Design of Dynamic Component Reuse and Reusability Metrics Library for Reusable Software Components in Context Level

V. Subedha
Research Scholar
Sathyabama University
Chennai, India

S. Sridhar
PhD, Research Supervisor
Sathyabama University
Chennai, India

ABSTRACT
Reusability is about building a library of frequently used components based on the functional requirements of the reuser. A well organized component reuse library is the key for successful reusability in terms of economics benefits. Reusability metrics is a set of guidelines to help reuser to judge the quality of the component that is to be reused. Reusability metric library is an essential ingredient of a successful reuse in context level. In this paper, we outline architecture for reusability driven methodology in context level and we also design dynamic libraries for qualitative analysis of the components. These libraries have to be designed for reusing efficient and quality reusable software components. Our approach for identifying and qualifying of reusable software components is based on functional coverage report, extraction time and reuse frequency of the component. In this paper we describe some case studies to validate our experimental approach. This architecture will be a base to develop efficient searchable, reuser-friendly, useful and well organized dynamic libraries. Component reuse percentage is measured by the percentage of qualified components for reuse. So, the proposed architecture and the dynamics libraries can be used to improve the productivity and quality of reusability.

Keywords
Reusable Software Components, Reusability, Reuse Metrics, Extraction Time, Component Identification, Component Qualification, Reuse libraries.

1. INTRODUCTION
Reuser in practice adopting many reuse approaches including reuse in product lines, design pattern templates, reference architectures, context independent components addresses reuse in different ways and have also demonstrated benefits [1]. Reusability of software components is a challenge in any environment. The reusability of high quality software components at an affordable cost and within in a limited time scale is always desired by reuser [2].

A great deal of research over the past several years has been devoted to the development of methodologies to create reusable software components and component libraries, where there is an additional cost involved to create a reusable component from scratch. That additional cost could be avoided by identifying and extracting reusable components from the already existing environment. But the issue of how to identify good reusable components from existing systems has remained relatively unexplored [3].

Reuse libraries are the critical element of successful reuse program. So reuser has to put more effort to develop and maintain reuse libraries. Also in order to measure reuse success, the library must collect and analyze considerable data. Effective classification schemes are necessary to assist the reuser in locating and comparing library components for reusability. There is different reuser for whom a component reuse library is necessary and each reuser have somewhat different component reuse library based on their different requirements. So the libraries are designed in a dynamic manner according to the reuser requirements.

The aim of Software Metrics is to predict the quality of software products. To ensure the quality of component we use four primitive metrics and classify them according to their quality. Then the qualified software components are stored in the component reuse library for potential reuse. The architecture which we proposed will help the reuser to identify, extract and qualify reusable components. In our libraries we adopt combination of metrics for classifying the components into two parts.

The remainder of the paper is organized as follows : The next section discusses the related work and existing solutions in reusable software components. Section 3 briefly explains the architecture of reusability in context level. Section 4 presents the cases study using the proposed dynamic approach and comparative analysis. The last section concludes the paper and outlines the future work.

2. RELATED WORK
In recent years, there has been an increasing awareness of reusability of the software components. In this scenario, a critical issue is to identify, extract and qualify reusable components. Therefore, finding a method to retrieve the reusable components from existing environment represents an important activity. In this section, we explain briefly about existing metrics and models for qualifying the components for reusability.

In [4], they presented a historical review of the reuse concepts, and ontology based on reuse definitions and its relation with quality assurance processes. Clarity and formal specification of concepts play a key role in the inclusion of reuse in the software development process, and in the subsequent development of a corporate strategy. The Systematic Reuse Approach uses a library for storage, control, distribution and management of reusable software elements.

The most common proposed approach was to define metrics to assess the reusability. In [5], a coupling and cohesion metrics suite is presented to evaluate object-oriented software. These metrics may be applied to assess reusability.

In [6], they proposed combined metrics to measure coupling and cohesion to rank the reusability of the components.

