A decision support system through conceptual reference framework for reusable and functional verification environment

Subedha V.¹, Mohanaprakash T.A.², Venkataraman³, Brindhadevi S.⁴
1. Professor & Head, Panimalar Institute of Technology, Chennai, India.
2. Assistant Professor-Grade I, Panimalar Institute of Technology, Chennai, India.
3. Assistant Professor-Grade I, Panimalar Institute of Technology, Chennai, India.
4. Assistant Professor, Panimalar Institute of Technology, Chennai, India.
DOI: 10.6088/ijaser.04018

Abstract: Software reuse is viewed as a means for achieving rapid system development, saving resources and time, and keeping up technologically in an increasingly advancing global environment. It is also proved to be an efficient way to help improving software development quality and productivity. It is always a challenge for the reuser to find out a quality component for reuse. To best utilize software reuse, we created a framework as a DSS, to enable reuse within a system. This research work focuses and proposes framework for component reusability in context level by describing all elements required for effective reuse. The fundamental motivation of this framework 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.

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 framework also provides diversity in the execution environment leading to a higher level of reliability of the system. Two issues that affect the reusability in an environment are accessibility and environmental difference and we have not provided proper solutions to these issues. Another difficulty that arises in attempting this framework is that some of the benefits of choosing the right component may not be evident in the short term. Furthermore, to make the framework useful in practice, there are needs for integrating our technology with supporting tools. There are scopes for further research in this direction.

Key words: Decision support system, reliable software systems, conceptual reference framework.

1. Introduction

Effective reuse of a software product will increase the productivity, reliability and maintainability. It saves the development and verification time and reduces the risk and the cost involved in the software development. The motivation for finding out the Reusable Software Components (RSC) of high potential and quality is got from business perspective. Since competitive products released earlier in market typically draw the most revenue, time-to-market is a critical factor for the commercial success of a product. We intend to design a conceptual reference framework which presents new ideas of verifying the functionality of the reusable software components before reusing in higher abstraction level. Another motivation is to improve the verification flow for higher assurance of quality reusable software components which can be
reused in critical and very large scale system development. In addition, we also derive a set of quality measure for reusable software components which are collected in different level to ensure the quality of the components for high potential reuse.

2. Related work

Mili et al (2002) defined the software reuse as “Software reuse is the process whereby an organization defines a set of systematic operating procedures to specify, produce, classify, retrieve, and adapt software artifacts for the purpose of using them in its development activities.” Margono and Rhoades (1992) pointed out some of the common reuse problems. Heineman and Councill (2001) proposed the definition. A summary of the challenges, that we should be following Ivica and Magnus (2002) are component abstraction, functional properties, cost, maintenance needed, configuration management, retrievable and re-implementation. Jiang Guo and Luqi (2000) claimed that reusable software component should have some functionality. They are simulation based verification, assertion based verification, formal verification, and coverage based verification (Wile et al, 2005). Min-An Song et al (2006) proposed a functional verification environment for advanced switching architecture.


2.1 Conceptual Reference Framework for Reusable Software Components
The Conceptual Reference Framework for reusing the software components is to identify, extract, qualify and integrate the components based on their functional behaviour. Figure 1 shows the conceptual reference framework for reusing the software components.

![Conceptual Reference Framework](image)

**Figure 1: Conceptual Reference Framework**

### 3. Reference framework for Reusable Software Components

The whole process consists of the following phases: Component Identification, Component Extraction, Component Qualification.

### 4. Proposed Algorithm

The Coverage Driven Functional Verification approach makes coverage rate as the core that drives the whole verification flow. The test generation engine takes the coverage metrics as the goal for vector generation and on the other hand relies on the coverage analysis technique for further generation. There are two key issues for functional verification task:– Generating proper testing scenarios that stress the functional behaviour of the Software component Under Verification (SUV); Determining the amount of different tests needed to reach enough code coverage to ensure that the SUV is bug free and satisfy the functional requirement of the user; A coverage driven functional verification based on genetic algorithm is used to optimize the coverage rate and to detect errors. Genetic algorithm is used to adaptively enhance the coverage rate through the automatic generation of appropriate test patterns. This approach is also used to speed up the verification process and ensure completeness. As result of this verification approach statement coverage and branch coverage is achieved. We use a simple equation for fitness evaluation that maximizes the coverage rate of the whole coverage system as follows:

\[
\text{Fitness function for Statement Coverage} = \frac{\text{No. of Lines Covered}}{\text{Total No. of LOC}}
\]

\[
\text{Fitness function for Branch Coverage} = \frac{\text{No. of Branches Covered}}{\text{Total No. of Branches}}
\]
4.1 Experimental results from coverage driven functional verification

For our test experiment, a generic security protocol which transfer data in reliable and secure manner is chosen as the reusable software component in a real time environment. The statement coverage and branch coverage metrics in represented in the Figure 2 and the comparative analysis is given in Figure 3.

