CHAPTER 1:- INTRODUCTION
According to Pressman, Software engineering process has been depended on three layers. These layers are following:

1. Methods
2. Process
3. Tools

Software process keeps the technology layers together and as sort timely develops the complex or large software applications. A method includes the analysis of requirements, architectural design, component design, coding, software testing, maintenance and deployments. Software tools give the support directly or indirectly to the software engineering process and the methods [Pressman S. R. (2006)].

![Figure 1.1 Relationships between Layers](image-url)
In the beginning of 40’s to 60’s, we have not available appropriate software development process or modules in which we can make a large or complex software application. That time layman person could not understand nor easily readable the programming languages programmed. So programmer thoughts to move towards into new approach which we known as assemble language. In this language programmer had used some numeric code for easily understanding or readable the program. But this language had also not comported for developing the complex software system.

Researcher has shown in the few last year, that the object-oriented programming (OOP) technology itself is not comfort to develop the complex software system and does not support the rapidly changing requirements of the applications. The reasons are the OOP methods enhance to develop complex models that means the object of problem domain.

Today software applications are large and very difficult to developments, are not compatible with previous version of software applications. So due to this, it has been packaged encompasses with a variety of advantages or features. This feature cannot be eliminate but upgraded the software system application separately or replaced.

The main features of CBSD are produce the high quality of software and reduced the development time. The benefits of CBSD are to produce the quality software using reusability, increase productivity, compatibility and flexibility. So provide these types of benefits, multiple organization used this approach because using this technology reduced the development time and cost of the software. Researcher used the reusable software functionality to develop the complex software system. The main aim of CBSE using the preexisting software component to develop, upgrade or design the software system.

Fred Brooks stated that, for development the software system used this philosophy “the ‘buy, don’t build’”. He speaks regarding the CBSE that “in the same way that early subroutines liberated the programmer from thinking about details, CBSE shifts the focuses from coding to composing software systems”. The CBSD is an approach for reusing the whole software component, not objects.
Software reuse is very active and challenging field of software engineering. The major functionality of software reuse is to enhance the software productivity, reduce the development time, reduce the cost and improve the quality of software. Because we use reusable software components which are already tested. The main benefits of reusable software component are ease maintenance, reduce the development time and improve the reliability. We have available many metrics in which we can easily identify the reliability of software system. These metrics are

- MTBF [Mean Time Between Failure]
- MTTR [Mean Time to Repair]
- ROCOF [Rate of Occurrence of Failure]
- MTBM [Mean Time Between Maintenance]
- Availability

Figure 1.2 shows an X-Model which represents the development of components with reusable software components.
Figure 1.2 X Model [Tomar P. (2013)]
1.1 EVOLUTION OF CBSE

1.1.1 Traditional Approach

Software reuse has been begun between 50’s to 60’s. That time libraries concept was rapidly used in software market. This concept stated that to allow the integration of reusable subroutine and pre-complied to be interconnection a program for executing this function.

Structured systems development began in the 1960s with the concept of systems development life cycle. Organizations continue to use traditional approach for software development, but they are coming to realize that this approach has shortcomings and these shortcomings are:

- They take no account of evolutionary change [Pressman S. R. (2006)].
- They impose a separation of data from the operations on the data.
- Modifications to the legacy system are more difficult to implement because of the nature of the implemented software.
- They do not encourage reusability.

Traditional software development methodologies suggest developing software from scratch rather than reusing existing code [Pressman S. R. (2006)]. So due to these shortcomings, developers think to shift this approach to another approach for developing the software.
Figure 1.3 Analysis Model and Design Models
1.1.2 Object-Oriented Approach

The OO approach has evolved in the starting of 80’s. OO approach does not provide the interoperability facility of software component at the run time level. Object-Oriented Programming has some discrimination in which OO approach is not completely optimal for develop a complex or large software system because it does not fully support to produce the reusable components. [Lewis T. et. al. (1995)].

Figure 1.4 Object-Oriented Applications
1.1.3 Component-Based Approach

Figure 1.5 CBD Maturity Phases
That time component-based software engineering approach has not evolved but COTS component market is rapidly growing. So researcher has thoughts to move a new technology which known as CBSE. CBD develops the complex software system using reusable software components for improve the reliability, reusability and productivity. [Chappell D. (1997)].

Figure 1.6 Evolution of CBSE
Better understanding of software evolution, especially for CBSE. To analysis the issues of CBSD in individual companies as well as in the IT-industry at large:

- Modern trends in CBSD technology enable main advantages of software reuse. These benefits include improvements in productivity, reliability, quality, effort (cost) and time-to-market, and standards [Sommerville (2010)].