Metrics are applied to three types of component to generate the results. An assessment framework for reusability and reusability attribute model for aspect oriented product line components are proposed in [7].

Classification of the reusability of software components using Support Vector Machine is presented in [8]. Also the identification of reusable software modules in Procedure Oriented System is based on software metrics like Cyclometric complexity, Volume, Regularity, Coupling and Reuse frequency.

In [9] they proposed the framework for evaluating reusability of procedure oriented system using software metrics. The proposed metrics for this framework are Cyclometric complexity, Volume, Regularity, Coupling and Reuse frequency.

The contribution of metrics to the overall objective of the software quality is very well understood and recognized. But how these metrics are collected and determining the reusability degree of a software component is still in the research stage. So, in this paper we collected the metrics like functional coverage report, extraction time and reuse frequency and based on the decision tree classifier the software components is classified into two parts : Qualified and Not Qualified for reusability.

3. ARCHITECTURE FOR REUSABILITY IN CONTEXT LEVEL

In this section we first explain the phases involved in Reusability of Software components based on the Functional behavior in context level.

The reuse process can be divided into three phases

- Component Identification
- Component Extraction
- Component Qualification

Fig. 1 shows the segments of the architecture for reusability in context level

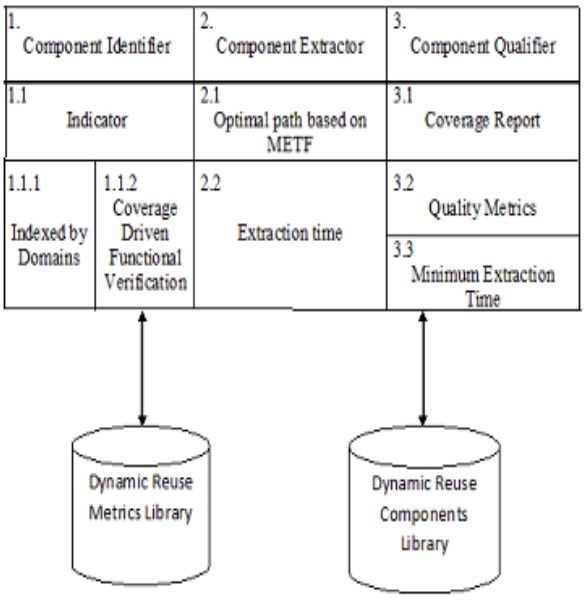

Fig 1 : Architecture for reusability in context Level

3.1 Component identifier

The component identifier supports the reuser to identify the candidate components based on the functional request queries. This system stores the reusable candidates in the dynamic reuse components library for processing in the remaining phase. Indicators are used to identify the candidate components based on keywords and coverage driven functional verification method. Initially the component reuse library will contain components with functional specification as keywords. The indicator has two segments

- Indexing by Domains: To identify the components for reusing the earliest and easiest method is indexing. The reuser must formulate a keyword that matches the functional requirements. The component identifier itself compares with the indexes and returns the components that match the Domain name.
- Coverage driven functional verification: Using the functional specification the reuser generates, executes and associates with component a set of test cases and functional coverage report of the component is collected. Three commonly used measures of coverage driven functional verification are statement coverage, branch coverage and logical path coverage. We use the statement coverage and branch coverage to identify the candidate component reuse. In [10] Fenton considers that normally a developer will insist on 100% statement coverage and high branch coverage of around 85%. So, we identify the component to be reused if the coverage report contains 100% statement coverage and 85% branch coverage. The main important advantages of coverage driven functional verification in reuse is to identify the faulty component whose faulty behavior matches the functional request of the reuser and this component is also added to the component reuse library dynamically. Since we are reusing the faulty components the reusability level of the environment is improved which yields to high potential benefits.

After identify the components in the environment the component are added to the dynamic component reuse library with the functionality and also component reuse metrics library is created with the following attributes, distance of the components from the current reuser, percentage of statement coverage, branch coverage and reuse frequency of the component. The component reuse library is centrally controlled and managed and it is updated dynamically when the component is identified for every reuser. But the component reuse metrics library is created separately and dynamically for each reuser and deleted after the task is completed

3.2 Component extractor

After identifying the components for reuse the component extractor find outs an optimal path for component extraction and extraction time of each component which behaves as a qualifier in qualification phase.

Optimal path : In this segment we get the optimal path just by travelling to the nearest components from the current reuser position. This optimal path helps us to calculate the extraction time for each and every component. To find out the optimal path and extraction time we use a method called Minimum Extraction Time First (METF).

Extraction time : The first component in the optimal path will have the minimum extraction time and extraction time of the component is used as a qualifier in qualification phase in terms of speed. After calculating this attribute is added to the dynamic reuse metrics library.

3.3 Component qualifier

The resuer associates each reusable component identified with set of attributes for qualification in dynamic reuse metrics library. The identified components are analyzed more carefully in the context of functional coverage report, minimum extraction time and reuse frequency to classify the components into two parts: Qualified and Not Qualified. The qualified set will yields the high quality and high potential reuse components. We call this process as "Qualification". The component qualifier has three segments

Functional coverage report : The functional coverage report consists of statement coverage and branch coverage. The cut-offs for qualifying the component based on statement coverage is 100% and the for branch coverage it is divided into three level 85-90% as LOW, 90-95% as MEDIUM and greater than 95% as HIGH.

Reuse Frequency : The reuse frequency is an indirect measure of the functional usefulness of a component. We measure the functional usefulness that frequently used system is a good candidate for reuse in context level in similar domain. Hence we choose the metrics reuse frequency as a qualifier for classifying the components. Reuse frequency of each component can be calculated using the equation (1).

$$\text{Reuse Frequency} = \frac{n(C)}{\frac{1}{n}\sum_{i=1}^{n} n(Si)} \quad (1)$$

where n(C) is total number of reference to the Component, n(Si) is total number of reference for each Standard Components in the existing environment & n is the total number of component in the existing environment

Minimum Extraction time : The components having the extraction time less than the average extraction time is qualified for reuse. The reason for choosing the extraction time as metrics is to speed up the process of reuse.

Component Reuse percentage measures how much components are qualified for reused from identified set and it is given as in equation (2)

$$\text{Reuse Percentage} = \frac{n(Q)}{n(Q)+n(NQ)} * 100\% \quad (2)$$

where n(Q) is the number of components qualified for reuse and n(NQ) is the number of components not qualified for reuse.

4. CASE STUDY AND ANALYSIS

In this section we describe experiments with proposed architecture for identifying and qualifying the components with our own test cases. Also the performance of dynamically created libraries is compared with some existing component reuse libraries in the literature. Some goals of the case studies were

- Evaluate the concept of reusability from the existing environment
- Study the application of the proposed architecture to the existing environment for reusability
- Analyze the metrics used in the different phases of reusability
- Classifying the set of component which is qualified for reuse.

For experimental study Local Area Network Environment with following specification where chosen:

- No. of Nodes=5000 i. e node 0 to node 4999 i. e Distance[C_{end}]=4999
- Present reuser position is 1919 i. e distance [C_{st}]=1919 th node
- C_i denotes the components in the Network

We performed the case study according the these constraints

- Making sure that all the necessary information are available
- Computing the four metrics and measuring criteria
- Reuse the qualified components based on the quality of the components

Initially the component reuse library will contain some set of components with functionality of the component was indexed by the domain keywords and will contain the reuse frequency of that particular component. In the existing environment in some node there may be faulty behavior components. The reuser according to their functional specification they generate the automatic test cases using some tool and based on coverage analysis report a set of components are identified and a dynamic metrics library is created for the particular reuser with the characteristics in Table 1. The metrics used for identification is statement coverage with 100% and branch coverage with 85% and the distance of the components from the current reuser position also maintained in the metrics library. The reuse frequency is calculates using the equation (1).

Table 1: Dynamic metrics library after identification phase

Identified Component	Distance of C_i	Statement Coverage	Branch coverage	Reuse frequency
C1	2496	100%	85%	1.36
C2	1285	100%	90%	0.30
C3	793	100%	85%	0.41
C4	2195	100%	95%	1.62
C5	86	100%	95%	0.68
C6	2693	100%	95%	1.09
C7	999	100%	90%	0.90
C8	1596	100%	85%	0.75
C9	2105	100%	95%	1.93
C10	121	100%	85%	1.78

In the component extraction phase the extraction time of each component in the identified set is calculated using the minimum Extraction Time First (METF) method. Also average extraction time is calculated where this metrics will

behave as a qualifier in the qualification phase in order to speed up the reuse process. The optimal path for the component extraction after qualifying is also decided in this phase itself. Table 2 outlines the updated metrics library with extraction time after extraction phase. The optimal extraction path for the component extraction is shown in the fig. 2.

Average Extraction Time : 0.492

Optimal path : C9 →C4→ C1→ C6→ C8→ C2→C7→ C3→ C10→ C5

Table 2: Dynamic metrics library after extraction phase

Identified Component	Statement Coverage	Branch coverage	Reuse frequency	Extraction time
C1	100%	85%	1.36	0.160
C2	100%	90%	0.30	0.606
C3	100%	85%	0.41	0.743
C4	100%	95%	1.62	0.077
C5	100%	95%	0.68	0.939
C6	100%	95%	1.09	0.215
C7	100%	90%	0.90	0.686
C8	100%	85%	0.75	0.520
C9	100%	95%	1.93	0.052
C10	100%	85%	1.78	0.929

The optimal extraction path for the component extraction is shown in the fig. 2.

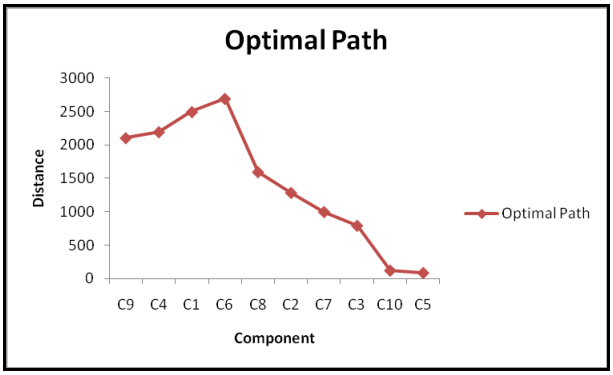

Fig 2 : Optimal path using METF

In the qualification phase the metrics are chosen for analyzing the quality of the components and the qualifying criteria is defined. If statement coverage is equal to 100% and branch coverage is above 90% and if the reuse frequency is HIGH and if the extraction time is less than the average extraction time then the component is classified as Qualified or Not Qualified. These analyses are done dynamically for each and every request of the reuser.

Once component is qualified and reused then the dynamic metrics library is deleted which is created for the particular reuser. The data in the last column of the Table 3 is the status of the component. If the component is reused by the reuser the reuse frequency is calculated with eq. (1) and component reuse library is updated with the reuse frequency value.

Table 3: Dynamic metrics library after qualification phase

Identified Component	Status
C1	Not Qualified
C2	Not Qualified
C3	Not Qualified
C4	Qualified
C5	Not Qualified
C6	Qualified
C7	Not Qualified
C8	Not Qualified
C9	Qualified
C10	Not Qualified

Accordingly, Table 3 presents the measurement data for high reuse components and it shows that in general 25 to 35 percent of identified components for possible reuse. These case studies show that reusable components have measurable properties that can be used as a qualifier for the quality of the components for reuse.

This study suggests the potential benefit of the components reuse percentage is measured by the equation (2)

Component reuse percentage = 30%

Comparison of Identification Indicators for all the repositories with our purpose architecture is listed in Table 4

Table 4 : Comparative analysis with the indicators of existing reuse libraries

Component Repositories	Identification by Indicators
SALMS	Keywords
ASRR	Keywords
AIRS	Facets Approach
RLT	Keywords
DSRS	Keywords
I-CASE	Keywords
MORE	Keywords
LID	Keywords
DAL	Keywords
CAPS	Browsing with Keyword
DYNAMIC COMPONET REUSE LIBRARY	Indexed with Keywords and Coverage driven Functional Verification Method

5. CONCLUSION

It is always a challenge for the reuser to find out a quality component for reuse. This paper focuses and proposes architecture for component reusability in context level. The fundamental motivation of the proposed architecture and dynamic libraries is to reduce the effort spent by the reuser to qualify the component candidates for reuse. The basic premise assumed by this approach is that reusable components have certain quality attributes like functional usefulness, extraction time, and reuse frequency. We foresee three major developments in this architecture.

- Supporting Component identification both by Keyword Queries and by Functional Specification.
- The keyword Query is the basic method where the identification is done by indexing the component reuse libraries.
- The proposed method coverage driven functional verification is used to identify the component based on functional behaviour and this method increases the reuse level by reusing not only the component repositories but also the faulty components which match the functional specification of the reuse.

The metrics library is a dynamic object where reusers can retrieve the measures on the time of reusability according to their specification. The component reuse library is also dynamic where components can be identified by indexing the keyword and by coverage driven functional verification. However, more research work needs to be done to develop perfect test cases to improve the component reuse. The future research efforts will be dedicated to come out with a perfect coverage report. We also need to broaden our analysis to different environments for broader verification of our proposed architecture. This architecture can be applied for service oriented architecture and cloud services.

6. REFERENCES

[1] Richard W. Selby, "Enabling Reuse-Based Software Development of Large-Scale Systems," IEEE Transaction of Software Engineering, Vol. 31, No. 6, PP. 495-510, Jun 2005

[2] Fazal-e-amin, Ahmad Kamil Mahmood and Alan Oxley, "A Review of Software Component reusability Assessment Approaches", Research Journal of Information Technology, pp. Vol. 3, 1-11, 2011

[3] Parvinder Singh Sandhu and Hardeep Singh, "Automatic Reusability Appraisal of Software Components using Neuro-Fuzzy Approach", International Journal Of Information Technology, vol. 3, no. 3, pp. 209-214, 2006

[4] Matteo Gaeta, Francesco Orciuoli, Stefano Paolozzi, and Saverio Salernol, "Ontology Extraction for Knowledge Reuse: The e-Learning Perspective ", IEEE Transactions on Systems, Man and Cybernetics, Vol. 41, No. 4, pp. 789-809 Jul 2011.

[5] Husein. S and A. Oxley, "A Coupling and Cohesion Metrics suite for Object-Oriented Software", Proceedings of International Conference on Computer Technology and Development, Malaysia, pp.421-425, Nov 2009

[6] Gui G and Paul D. Scott, "Measuring Software Component Reusability by Coupling and Cohesion Metrics", Journal of Computers, Vol 4, No 9, 797-805, Sep 2009

[7] Fazal-e-amin, Ahmad Kamil Mahmood and Alan Oxley, "A Reusability attribute model for aspect oriented software product line core assests." Proceedings of the International Symposium on Information Technology, Malaysia, pp.1138-1141, Jun 2010.

[8] Ajay Kumar, "Measuring Software Reusability using SVM based Classifier Approach", International Journal of Information Technology and Knowledge Management.", Vol. 5, No. 1, pp.205-209, Jan 2012

[9] Sonia Manhas, Rajeev Vashisht and Reeta Bharrdwaj, "Framework for Evaluating Reusability of Procedure Oriented System using Metrics Based Approach ", International Journal of Computer Applications (0975 – 8887), Vol 9, No.10, Nov 2010.

[10] Norman E Fenton and Martin Neil, "Software Metrics: Successes, Failures and New Directions", Journal of Systems and Software, Nov 1998.