![Figure 2: Statement coverage and Branch coverage](image)

![Figure 3: Comparative analysis of coverage report](image)

4.2 Experimental results from extraction methods

We have done simulation and the comparative analysis of these two methods is given in the Figures 4 and 5.
A decision support system through conceptual reference framework for reusable and functional verification environment

4.3 Component qualification phase

In the component identification phase, we may get many candidate components and in that identified set the minimum extraction time is used as one of the qualifier to qualify the component to be reused.

The proposed qualitative analysis model is used to assess the component quality for high potential benefits and high quality reuse in order to increase the productivity.

**Figure 4:** Extraction Time analysis FIFE and METF

**Figure 5:** Total and Average Extraction Time using FIFE and METF
A decision support system through conceptual reference framework for reusable and functional verification environment

Figure 6: Qualitative analysis Model

As shown in the Figure 6 the specific steps involved in the qualitative analysis model.

4.3 Metrics for coverage analysis, time and functional usefulness

Four primitive metrics are selected to measure the three factors. Table 1 contains the definitions of the metrics used for measuring the coverage analysis, time and functional usefulness.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statement coverage</td>
<td>This metric reports whether each executable statement is encountered</td>
</tr>
<tr>
<td>Branch Coverage</td>
<td>This metric reports whether Boolean expressions are tested in control structures</td>
</tr>
<tr>
<td>Extraction Time</td>
<td>This metric is used to measure the Extraction time of each component</td>
</tr>
<tr>
<td>Reuse Frequency</td>
<td>This metric is an indirect measure of the functional usefulness of a component</td>
</tr>
</tbody>
</table>

Using these measures, the identified set of components is classified into two categories: qualified set and not qualified set for reusability. The qualified set for reusability will give high potential and high quality reusable components which will increase the reuse frequency and reuse utility level. The reusability degree of the component can be obtained with the help of Fuzzy Rules. Let the membership functions of fuzzy sets $A_i, B_i, C_i, D_i$ where $i=1, 2, 3, \ldots, n$ be $\mu_{A_i}, \mu_{B_i}, \mu_{C_i}, \mu_{D_i}$. Evaluating the rule premises results in

\[ R_i = \mu_{A_i}(u) + \mu_{B_i}(x) + \mu_{C_i}(y) + \mu_{D_i}(z) \quad (3) \]

where $R_i$ is the reusability degree of the $i^{th}$ component. With the help of the fuzzy-rules the knowledge base
for calculating the reusability degree of the reusable software components is calculated. After collecting the metrics for assessing the reusability degree and assigning the linguistic variable for each metrics, we select the suitable weight values to compute the quality measurement value for assessing the reusability degree. Production rules can be generated to select the weight value. Based on the qualitative analysis model a web based tool is designed in order to qualify the component for reusability in an efficient manner.

4.4 Experimental results

We describe experiments with proposed Qualitative Analysis Model for qualifying the components with controlled experiment in real-time environment. All necessary information for calculating the quality of the components is given in the following Table 2.

**Table 2: Identified Set of components for reusability with all metrics**

<table>
<thead>
<tr>
<th>Component Number</th>
<th>Statement Coverage</th>
<th>Branch Coverage</th>
<th>Extraction Time</th>
<th>Reuse Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>95%</td>
<td>85%</td>
<td>0.941</td>
<td>0.80</td>
</tr>
<tr>
<td>2</td>
<td>100%</td>
<td>95%</td>
<td>0.047</td>
<td>1.70</td>
</tr>
<tr>
<td>3</td>
<td>100%</td>
<td>80%</td>
<td>1.023</td>
<td>0.71</td>
</tr>
<tr>
<td>4</td>
<td>100%</td>
<td>95%</td>
<td>0.177</td>
<td>0.98</td>
</tr>
<tr>
<td>5</td>
<td>100%</td>
<td>90%</td>
<td>0.056</td>
<td>0.41</td>
</tr>
<tr>
<td>6</td>
<td>100%</td>
<td>85%</td>
<td>0.320</td>
<td>1.50</td>
</tr>
<tr>
<td>7</td>
<td>100%</td>
<td>95%</td>
<td>0.913</td>
<td>1.40</td>
</tr>
<tr>
<td>8</td>
<td>100%</td>
<td>95%</td>
<td>0.781</td>
<td>0.30</td>
</tr>
<tr>
<td>9</td>
<td>90%</td>
<td>75%</td>
<td>1.001</td>
<td>0.79</td>
</tr>
<tr>
<td>10</td>
<td>100%</td>
<td>80%</td>
<td>0.328</td>
<td>1.25</td>
</tr>
</tbody>
</table>

