COUPLING AND COHESION METRICS BASED MEASUREMENT OF SOFTWARE REUSABLE COMPONENT STRENGTH

Dr. M. Iyapparaja
Associate Professor, SITE School, VIT University, Vellore, Tamilnadu, India.
Email: iyapparaja85@gmail.com

Received on: 18.10.2016
Accepted on: 11.11.2016

Abstract

Reusability plays an irreplaceable role in software project development. Measure the individual strength of the reusable component which may fall in any relations like Uni-Directional / Bi – Directional / Virtual component relation. By preparing the coupling and cohesion metrics, additional tests are performed to measure its updating as the value is ‘1’, Then the reusable component is taken for integration otherwise the component is eliminated from the list of possible outcomes at once. In addition the component stability level is checked by analyzing the measurement value of Instability metric. If the result is ‘0’ then it confirms that the component has high stability and should be considered for incorporation process or it isn’t stable enough for industry application environment.

Keywords: Software reusability, RRRA model, Coupling, Cohesion, Stability.

Introduction

Components are core implications of the Software development life cycle process and hence every component which is tested for reusability is computed for its dependency with the other components. An object oriented programming may be designed with numerous objects related to each other and a level of dependency on the other objects for its function (7). Task focused and aligned on purpose object may or may not work independently. This factor also needs consideration as incomplete programs (dependent on other sub programs) or unwanted coding (independent but provides results to other sub-programs) may affect the performance of the system (4). In Rank based Reusability and Risk Assessment (RRRA) model using the SPRE(Scrutinizing, Prioritization and Rank based Extraction) framework the suitable reusable component (Re(C)) is extracted from the Software reusable component repository (SRCR)(5). Once the component is extracted from the repository it is mandatory to measure its individual strength according to the measurement of how effectively it depends with the non-extracted reusable components in SRC Repository. Appropriate measurement of dependency exists in each Re(C) with three key concepts in the object oriented design
Dr. M.Iyapparaja* et al. /International Journal of Pharmacy & Technology

namely coupling, cohesion and stability. Coupling is the degree to which the different subcomponents cooperate. On
the off chance that they are profoundly reliant then changes to one can have critical consequences for the conduct of
others. Subsequently free coupling between the sub components is the attractive qualities of a component. Cohesion
is the degree to which the task performed by a subsystem is connected. On the off chance that sub components are in
charge of various irrelevant tasks then the usefulness has been ineffectively circulated to the sub components. Hence
high cohesion is a normal for an all-around composed sub component (2). Stability is the extent to which the
component executes its task independently and measures how many number of implicit and explicit calls take place
in the component (8). In the following section, metrics have been proposed to measure the stability, coupling and
cohesion to predict the fault proneness and maintainability of the software system.

1. Coupling And Cohesion Metrics For Adaptive Reusability Risk Analysis

The successful identification of a Re(C) is generally adequate for reusability. The coupling and cohesion metrics have
to be evaluated to ensure its intended function with or without the help of the other components.

1.1 Dependency of the Component

The component dependence relationship comprises two divisions namely, Interdependency among two components
and Virtual component interdependency (Interdependency among multiple components).

2.1.1 Interdependency among two components

In order to maintain the coupling and cohesion level of the component, Anshu Maheshwari et al proposed different
metrics like, Lack of Cohesion of Methods (LCOM), Tight and Loose Class Cohesion (TCC & LCC) metric,
Afferent and Efferent coupling and Loose and Tight Coupling (1). But all those are only focusing on the following
interdependency among the components as shown in Figure 1.

![Figure 1 Interdependency among two components](a) (b) (c)

Uni- Directional Bi – Directional

Figure 1 Interdependency among two components

Where, C1 & C2 are refers to components.

The above Figure 1 illustrates two types of dependency relationship namely, unidirectional and bidirectional.
i. **Uni-Directional:** In Figure 1-(a) Component2 (C2) requires the output of component1 (C1) in order to fulfill its task completion as in Figure 1-(b) component1 (C1) requires the output of component2 (C2) for the completion of its task. This relationship looks like half duplex association between two components C1 and C2.

ii. **Bi-Directional:** In Figure 1-(c) Component1 (C1) requires the output of component2 (C2) as well as component2 (C2) requires the output of component1 (C1) for the completion of each task. This relationship looks like full-duplex association between two components C1 and C2.

### 2.1.2 Virtual component interdependency

In addition to these, it is significant to test the following interdependency among the multiple numbers of components as shown in Figure 2.

![Figure 2 Virtual component interdependency](image)

Where, C1, C2, C3 & C4 are refers to components.

