Design of a Conceptual Reference Framework for Reusable Software Components based on Context Level

V. Subedha[1], Dr. S. Sridhar[2]

**[1] Research Scholar, Department of CSE, Sathyabama University
Chennai, Tamilnadu, India**

**[2] Research Supervisor, Department of CSE, Sathyabama University
Chennai, Tamilnadu, India**

Abstract

Reusable software components need to be developed in a generic fashion that allows their reusability in context level. Components identification based on quality metrics for reusability and indexing had been the desired technique in the field of reusable software components. However, the methodologies utilized for the identification of reusable components are not able to handle the reusability of faulty behavior component. In this paper we propose a conceptual reference framework for reusable software components which is available in reusable component repositories and also based on the faulty functional behavior of the components in the environment. Also we propose a Component Extraction scheme named as Minimum Extraction Time First (METF) based on extraction time of the component. The component for reuse is qualified based on the functional coverage report, software reuse metrics and minimum extraction time from the collection of components identified. Reuse-Utility-Percent and Reuse-Frequency metrics were used to assess the reusability in the environment. So, the proposed framework can be used to achieve high potential and high quality reuse.

Keywords: *Software reuse, Reusable Software Components, Components Identification, Component Extraction, Component Qualification, Reusability metrics.*

1. Introduction

Effective reuse of requirements, architecture, design, process, technology, knowledge and components from previous software developments can increase the productivity & quality in software environment [4]. Software reuse catalyzes improvements in productivity by avoiding redevelopment and improvements in quality by incorporating components whose reliability has already been established [7]. In fact, Software production using the reusable components will probably be very crucial to the higher level of software industry maturity [9]

So, in recent years, there has been an increasing awareness of the reusability. As a consequence there are lots of research studies which focus on the software components extracted from the existing sources [1]. The most extensive effort to date has been the focus on software reuse is to examine the issues ranging from methods and techniques and also to improve productivity and economic impact [2].

Reusability can be improved not only by replicating the software components but also continue to reuse the components with faulty functional behavior. Coverage driven functional verification plays a more and more important role in reusability of components with faulty behavior. Based on the coverage report generated by the Coverage driven test-cases the components are identified for reusability.

Component-based reuse is widely accepted as an important reuse strategy and component-based reuse programs heavily depend on software reuse repositories for achieving success [12].

Reuse Frequency metric and Reuse-Utility-Percent metric values are used as the assessment attributes for reusability of the software component in Context level. In this paper, we outline a way to reuse the software components corresponding to the context-dependent reusability.

The rest of this paper is structured as follows. Section 2 deals with some of the related research works. Section 3 describes our design of a conceptual reference framework for Reusable Software Components and the proposed algorithm for the framework. Section 4 deals with metrics to assess reusability and their evaluation. Section 5 discusses about the case study and analysis results. Finally, Section 6 concludes.

2. Background and Related Work

In the literature, the area of studies addressed by this paper is called Reusable Software Components. This term means the process of effectively reusing the components from existing environment on context level.

Reusability is an important goal in well engineered systems. Two main approaches have been developed to identify the components for reusability in existing software components. The first one facilitates the component identification based on indexing and the second approach based on metrics and models.

P. Vitharana et al.[11] describes a component retrieval mechanism that the reuser formulates a request based on the keyword. To be retrieved, the information about the components must be encoded. This process of identification is also called as indexing which may be manual or automatic but result in loss of information.

Richard W. Selby [7] proposed a Reuse based software development model in order to achieve an average reuse of 32% per project. They identify two categories of factors that characterize successful reuse-based software development of large-scale systems: module design factors and module implementation factors. Also they evaluate the fault rates of the reused, modified, and newly developed modules.

Matteo Gaeta et al. [1] proposed an approach to extract relevant ontology concepts and their relationships from a knowledge base of heterogeneous text documents. The proposed model is a complete methodology for automatic knowledge extraction for reuse.

