Abstract: The major advancement in the field of software engineering is Component-based software engineering. A set of independent components which work together are used to develop Component-based software system. A component is independent and reusable if it has low coupling and high cohesion values. The degree of interdependence between two or more components is coupling while the degree to which all elements of a component work together as a functional unit is cohesion. The coupling and cohesion value ranges from 0 to 1. The coupling and cohesion measures of a software component can be evaluated using design or source code of a component. The focus of this paper is to evaluate the coupling and cohesion measures of software components using UML diagrams.

Keywords: Coupling, Cohesion, Software Components, Evaluation of Software Component, Independent Component
1. Introduction:

In a component-based approach, a software component is deployed as an isolatable part of a system. The characteristic properties of a component are:

1. A component is a unit of independent deployment;
2. A component is a unit of third-party composition;
3. The component has no externally observable state.

From the above characterization, “A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties” [5].

A component that has low coupling and high cohesion value is independent and reusable. Cohesion is the glue that keeps modules of a component together. Coupling is the strength of the connection between components. A component with weak coupling and strong cohesion is:

1. Simple interface
2. Simpler communication
3. Simpler correctness proofs
4. Changes influence other components less often
5. Reusability increases
6. Comprehensibility improves [2].

The focus of this paper is to evaluate the coupling and cohesion measures of software components using UML diagrams after the design phase of the software development.

In 1997, the OMG (Object Management Group) developed the Unified Modeling Language (UML) as a common architectural framework for modeling object oriented systems and applications. The OMG described the Unified Modeling Language as a language representing unified best engineering practices for specifying, visualizing, constructing, and documenting the elements of business modeling, software and even non-software systems.
1.1. **Specification:**

UML can be used for specifying “what” is required of a system and “how” a system may be implemented. It captures the all-important requirements, analysis, design and implementation decisions that need to be established during a system development life cycle.

1.2. **Visualization:**

The graphical nature of UML allows the visualization of systems before they are implemented. Using shapes representing well defined semantics to communicate to a wider audience more succinctly than a descriptive narrative and more comprehensive than what often can be represented by a programming language.

1.3. **Construction:**

UML can be used to guide and craft the implementation of a complicated system. Furthermore, with the aid of various case tools on the market, it is possible to generate object oriented source code from UML models and also possible to reverse engineer source code into UML Models.

1.4. **Documenting:**

UML offers a means of capturing knowledge and documenting deliverables such as requirements documents, functional specifications and test plans. These are all critical in controlling, measuring and communicating a system throughout its lifecycle [1].

This paper uses only the use case model and analysis model of UML to evaluate the coupling and cohesion of software components.

The Use Case Model describes the proposed functionality of the new system. A Use Case represents a discrete unit of interaction between an actor (human or machine) and the system. An actor is a human or a machine entity that interacts with the system to perform meaningful work [4].
The analysis objects model instantiates the Entity-Control-Boundary Pattern (ECB). The ECB is a simplification of the Model-View-Controller Pattern. ECB partitions the system into three types of classes: entities, controls, and boundaries.

**Entity Class** – persistent data.

**Boundary Class** – interface between the actor and the system.

**Control Class** – encapsulates business functionalities.

Entity, control, and boundary are official UML class stereotypes. UML has some special icons to represent them [6]:

![Diagram](image)

- **USE CASE**
- **ACTOR**
2. Related Work:

2.1. Jarallah S. Alghamdi Coupling Metric:

Jarallah S. Alghamdi [3] proposes a coupling metric that has two steps to find the coupling of a component. The first step is to generate a description matrix that captures the factors that affect coupling in a system. The second step is to calculate the coupling between each two components of the system from the description matrix to produce a coupling matrix.

2.1.1. Generation Of The Description Matrix:

The objective of generating a description matrix is to create a structure that captures all of the characteristics of a software system that relate to coupling, which can then be used to calculate coupling information for that system. The description matrix, shown in Table 2, is a m components by n members matrix. Each component of the software system is represented by a row of the description matrix; these are classes in an object-oriented system, or functions, procedures and subroutines in a procedural system. The columns of the description
matrix represent members, which are methods and instance variables in an object-oriented system, or variables and parameters in a procedural system.