- Software evolution is inevitable in any software system since changes in society and technology will require subsequent changes to software systems to keep them up to date [Vliet V. (2008)]. Moreover, efficiency in the software process is paramount due to the ever-increasing demand on available development capacity.

- A related theme is the increased usage of COTS components and Open Source Software in new development. These are different characteristics than in-house developed non-reusable components due to e.g. vendor control, selection and integration issues [Li (2004)]. Their impact on the development and maintenance processes is therefore also different.

To understand CBSE in detail on the software component/software system and software process levels in order to develop solutions to these issues:

- Software Process Improvement (SPI) [Aaen (2001)], i.e. systematically incorporating these solutions as part of revised development practices, is key. This will enable software engineers, software designers, software architects and the like to improve their cost-effectiveness in developing quality software based on reusable components. It will also enable them to improve their ability to develop and use reusable components, such as code and process models.

- Risks management strategy is closely correlated to the above-mentioned issues in CBSE [Boehm (1991)]. Knowledge of the impact and effectiveness of risk management strategy in CBSE-driven software evolution is paramount to the success of the key points mentioned above.
Software architecture design is a central pillar of any software applications [Bass (2004)], and is also an important concern seen in our investigation. We must therefore pay close attention to the design, maintenance and evolution of the architecture, to secure the continued success of the system. Awareness of potential architectural evolution risks is important as architectural changes can permeate a software system.

On the one hand, earlier investigations in risk management have commonly focused on risks and risk management strategies on the project level. Software architecture investigations commonly study the architecture design, component design, coding and maintenance of the software system [Babar (2007)].

1.2 COMPONENT-BASED DEVELOPMENT VERSUS OBJECT-ORIENTED DEVELOPMENT

The difference between CBD and OOD is going to discuss but before to understand the differences between components and objects.

<table>
<thead>
<tr>
<th>Factor</th>
<th>OOD</th>
<th>CBD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reusability</td>
<td>Development-time</td>
<td>Run-time</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Less flexible</td>
<td>Increase flexibility regarding hardware and software</td>
</tr>
<tr>
<td>Interoperability</td>
<td>Development has restricted with one technology</td>
<td>Provides communication between different technologies</td>
</tr>
<tr>
<td>Deployment</td>
<td>Monolithic software Application</td>
<td>Independent components of software</td>
</tr>
<tr>
<td>Building Strategy</td>
<td>Inheritance and composition are both used</td>
<td>Composition is greater and Inheritance is not necessary</td>
</tr>
</tbody>
</table>

*Table 1.1 Comparisons between OOD and CBD*
1.3 CBSE PROCESS MODEL

In CBSE has different type of process models but in this thesis, we are discussed only main two process models.

1.3.1 Stojanovic Process Model

This model has emphasized on the concept of component from requirements analysis to implementation. Theses phases’ requirement analysis, architectural design, implementation, deployment and maintenance are support by requirements service, identification of components, specification of component, assembly of component and deployment of component. And the decision of components should be made to build components, purchase using COTS software components rather than scratch development [Stojanovic Z.].

![Figure 1.7 Stojanovic Process Model](image-url)
1.3.2 COSE Process Model

According to Dogru and Tanik has stated that CBSD methodology is not mature because researcher reported that software development methodologies has started the traditional development technology that includes the software development waterfall model and then moved into object-oriented technology.

![Diagram of COSE Process Model]

**Figure 1.8 COSE Process Model**

COSE model categorized into four important phases and integrated with software system phase:

- System decomposition
- System specification
- Component Specification, modification, search, creation
- Integration

Software System Specification | Decomposition | Component Specification, Search, Modification, Creation | Integration | Software System
1.4 PROBLEM OUTLINES

A wide range of algorithms can aid in increasing reusability, testability, correctness and maintainability in the code to evaluate and select a component in CBSD. Component Selection should be selecting the optimal component from component repository. CBSD paradigm requires an altogether different approach from functional development techniques. As a considerable portion of CBSD, selection process of component face so many problems - excess budgets, deliver software with poor quality, testable component with not good interface and documentation, poor testable and reusable component repository. Moving towards this direction, this study is a small step to analyze, design, and develop new component selection process and algorithm for selection of appropriate component to solve all these problems.

1.5 OBJECTIVES

In this thesis, component selection is considered as valuable factor to enhance the software quality and reduces the development and testing cost. So this thesis focuses on how to select the optimal software component from component repository.

- To design a new process for component selection of software component.
- To development a new algorithm for selection of software component
- To implement the new algorithm is optimal for component selection from component repository