Parvinder Singh and Sandhu and Hardeep Singh, [5] [6] have used metric based approach for identifying a software module and the reusability was obtained with the help of Fuzzy Logic and Neuro-Fuzzy. This research shows how metrics can be used to identify the quality of a software component.

Keiji Hokamura et al [3] proposed an approach of defining reusable components for multiple Web applications using a domain-specific aspect-oriented (AO) mechanism based on an abstraction model common to all Web applications. The domain-specific AO mechanism based on the server-client model of Web is useful to describe reusable components which implement functionalities affecting user page accesses. B. Morel et.al [10] proposed adapting software components at the architecture level.

3. Conceptual Reference Framework for Reusable Software Components

Our reference framework for reusable software components is to identify, extract, qualify and integrate reusable software components based on the functional behavior. This approach follows the well planned, efficient by cost and quality product. Fig. 1 shows the conceptual reference framework for reusing the software components.

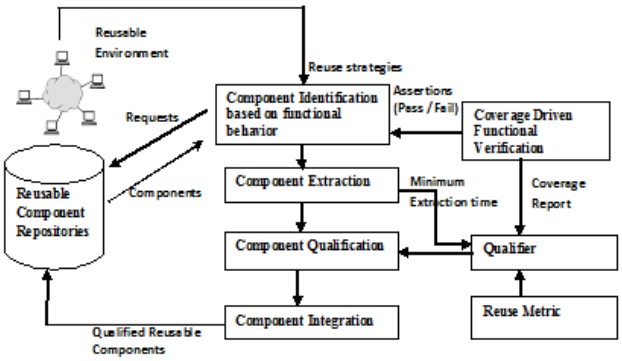

Fig. 1 Conceptual Reference Framework for Reusable Software Components

The whole process consists of the following phases

1. Component identification
2. Component extraction
3. Component Qualification

We focus on a problem as how to analyze the existing replicant and faulty component for identifying the collection of suitable components for use. After once they identified how it is extracted and how to qualify them for appropriate use is the next issue. Our approach to identification, extraction and qualification of reusable software components is based on functional behavior of the component, minimum component extraction time and reusability metrics.

In the component identification phase, to identify the component the information about the components must be encoded. In our approach both replication of software components and the failover components are maintained in the repositories. The functionality of the component is indexed and the functionality request from the existing environment will be compared with the indexes and a collection of match components will be extracted. So the Reuse Utility Percent and Reusable Frequency can be improved by the reuse of replication and faulty components. Coverage driven functional verification method generates the test case for functional verification of the components and the reused components were identified.

Component Identification process is divided into two steps.

(1) The existing environment sends the request based on functionality.
(2) The coverage driven functional verification is done for both replicant components and as well as faulty behavior components. And a set of components that possibly satisfying the need is returned.

The next phase in this framework is component extraction. In this phase the identified components which are scattered in the existing environment is extracted for reusability. In our approach we propose a new Component Extraction scheme named as Minimum Extraction Time First (METF).

In this proposed scheme we extract a component which is in the nearest distance to the current position of the environment. By using this scheme we can extract the entire identified reusable component in average minimum extraction time from the current position in the existing environment. This scheme defines an optimal path for component extraction and also calculates extraction time for each component.

After extracting a set of reusable software components, in the qualification phase we qualify the reusable software components based on the coverage report, minimum extraction time to extract the components and software reusability metrics. After qualification the component is integrated with existing environment so that the system is able to continue its usual work.

The following general algorithm illustrates the software component reusability process proposed in this framework

Begin

Step 1: Specify the Reuse Strategy
Step 2: Index the components with coverage driven functional verification Report
Step 3: **If** identical match with Components in repositories or Components with faulty behavior **then** identify the component for reuse
Step 4: **For** entire set of identified component for reuse **do**
Calculate the minimum extraction time with METF scheme
Step 5: Qualify the component based on Coverage report, Minimum extraction time and quality metrics
Step 6: Integrate the reusable software component with the environment

End

4. Metrics to assess reusability

A pair of metrics Reuse-Utility-Percent and Reuse Frequency has been used to measure the reusability of the components. Linguistic variables are then assigned to the metrics based on the measurement. The assignment of the linguistic variables depends on the range of the values of the measurement.

Reuse-Utility-Percent is the most important reuse metrics and this metric is very simple to measure. It is the ratio of No. of Reused software components to the No. of software components available in the existing environment. Reuse-Utility-Percent is assigned with six linguistic variables VERY HIGH, HIGH, MEDIUM, LOW, VERY-LOW and NIL as constants in the range of 0-100 in Table 1.

$$\text{Reuse UtilityPercent} = \frac{n(RSC)}{n(SC)} * 100 \quad (1)$$

where n(RSC) is total number of Reusable Software Components & n(SC) is total number of Standard Components existing environment

Table 1 : Linguistic variables for Reuse Utility Percent

Reuse Frequency Range	***Linguistic variables***
0 – 10	NIL
10 – 30	VERY LOW
30 – 50	LOW
50 – 70	MEDIUM
70 - 90	HIGH
90 - 100	VERY HIGH

Reuse-Frequency is the ratio of the count of a component refereed for reuse to the total count of references of the entire standard component in the existing environment.

Reuse-Frequency is assigned to two linguistic variables LOW and HIGH as constants in the range of less than 1 and greater than 1.

$$\text{Reuse Frequency} = \frac{n(C)}{\frac{1}{n}\sum_{i=1}^{n} n(Si)} \quad (2)$$

where n(C) is total number of reference to a Reusable Software 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 in the Table 2

Table 2: Linguistic variables for Reuse-Utility-Ratio

Reuse Utility Percent Range	***Linguistic variables***
<=1	LOW
>1	HIGH

The equation (2) shows that the Reuse Frequency is the measure of function usefulness of a component. Hence there should be some minimum value of Reuse Frequency to make software component really reusable.

5. Case Study and Analysis

In this section we present a case study to evaluate the effectiveness of component extraction scheme and analysis of reusability metrics with our own test cases

5.1 Evaluation and Discussion

The minimum extraction time for all the components in the entire identified set is calculated. The Total Component Extraction time and Average Component Extraction for the proposed scheme is derived in this subsection. Total Component Extraction time is the total time taken to extract all the reusable components which are identified based on the functionality in Component Identification Phase. Average Component Extraction time is the average time taken to extract all the reusable components

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

1. No. of Nodes=5000 i. e node 0 to node 4999 i. e Distance[C_{end}]=4999
2. Present reuser position is 143 i. e distance [C_{st}]=143 rd node
3. The Component Extraction distances of [C_i] were chosen in Table 3
4. C_i denotes the position in the Network from where the Component for the i^{th} position has to be extracted.

Table 3: Distances of C_i for different Reusable Components of 'i'

Identified Component i	***Distance of C_i***
1	86
2	1470
3	913
4	1774
5	948
6	1509
7	1022
8	1750
9	130

The component Extraction path and minimum extraction time each component in the optimal path for proposed scheme shown in the Table 4.

Table 4: Extraction path & Extraction time using METF

Extraction Path	***Distance of C_i***	***Extraction Time of component C_i***
9	130	0.004
1	86	0.016
3	913	0.246
5	948	0.255
7	1022	0.276
2	1470	0.400
6	1509	0.411
8	1750	0.478
4	1774	0.485

The Total distance travelled is, T_D = | Distance [C_{st}] - Distance [C_9] | + | Distance [C_9] - Distance [C_1] | + | Distance [C_3] - Distance [C_1] | + | Distance [C_5] - Distance [C_3] | + | Distance [C_7] - Distance [C_5] | + | Distance [C_2] - Distance [C_7] | + | Distance [C_6] - Distance [C_2] | + | Distance [C_8] - Distance [C_6] | + | Distance [C_4] - Distance [C_8] |

T_D =13+44+745+827+35+74+448+39+241+24 = 1745
Average distance = 1745/9 = 193.88

Total extraction time = 2.570
Average extraction time = 0.286

Extraction path for proposed Extraction Minimum Extraction Time First (METF) method is given in the below Fig 2.