<table>
<thead>
<tr>
<th>Component \ Element</th>
<th>$E_1$</th>
<th>$E_2$</th>
<th>...</th>
<th>$E_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$</td>
<td>$d_{11}$</td>
<td>$d_{12}$</td>
<td>$d_{1n}$</td>
<td></td>
</tr>
<tr>
<td>$C_2$</td>
<td>$d_{21}$</td>
<td>$d_{22}$</td>
<td>$d_{2n}$</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_m$</td>
<td>$d_{m1}$</td>
<td>$d_{m2}$</td>
<td>$d_{mn}$</td>
<td></td>
</tr>
</tbody>
</table>

*Table 2: The Description Matrix*

Each entry $d_{ij}$ in the description matrix shows the weight of a member $E_j$ with respect to component $C_i$. The greater the value of $d_{ij}$, the more influence member $E_j$ will have on the coupling value between components $C_i$ and $C_k$, when $d_{xj} > 0$. A zero value for $d_{ij}$ indicates that $E_j$ is inaccessible from $C_i$.

2.1.2. Generation of the Coupling Matrix:

The coupling matrix for a software system of $m$ components is a matrix of order $m \times m$, where each row and each column represents a component of the system, as shown in Table 3. The value of each entry $C_{ij}$ of the coupling matrix indicates the extent to which the row component, $C_i$, is coupled to the column component, $C_j$. The values can be calculated from the description matrix in various ways; for example, the degree of coupling between two components can be the sum of the weights of all members shared by the two components.

<table>
<thead>
<tr>
<th>Component</th>
<th>$C_1$</th>
<th>$C_2$</th>
<th>...</th>
<th>$C_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$</td>
<td>$C_{11}$</td>
<td>$C_{12}$</td>
<td>$C_{1m}$</td>
<td></td>
</tr>
<tr>
<td>$C_2$</td>
<td>$C_{21}$</td>
<td>$C_{22}$</td>
<td>$C_{2m}$</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_m$</td>
<td>$C_{m1}$</td>
<td>$C_{m2}$</td>
<td>$C_{mn}$</td>
<td></td>
</tr>
</tbody>
</table>

*Table 3: The Coupling Matrix*

The formula used to calculate each entry $C_{ij}$ of the coupling matrix is:

$$C_{ij} = \left( \sum_{k=1}^{n} d_{ik} \times d_{jk} \times \beta_k \right) \alpha_i$$

Where,
dik and dkj d are entries of the description matrix,

αi is the sum of the entries of the ith row of the description matrix, and

βk is the reciprocal of the sum of the kth column of the description matrix.

Mathematically they are:

\[ \alpha_i = \sum_{j=1}^{n} d_{ij} \quad \text{and} \quad \beta_k = \frac{1}{\sum_{i=1}^{n} d_{ik}} \]

From the coupling properties the coupling of a component and its system is calculated using the following metric:

The coupling value of component

\[ C_i = 1 - C_{ii} \]

The overall coupling of the system

\[ C_z = \sum_{i=1}^{n} C_i. \]

3. Evaluation Of Coupling And Cohesion Measure:

To find the coupling and cohesion measures two matrices have been used.

1. Use case matrix
2. Similarity matrix

3.1. Use Case Matrix:

The use case matrix is formed by extracting the use case relationship from the use case model. The use case matrix is shown in Table 4.
The use case matrix can take the value either 1 or 0. The actor takes a value 1 if it is related to
the use case; otherwise it takes a value 0. Similarly Entity takes a value 1 if it is related to the
use case; otherwise it takes a value 0.

### 3.2. Similarity Matrix:

The similarity matrix is generated from the use case matrix. The similarity matrix is shown in
Table 5.

<table>
<thead>
<tr>
<th>USECASES</th>
<th>USECASE 1</th>
<th>USECASE 2</th>
<th>...</th>
<th>USECASE n</th>
</tr>
</thead>
<tbody>
<tr>
<td>USECASE 1</td>
<td>USECASE 11</td>
<td>USECASE 12</td>
<td>...</td>
<td>USECASE 1n</td>
</tr>
<tr>
<td>USECASE 2</td>
<td>USECASE 21</td>
<td>USECASE 22</td>
<td>...</td>
<td>USECASE 2n</td>
</tr>
<tr>
<td>USECASE n</td>
<td>USECASE n1</td>
<td>USECASE n2</td>
<td>...</td>
<td>USECASE nn</td>
</tr>
</tbody>
</table>

The similarity matrix is generated from the use case matrix using the following metric:
Similarity \((\text{USECASE}_i, \text{USECASE}_j)\) = \((n_{11}+n_{00})/(n_{11}+n_{00}+n_{10}+n_{01})\)

Where,

\(n_{11}\) = number of 1 present in both \(\text{USECASE}_i\) and \(\text{USECASE}_j\)

\(n_{10}\) = number of 1 present in \(\text{USECASE}_i\) but not in \(\text{USECASE}_j\)

\(n_{01}\) = number of 1 present in \(\text{USECASE}_j\) but not in \(\text{USECASE}_i\)

\(n_{00}\) = number of 1 not present in both \(\text{USECASE}_i\) and \(\text{USECASE}_j\).