This relation shows that, when C1 → C2, C2 → C3 & C3 → C4 then it is obvious that the dependency relation exists between C1 → C3, C4 and C2 → C4. Hence C3 and C4 virtually depend on Component C1 as like Component C4 virtually depends on Component C2 for its task completion.

This relationship can be proved by the Associative Property of Composition of function as shown in Equation 1(6).

If functions, f: C1 → C2, g: C2 → C3 and h: C3 → C4 then,

\[ h \cdot (g \cdot f) = (h \cdot g) \cdot f \]  \hspace{1cm} (1)

**Proof:**

Since f: C1 → C2 and g: C2 → C3, then g ∙ f: C1 → C3

Since g ∙ f: C1 → C3 and h: C3 → C4, then h ∙ (g ∙ f): C1 → C4

Now f: C1 → C2 and h ∙ g: C2 → C4

Hence (h ∙ g) ∙ f: C1 → C4

Thus, the domain and co-domain of (h ∙ (g ∙ f)) and those of ((h ∙ g) ∙ f) are the same.

Let Sc1, Sc2 & Sc3 are sub components of the reusable components C1, C2 & C3 that is, Sc1 ∈ C1, Sc2 ∈ C2 and Sc3 ∈ C3, so that, Sc2 = f(Sc1) and Sc3 = g(Sc2).

Then (g ∙ f)(Sc1) = g(f(Sc1)) = g(Sc2) = Sc3
Since \( h \cdot (g \cdot f)( S_{c1}) = h(S_{c3}) \) (2)

Also,
\[
{( h \cdot g) \cdot f} (S_{c1}) = (h \cdot g){f(S_{c1})} = h[g(f(S_{c1}))] = h(S_{c2}) = h(s3)
\] (3)

According to the Equations 2 and 3 it proves that the Equation 1.
\[
h \cdot (g \cdot f) = (h \cdot g) \cdot f
\]

Hence the relationship proved.

2. Metrics Based Evaluation For Reusability Adaptation

The assessment of coupling and cohesion metrics serves to recognize the level of reliance exist among the Re(C).

Coupling denotes the dependency of one component on any other component for its specified task. If the depended process is not complete, then the component cannot work.

Cohesion is the measure of individual strength and reliability of a particular component to perform its task independently. They are elaborated in the subsequent sections (9).

3.1 Coupling and Cohesion Metric Measurement for Examining Risk in Reusability

This metric calculates how effectively the Re(C) is dependent or independent on other components in the SRCR. Let us consider the following notations,

i. Let \( W_s \) – Whole System, \( C \) – Component , \( f \) – Function, \( S_f \) – Sub Function, \( N_c \) - Number of call

ii. The component ‘C’ have k-number of sub components \( (S_c) \) then, \( C = \{ S_{c1}, S_{c2}, …, S_{ck} \} \)

iii. The Sub component ‘S_c’ has n-number of functions (f), \( S_c = \{ f_1, f_2, f_3, …, f_n \} \)

iv. Each function (f) has m-number of sub functions (S_f), \( f = \{ S_{f1}, S_{f2}, S_{f3} , …, S_{fm} \} \)

v. Explicit Call - When, a component(C) call another component function (f) or sub function (S_f ) for its task completion is known as Explicit Call. It is denoted by \( E_{call}(C) \).

vi. Implicit Call - When, a component (C) called by another component function (f) or sub function (S_f) for calling component task completion is known as Implicit call. It is denoted by \( I_{call}(C) \).

vii. Total Call – The total number of call made either implicit or explicit within the component is called as Total call. It is denoted by \( T_{call} \).

\[
T_{call}(C) = \sum(I_{call}(C)) + \sum(E_{call}(C))
\]

2.1.1 Coupling and cohesion range metric for reusable component (CCR(C))

This metric calculates how effectively a component is coupled with another component in the SRCR. By considering above notations, the Coupling and cohesion range (CCR) metric of component is measured by CCR(C),
Coupled and Cohesion Range (CCR) metric for the whole system (W_s) is shown in CCR(W_s),

\[ CCR(W_s) = \frac{\sum_{i=1}^{k} CCR(C_i)}{k} \] (5)

Where, \( k \) = \( k^{th} \) Number of reusable components & \( i = 1...k \).

The whole system coupling and cohesion strength can be calculated by using Equation 5. The strength of coupling and cohesion for individual component can be measured by using the Equation 4.

The individual component strength using CCR metric is illustrated in Table 1.