**Table 3: Classified Set of components for reuse and not reuse**

<table>
<thead>
<tr>
<th>Component Number</th>
<th>Weight for reusability degree using fuzzy rules</th>
<th>Linguistic variables For Reusability Degree</th>
<th>Status of the Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.36</td>
<td>LOW</td>
<td>Not Qualified</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>HIGH</td>
<td>Qualified</td>
</tr>
<tr>
<td>3</td>
<td>0.43</td>
<td>LOW</td>
<td>Not Qualified</td>
</tr>
<tr>
<td>4</td>
<td>0.84</td>
<td>HIGH</td>
<td>Qualified</td>
</tr>
<tr>
<td>5</td>
<td>0.80</td>
<td>LOW</td>
<td>Not Qualified</td>
</tr>
<tr>
<td>6</td>
<td>0.91</td>
<td>HIGH</td>
<td>Qualified</td>
</tr>
<tr>
<td>7</td>
<td>0.85</td>
<td>HIGH</td>
<td>Qualified</td>
</tr>
<tr>
<td>8</td>
<td>0.69</td>
<td>LOW</td>
<td>Not Qualified</td>
</tr>
</tbody>
</table>
A decision support system through conceptual reference framework for reusable and functional verification environment

Accordingly, Table 3 presents the measurement data for high reuse components and it shows that in general 40 to 50 percent of identified components for possible reuse. The reusability degree for the software components calculated using the fuzzy rules are given in the Figure 7.

![Figure 7: Reusability degree value for the components](image)

Table 4, summarizes the number of components qualified for reusability based on the reusability degree value in the context-specific environment

<table>
<thead>
<tr>
<th>Component number</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2</td>
<td>QUALIFIED</td>
</tr>
<tr>
<td>C4</td>
<td>QUALIFIED</td>
</tr>
<tr>
<td>C6</td>
<td>QUALIFIED</td>
</tr>
<tr>
<td>C7</td>
<td>QUALIFIED</td>
</tr>
</tbody>
</table>

Table 5, summarizes the number of components not qualified for reusability based on the reusability degree value in the context-specific environment

<table>
<thead>
<tr>
<th>Component number</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>NOT QUALIFIED</td>
</tr>
<tr>
<td>C3</td>
<td>NOT QUALIFIED</td>
</tr>
</tbody>
</table>
The Component reuse percentage for reusability is calculated and shown in the following Figure 8.

**Figure 8**: Component reuses percentage

### 4.5 Metrics to assess reusability

1. **Reuse-Utility-Percent** - is the ratio of the number of identified components for reusability with the standard available component in the repositories
2. **Reuse-Frequency** - is the ratio of the count of a component referred for reuse to the total count of references of the entire standard component
3. **Component Reuse percentage** - is the measures that how many components are qualified for reuse from identified set
4. **Precision** - is the number of true positives divided by the total number of elements labeled as positive class and false class.
5. **Recall** - is defined as the number of True Positives divided by the total number of elements that actually belong to the positive class (i.e. the sum of true positives and false negatives).
6. **Accuracy** - is the percentage of the predicted values that match with the expected values of the reusability for the given data.

### 5. Conclusions

Software reuse is viewed as a means for achieving rapid system development, saving resources and time, and keeping up technologically in an increasingly advancing global environment. It is also proved to be an efficient way to help improving software development quality and productivity. At present there is very
little guidance in standards on how best to utilize software reuse within an industry. It is always a challenge for the reuser to find a quality component for reuse. To best utilize software reuse, we created a framework to enable reuse within a system. This research work focuses and proposes framework for component reusability in context level by describing all elements required for effective reuse. The fundamental motivation of this framework 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. 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. This framework also provides diversity in the execution environment leading to a higher level of reliability of the system. My research differs from existing component reuse models because we describe how developers may actively explore a component during the evaluation phase.

6. References