3.3. **Cohesion**:  
The cohesion of the software and its components are calculated using the following metrics:

\[
\text{Cohesion of Component}_C = \frac{\sum \text{Similarity (USECASE}_i, \text{USECASE}_j)}{m_C}
\]

Where,

\(m_C\) = number of use cases in the Component\(C\).

\[
\text{Cohesion of Software} = \frac{\sum_{C=1}^{n} \text{Cohesion of Component}_C}{n}
\]

Where,

\(n\) = number of components.

3.4. **Coupling**:  
The coupling of software and its components are calculated using the following metrics:

\[
\text{Coupling of a Component}_C = \frac{\text{Component dependency}}{\text{Number of Components} - 1}
\]
Component dependency takes the value 1 if the ComponentC depends on any other component otherwise it takes the value 0. Dependency is based on the Use Case model relationships include, extend and generalization.

\[
\sum_{C=1}^{n} (\text{Coupling of a Component}_C)^2
\]

\[
\text{Coupling of Software} = \frac{\text{Number of Use Cases}}{\text{Number of Use Cases}}
\]

3.5. Independent Component:

The component which has low coupling value and high cohesion value is said to be the Independent Component.

4. An Example:

In this section an example is given to evaluate the coupling and cohesion metrics. The example taken here is a simple ATM use case diagram.

4.1. ATM Use Case Diagram:

![Figure 1: ATM Use Case Diagram]
4.2. ATM Analysis Model:

![ATM Analysis Model Diagram]

*Figure 2: ATM Analysis Model*

4.3. Components Segregation:

From the ATM use case diagram the components are segregated as below:

<table>
<thead>
<tr>
<th>Component 1</th>
<th>Component 2</th>
<th>Component 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Withdraw Money</td>
<td>Validate Pin</td>
<td>Refill Machine</td>
</tr>
<tr>
<td>Check Balance</td>
<td>Print Receipt</td>
<td></td>
</tr>
</tbody>
</table>

*Figure 3: Component Segregation*
4.4. Use Case Matrix:

From the above details the use case matrix is formed. The input parameters from the above diagrams to create a use case matrix are:

1. Number of Use Cases i.e. 5.
2. Number of Actors i.e. 3.
3. Number of Entities i.e. 2.
4. Number of Components i.e. 3.

<table>
<thead>
<tr>
<th>Actors\Entities</th>
<th>Card Holder</th>
<th>Bank Server</th>
<th>Maintenance Person</th>
<th>Customer</th>
<th>Account</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use Cases</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Withdraw Money</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Validate Pin</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Check Balance</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Print Receipt</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Refill Machine</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

*Table 6: ATM Use Case Matrix*

The above ATM use case matrix based on the following relation:

1. Withdraw Money, Card Holder takes the value 1 since the use case Withdraw Money is related to the actor Card Holder.
2. Withdraw Money, Bank Server takes the value 1 since the use case Withdraw Money is related to the actor Bank Server.
3. Withdraw Money, Maintenance Person takes the value 0 since the use case Withdraw Money is not related to the actor Maintenance Person.
4. Withdraw Money, Customer takes the value 0 since the use case Withdraw Money is not related to the entity Card Holder.
5. Withdraw Money, Account takes the value 1 since the use case Withdraw Money is related to the entity Card Holder.

The relation rule used for the use case Withdraw Money is used to all the use cases to create the use case matrix.
4.5. Similarity Matrix:

The similarity matrix is generated from the use case matrix as below:

<table>
<thead>
<tr>
<th>Use Cases</th>
<th>Withdraw Money</th>
<th>Validate Pin</th>
<th>Check Balance</th>
<th>Print Receipt</th>
<th>Refill Machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Withdraw Money</td>
<td>1</td>
<td>0.6</td>
<td>1</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>Validate Pin</td>
<td>0.6</td>
<td>1</td>
<td>0.6</td>
<td>0.6</td>
<td>0.2</td>
</tr>
<tr>
<td>Check Balance</td>
<td>1</td>
<td>0.6</td>
<td>1</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>Print Receipt</td>
<td>1</td>
<td>0.6</td>
<td>1</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>Refill Machine</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 7: ATM Similarity Matrix

The above similarity matrix is generated using the following metric:

\[
\text{Similarity (USECASE}_i, \text{USECASE}_j) = \frac{(n_{11}+n_{00})}{(n_{11}+n_{00}+n_{10}+n_{01})}
\]

Where,

- \(n_{11}\) = number of 1 present in both USECASE\(_i\) and USECASE\(_j\)
- \(n_{10}\) = number of 1 present in USECASE\(_i\) but not in USECASE\(_j\)
- \(n_{01}\) = number of 1 present in USECASE\(_j\) but not in USECASE\(_i\)
- \(n_{00}\) = number of 1 not present in both USECASE\(_i\) and USECASE\(_j\).