**Table 1 Component coupling and cohesion range (CCR) metric value.**

<table>
<thead>
<tr>
<th>S.No</th>
<th>Explicit Call (E_{call})</th>
<th>Implicit Call (I_{call})</th>
<th>Component Coupling and cohesion range (CCR) metric value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0 (Tightly Coupled)</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1 (Loosely Coupled)</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0 (Loosely Coupled)</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1 (Tightly Coupled)</td>
</tr>
</tbody>
</table>

In Table 1, Using Explicit and Implicit call value determines the strength of the coupling between the Re(C). Here, if the component has explicit call value then it confirms that the component is tightly coupled with some other component or else it shows that the component is loosely coupled with other components.

Once the ‘coupling and cohesion’ value of the component is calculated and if it shows high cohesion and low coupling then it will be moved into the subsequent phase otherwise the component is rejected and the searching process is started anew.

**3.2 Measuring Instability (I_m) Metric**
This design metric finds that, the component is stable or not. A stable component will not affect the normal execution of the proposed software project after the Re(C) integrated with it. Only a stable component will provide a reliable output. Even any changes in the proposed system execution will not affect the stable component. But unstable component may lead to unexpected outcome of the new software system. Hence it is mandatory to measure the Re(C)’s stability level before taking up the next phases of assessment. After measuring coupling and cohesion strength of the Re(C), it is also necessary to measure the stability of the component.

Instability ($I_m$) of a component is measured using the values of Explicit call ($E_{call}(C)$) and Implicit call ($I_{call}(C)$). The Instability ($I_m$) can be measured by,

$$I_m = \frac{E_{call}\left(\sum_{i=1}^{k} C_i \in W_i, \sum_{j=1}^{l} C_j \in W_j\right)}{E_{call}\left(\sum_{i=1}^{k} C_i \in W_i, \sum_{j=1}^{l} C_j \in W_j\right) + I_{call}\left(\sum_{i=1}^{k} C_i \in W_i, \sum_{j=1}^{l} C_j \in W_j\right)}$$

$$= \begin{cases} 
1, & \text{Tightly - coupled (Instable)} \\
0, & \text{Loosely - coupled (Stable)} 
\end{cases}$$

Where, $k = k^{th}$ Number of reusable components & $i, j = 1..k$.

### 3.3 Integration with Present Software Domain

These analyses and test results would finally yield the much awaited Re(C) for integrating into the new software product. Successful implementation would incorporate the component with the recent product and function as a whole system. The complete flow of metrics measurement is illustrated in Figure 3.

The need of stability measure arises to confirm whether the new product is working normally as it should be. The new product may lead to certain errors which may not be predicted by the developer. The coupling, cohesion and stability test would eliminate the probability of errors before the product reaches to the customer.

![Figure 3: Metrics based measurement.](image_url)
If there is an occurrence of an error, the remedy is taken at once or the whole process is restarted in case of system failures. This test is further used to determine the risk impact level using the status produced by the outcomes of the test cases and the illustration of the normal distribution curve.

Finally the Re(C) is revised, reviewed and thoroughly verified to perform its expected function in the new environment with the least or no possibility of errors, satisfying the customer/end user. The point where this model gains advantage is in a complex program of thousands of lines of coding. Re(C) would perform the predefined function in the respective platforms with fewer changes in their discipline; conserve time and resources needed for creating anew.

3. Results And Discussion

The metric value of the proposed phase will reveal, how well the components are coupled with each other (Loosely coupled (or) Tightly coupled) and also measure the individual strength of each based on its stability value. In order to measure the stability, coupling and cohesion of the Re(C), consider the following ten numbers of C++ classes as components (3) as illustrated in Table 2 and those are organized in random. Measure those Instability, coupling and cohesion range metrics value by using proposed metric Equations 4 and 6. The final result will be noted in the following Tables 3 to 5 with proper graphical illustrations of each table using the Figures 4 to 6.