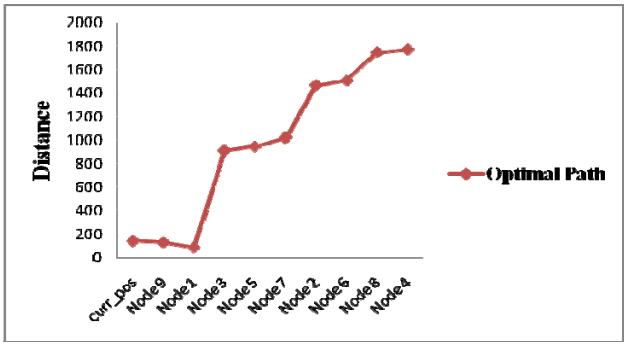

Fig. 2 Extraction path of Reusable Component Extraction (METF)

5. 2 Analysis of reusability metrics

For a better explanation of reusability metrics, consider the example of a Local Area Network Environment consists of N=500 components and for 10 scenarios in various cases, the Reuse Utility Percent metric is calculated by using the equation (1) and the linguistic variables are assigned for the range of metric value which is already assigned in the Table 1

Total No. of Standard Components n(SC) = 500

Table 5: Linguistic variables for Reuse-Utility-Percent for different Test cases from NIL to VERY HIGH

Test Cases	*No. of Reused Components n(RSC)*	*Reuse Utility Percent*	*Linguistic variables*
1	59	11.8	VERY LOW
2	385	77	HIGH
3	193	38.6	LOW
4	412	82.4	HIGH
5	323	64.6	MEDIUM
6	215	43	LOW
7	89	17.8	VERY LOW
8	5	1	NIL
9	455	98	VERY HIGH
10	299	59.8	MEDIUM

Table 5, summarizes the number of test cases collected under the LAN environment and Reuse Utility Percent is calculated using the equation (1) and linguistic variables are assigned. From this table, it is inferred that the reuse-utility percent increases when number of component reused is increases

From the below graph in Fig.3 we can infer the relationship between the No. of Reused Components and Reuse-Utility-Percent as Reuse-Utility-Percent is directly proportional to the no. of Reused Components.

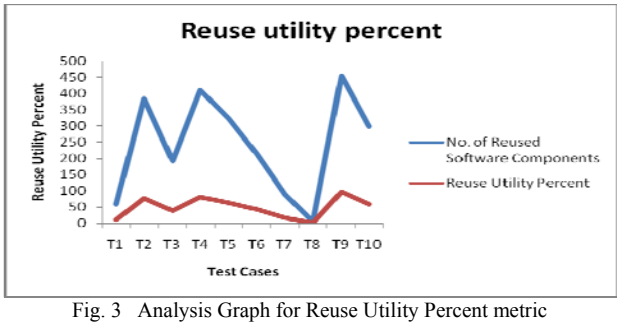

Fig. 3 Analysis Graph for Reuse Utility Percent metric

Consider an another example application of a local area network consists of N=10 reusable components in an application for various references, the reuse frequency metric is calculated and the linguistic variables are assigned for the range of metric value which is already assigned in the Table 2

Total No. of reference for Standard Components Σn(Si) = 225

Table 6: Linguistic variables for Reuse-Frequency for the reused Components

Component Number	*No. of Reference to a Component n(C)*	*Reuse Frequency*	*Linguistic variables*
1	8	0.35	LOW
2	0	0	LOW
3	15	0.6	LOW
4	54	2.4	HIGH
5	23	1	LOW
6	33	1.46	HIGH
7	4	0.1	LOW
8	42	1.86	HIGH
9	19	0.84	LOW
10	27	1.2	HIGH

Table 6, summarizes the number of components reused under the LAN environment and reuse-frequency is calculated using the equation (2) and linguistic variables are assigned. From this table, it is inferred that the reuse-frequency for a component increases when number of references for a component is increases

From the below graph in Fig.4 we can infer the relationship between No. of references to a Component and Reuse-frequency as Reuse-frequency is directly proportional to the total no. of references to a Component in the existing environment.

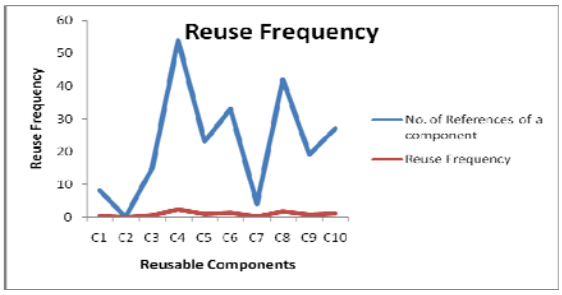

Fig. 4 Analysis Graph for Reuse Frequency metric

6. Conclusions

In this paper, we described a framework for reusing the software components in context level. This framework also provides diversity in the execution environment leading to a higher level of reliability of the system. By this new approach we can includes the reuse benefits as reduction in development effort and maintenance effort. This approach is feasible for building reliable software systems using the reusable components. Finally, the complete cycle of phases can enable the reuser to determine which components have high reuse potential with regards to specific functional requirements, minimum extraction time and measures of reusability metric. This research also shows how metrics can be used to find the quality attributes of a software component. The long term goal of this approach is to provide a distributed software component repository to support the system development though internet and cloud services.

References

[1] 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.

[2] Gan Wang, Ricardo Valerd and Jared Fortune, "Reuse in Systems Engineering IEEE Systems Journal, Vol. 4, No. 3, pp. 376-384, Sep. 2010

[3] Hokamura, K., Ubayashi, N., Nakajima, S., Iwai. A, "Reusable aspect components for Web applications" in TENCON 2010 - 2010 IEEE Region Conference , pp. 1059 – 1064, Nov. 2010

[4] Freya H. Lin, Timothy K. Shih, , and Won Kim, "An Implementation of the CORDRA Architecture Enhanced for Systematic Reuse of Learning Objects", IEEE Transactions on Knowledge and Data Engineering , Vol. 21, No. 6, pp 925 – 938, JUNE 2009

[5] Parvinder Singh Sandhu and Hardeep Singh, "A Fuzzy-Inference System Based Approach for the Prediction of Quality of Reusable Software Components", International Conference on Advanced Computing and Communications, ADCOM 2008. On page(s): 349 – 352

[6] 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, 2006, pp. 209-214.

[7] 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

[8] Tomer, L. Goldin, T. Kuflik, E. Kimchi, and S.R. Schach, "Evaluating Software Reuse Alternatives: A Model and its Application to an Industrial Case Study," IEEE Trans. Software Eng., vol. 30, no. 9, pp. 601-612, Sept. 2004

[9] Marcus A. Rothnberger, Kevin j. Dooleg, Uday R. Kulkarni and Nader Nada, " Strategies of Software Reuse : A Principal Component Analysis of Reuse Practices", IEEE Trans. Software Eng., vol. 29, no. 9, pp. 825-837, Sep. 2003.

[10] Morel. B and Alexander, "A Slicing Approach for Parallel Component Adaptation," Proc. 10th IEEE International Conference and Workshop the Eng. of Computer-Based Systems, pp. 108-114, Apr. 2003

[11] Vitharana. P, Zahdi. F and Jain. H, "Design, retrieval and assembly in component-based software development", Communications of the ACM, 46(11), 97-102, 2003

[12] Guo. J, Luqui, "A Survey of Software Reuse Repositories", 7th IEEE International Conference and Workshop on the Engineering of Computer Based Systems, pp. 92-100, Apr. 2000.