The similarity between the use case Withdraw money and other use cases is calculated as follows:

1. Similarity (Withdraw Money, Validate Pin) = \(\frac{(2+1)}{(2+1+1+1)} = 0.6\)
2. Similarity (Withdraw Money, Check Balance) = \(\frac{(3+2)}{(3+2+0+0)} = 1\)
3. Similarity (Withdraw Money, Print Receipt) = \(\frac{(3+2)}{(3+2+0+0)} = 1\)
4. Similarity (Withdraw Money, Refill Machine) = \( \frac{(0+1)}{(0+1+3+1)} = 0.2 \)

Similarly the similarity between other use cases has been calculated to generate the similarity matrix.

**4.6. Cohesion:**

**4.6.1. Cohesion of Components:**

From the similarity matrix the cohesion of software components is calculated using the following metric:

\[
\text{Cohesion of Component}_C = \frac{\sum \text{Similarity}(\text{USECASE}_i, \text{USECASE}_j)}{m_C}
\]

\( \forall \text{USECASE}_i, \text{USECASE}_j \in \text{Component}_C \)

Where,

\( m_C = \) number of use cases in the Component\( C \).

Using the above metric cohesion of software components has been calculated as below:

- Cohesion of Component 1 = \( \frac{1}{2} = 0.5 \)
- Cohesion of Component 2 = \( \frac{0.6}{2} = 0.3 \)
- Cohesion of Component 3 = 1

The number of the use cases in the component 1 is 2 that are Withdraw Money and Check Balance (Figure 3). The similarity between the Withdraw Money and Check Balance is 1. Therefore the cohesion of component 1 is \( \frac{1}{2} \) that is equal to 0.5. Similarly, other components have also been calculated.

**4.6.2. Cohesion Of Software:**

From the cohesion measures of software components, the cohesion of software is calculated using following metric:
∑_{c=1}^{n} \text{Cohesion of Component}_c

\text{Cohesion of Software} = \frac{\sum_{c=1}^{n} \text{Cohesion of Component}_c}{n}

Where,

n = number of components.

Using the above metric the cohesion of software has been calculated as below:

Software Cohesion = \frac{(0.5+0.3+1)}{3} = 0.6

The number of components in the ATM system is 3 and the cohesion of component 1 is 0.5, component 2 is 0.3 and component 3 is 1. Therefore cohesion of software is \frac{(0.5+0.3+1)}{3} that is equal to 0.6.

4.7. Coupling:

4.7.1. Coupling Of Components:

The coupling of components is calculated using the following metric:

Component dependency

Coupling of a Component = \frac{\text{Component dependency}}{\text{Number of Components} - 1}

Using the above metric, the coupling of software components has been calculated as below:

Coupling of Component 1 = \frac{1}{(3-1)} = 0.5

Coupling of Component 2 = \frac{0}{(3-1)} = 0

Coupling of Component 3 = \frac{0}{(3-1)} = 0

The number of components is 3 and the component 1 is depend on component 2 by the relation <<include>> (Figure 3) hence component dependency takes a value 1. Therefore coupling of component 1 is 1/(3-1) that is equal to 0.5. The component 2 and 3 are not depending on any other components, hence its component dependency is 0 and coupling measures are also 0.
4.7.2. Coupling Of Software:

From the coupling measures of software components, coupling of software is calculated using the following metric:

\[
\text{Coupling of Software} = \sqrt{\frac{\sum_{c=1}^{n} (\text{Coupling of a Component}c)^2}{\text{Number of Use Cases}}}
\]

Using the above metric, the coupling of software has been calculated as below:

\[
\text{Software Coupling} = \sqrt{(0.5*0.5+0*0+0*0)/5} = 0.223607
\]

The coupling of component 1, 2 and 3 are 0.5, 0 and 0. The numbers of use cases are the 5 (Figure 1). Therefore coupling of software is \( \sqrt{(0.5*0.5+0*0+0*0)/5} \) that is equal to 0.223607.

4.8. Independent Component:

In the above example the independent component is Component 3 because it has the low coupling value (0) and high cohesion value (1) among the three components.

5. Conclusion:

This paper describes the coupling and cohesion metrics to evaluate the software components and the software. Based on the above coupling and cohesion metrics, a system to evaluate coupling and cohesion of software components has been created. This system results the coupling and cohesion measures of software components and the software. This system also identifies the independent component.
References: