 Third ACIS Int'l Conference on Software Engineering Research, Management and Applications (SERA'05), (2005).

[10] A. O. Bajeh. "A Novel Metric for Measuring The Readability of Software Source Code", Journal of Emerging Trends in Computing and Information Sciences. Vol. 2. No. 10. October (2011), pp.492-497.

Muhammad Ilyas received a Master degree in Software Project Management in 2004 from National University of Computer and Emerging Sciences, Lahore and a Doctor of Informatics from Johannes Kepler University, Linz Austria in 2010. His research interests include Software Engineering, Design Pattern and knowledge base systems. He is currently an assistant professor in the Department of Computer Science and Information Technology at the University of Sargodha, Pakistan.

Mubashir Abbas is a student of the Master of Science in Computer Sciences at the University of Sargodha. He has already completed his BS 4 year degree in computer sciences. His research interests include Software Engineering, Software Reusability and other topics.

Khansa Saleem is a student of the Master of Science in Computer Sciences at the University of Sargodha. Her research interests include Software Engineering, Semantic searches and other topics.