Table 2: Number of class components

<table>
<thead>
<tr>
<th>Class maximum</th>
<th>6. Class factorial</th>
</tr>
</thead>
<tbody>
<tr>
<td>`{ Public: int max(int x, int y) { } if(x&gt;y) return x; else return y; Other Statements; };</td>
<td>{ public: int fact(int n) { } if(n==1) return 1; else if(n&gt;1) return n*fact(n-1); else return n; Other Statements; };</td>
</tr>
<tr>
<td>2. Template&lt;class T&gt; class addition</td>
<td>7. Class display: public factorial</td>
</tr>
<tr>
<td>`{ public: void add() { T x,y; cin&gt;&gt;x&gt;&gt;y; return x+y; Other Statements; };</td>
<td>`{ public: void disp() { int n,f; cin&gt;&gt;n; f=fact(n); cout&lt;&lt;f; Other Statements; };</td>
</tr>
<tr>
<td>3. Class output : public addition,</td>
<td>8. Template&lt;class T&gt;</td>
</tr>
</tbody>
</table>
public maximum
{  
  Public:
  void show() {  
    {  
      int a,b,m;
      a=add(); cout<<a;
      b=add(); cout<<b;
      m=max(a,b);
      cout<<"maximum is"<<m;
    }
  }
  Other Statements;
};

class swaping
{  
  public:
  void swap(T x,T y) {  
    {  
      T t;
      t=x;
      x=y;
      y=t;
    }
  }
  Other Statements;
};

4. Template<class T>
class getdata
{  
  public:
  void getd() {  
    T x;
    cout<<"ente the value of x";
    cin>>x;
    return x;
  }
  Other Statements;
};

9. Class swapinteger: public getdata, public swaping
{  
  public:
  void swap() {  
    int a,b;
    a=getd();
    b=getd();
    cout<<"value of a is"<<a;
    cout<<"value of b is"<<b;
    swaping::swap(a,b);
    cout<<"value of a is"<<a;
    cout<<"value of b is"<<b;
  }
  Other Statements;
};

5.  Class searching : public getdata
{  
  public:
  void search() {  
    int a[5]={3,5,2,8,9};
    int b=getd();
    for(i=0;i<5;i++)
    {  
      if(a[i]==b)
        cout<<"element found";
      else
        cout<<"Element not found";
    }
  }
  Other Statements;
};

10. class square : public getdata
{  
  public:
  void area() {  
    int a;
    a=getd();
    cout<<"area of square is"<<a*a;
  }
  Other Statements;
};

According to the relation depicted in Table 2, the count of explicit and implicit call of each class is illustrated in Table 3.

Table-3: Coupling and cohesion range metric for class components.

<table>
<thead>
<tr>
<th>Class Number</th>
<th>Number of Explicit call</th>
<th>Number of Implicit call</th>
<th>Coupling and cohesion range (CCR) metric value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Coupled Level
Loosely coupled
Tightly Coupled
<table>
<thead>
<tr>
<th>Class Number</th>
<th>Number of Explicit call</th>
<th>Number of Implicit call</th>
<th>Instability ($I_m$) metric</th>
<th>Stability Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Stable</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Stable</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>Instable</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>Stable</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Instable</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>Stable</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Instable</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Stable</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>Instable</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Instable</td>
</tr>
</tbody>
</table>

The values of Table 4 will be illustrated in Figure 5.

Figure 4 Coupling and cohesion range (CCR) metric value

As per the values of number of explicit and implicit call for each class component the stability level will be measured in Table 4. The development team can judge whether these components will be reusable or non-reusable in the proposed software project by using this tabulated values.

Table-4: Instability ($I_m$) metric for class components.
Figure 5 Instability ($I_m$) metric value

According to the Coupling and cohesion range (CCR) metric value (0 or 1), Instability ($I_m$) metric value (1 or 0) in Tables 3 and 4, form Table 5. From this comparison, the development team can confirms that whether the specified Re(C) will be reusable (‘1’- Loose coupling and Stable) or not (‘0’- Tight coupling and Instable).

Table-5: Components reusable status.

<table>
<thead>
<tr>
<th>Class Number</th>
<th>CCR metric Value</th>
<th>Instability ($I_m$) Metric value</th>
<th>Reusability</th>
<th>Reusable status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Reusable</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Reusable</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Non-reusable</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Reusable</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Non-reusable</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Reusable</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Non-reusable</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Reusable</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Non-reusable</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Non-reusable</td>
</tr>
</tbody>
</table>

The values of Table 5 will be illustrated in Figure 6,
Metrics like coupling, cohesion and stability can also be used to reduce the complexity of the component. This promotes the confidence level of the professionals to develop the project within a short span of time. It also assures the product future development and maintenance.

5 Conclusion

Broadened perceptions and dynamics are central theme in this chapter as it focuses on the various metrics which measure the suitable Re(C) from phase-1 (SPRE Framework) of RRRA model. User specific needs are focused by the value of coupling and cohesion range metric component dependency range with the neighboring components. Fundamentally it will be measured and the Re(C)’s stability level will be confirmed by computing the value of the Instability metric. Astute selection of Re(C) is based upon the metric measurements observed during the evaluation process.

6 References