Abstract—As the time of developing the software applications is to be reduced, software reuse is the need of the hour. For reuse, the components may be developed in house using components based software engineering (CBSE) or they may be commercial off the shelf components (COTS) developed by the third parties. This paper presents a comparison of how the software development varies in using these two types of paradigms of software reuse.

Key words — CBSE, COTS based development, certification, testing, risk.

I. INTRODUCTION

Software reuse is very useful for software engineers. It allows to reuse many artifacts in all phases of software development. These include software requirement document, plans, design, code, test cases, test data, test plans etc. The piece of code which is reusable in called a software component. The benefits of software reuse are improved quality, increased productivity and shorter time to market. Though savings in cost is required from software reuse but software development cost may be more than development from scratch initially but may prove beneficial if that application or development is reused in future. So the emphasis is on high quality and savings in time to develop the application. Thus there are two approaches which can be used for software reuse. These are component based software engineering and commercial off the shelf based software development. Component-based software engineering (CBSE) is a process that emphasizes the design and construction of computer-based systems using reusable software “components” [2]. It is concerned with the development of systems from software components, the development of components, and system maintenance and improvement by means of component replacement or customization [1]. The component may be developed in one application and may be reused in another similar application. This can be stored in the software repository of the organization for reuse. An organization who wants to sell the software components for reuse can also develop these as independent components and made available for purchase. These types of components are called commercial off the shelf components i.e. COTS components. Formally, an item that is commercially available, leased, licensed, or sold to the general public and which requires no special modification or maintenance over its life cycle is a COTS component [3]. One example of COTS solution is Enterprise Resource Planning software. Some of the major examples of ERP are SAP, PeopleSoft, and Oracle Applications. Now COTS become the new “silver bullet” of software engineering [4], [5]. The next section discusses the two approaches.

II. PHASES OF CBSE

In CBSE process, the software team examines the requirements to determine what subset is directly amenable to composition, rather than construction. For each requirement, the team will ask:

- Are internally developed reusable components available to implement the requirements?
- Are the interfaces for available components compatible within the architecture of the system to be built?

The team will attempt to modify or remove those system requirements that cannot be implemented with in-house components. The CBSE process can select the components which may be approximately matching or exactly matching the requirements. This is because the changes to the code can be made by the developer. The CBSE process identifies not only candidate components but also qualifies each component’s interface, adapts components to remove architectural mismatches, assembles into selected architectural style. Various phases in CBSE are:

1) Requirements Analysis: The user requirements as well as developer requirements are analyzed during this phase and specific design objectives are provided e.g. understandability, learnability, supportiveness, flexibility etc.

2) Design with reuse: After the analysis of the requirements, a design with reuse is to be developed. The design is modified according to the software components available in the software repository for reuse. The software components may be matching exactly according to the requirements or it may be an approximate match.
3) Software Repository: After analysis of the requirements we search/find the software components in the repository so that we can make a design with the reusable software components. Reusable software artifacts like software requirement document and design etc can be put in the software repository for future reuse.

4) Coding: After the design is finalized, some modifications may be done in the existing components as per the requirements for reuse. The code is developed from scratch for the part of application not available as reusable components.

5) Integration: Integration means to integrate all new components with the old components and develop the current application at hand.

6) Testing: After the integration of the components testing is done to check the correctness of application. This is the last phase of the process, after that the new component (application) ready for use(or reuse). It is shown in fig 1.

**Figure 1 : Phases in CBSE**

The primary goal of developing COTS components based system involves mapping the functionality that are offered by COTS components onto the requirements of users. For each requirement, the team will ask:
- Are commercial off-the-shelf (COTS) components available to implement the requirements?

The team will attempt to modify or remove those system requirements that cannot be implemented with COTS components. The developer here should select the COTS with exact match only because he does not have access to the code. Also the mismatch in requirements may lead to failure. Additional services provided by the COTS component may prove to be risky as they may consume more resources and may conflict with other requirements of the software application being developed. Traditional requirements engineering models are inappropriate for formulating requirements for COTS based systems [6]. Although it is generally acknowledged that good requirements engineering is essential for successful component-based system development [7], methods adequately address the problem of how requirements for component-based systems are derived. The COREx approach [7] is part of the COMPOSE—COMPonent-Oriented Software Engineering—process that we have been developing. A summary of the process is that requirements are elicited in parallel with assessing the availability of potentially suitable COTS components. Requirements can be negotiated according to available functionality and, where appropriate, traded to achieve an optimal configuration. In some cases, it may not be possible to identify COTS components to deliver what is deemed to be critical functionality in which case, to achieve the functionality might require custom development [8]. The phases during COTS based software development are:

1) Requirements Analysis: Requirements analysis is critical to the success of software, if the requirements are not clear, then it is very difficult to develop the correct software. The requirements of the software should be documented, measurable, testable, actionable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

2) Design: During the design phase, the system is designed to satisfy the requirements identified in the previous phase. In COTS based software development we are seeking for exact match in the repository. If it is found, then we purchase COTS components and complete the design including this.

3) Coding: In this phase we do basically coding of part of the application which was not found as reusable component. Also if some conflicts in the COTS components are identified , then glue code/wrapper code is written to remove the conflicts between components and reuse them.
4) Integration: Integration is the process of combining all the components into one system and ensuring that the system’s function running properly.

5) Testing: To verify that the software meets the user's requirements and the system specifications. Fig 2 shown below shows these phases:

![Figure 2: Phases in COTS based Software Development](image)

### IV. CBSE VERSUS COTS BASED DEVELOPMENT

Both CBSE and COTS based software development may seem to be requiring same efforts and cost for software development but this is not the case. There are many criteria which can make these software components based development processes difficult to develop, maintain and further reuse. The direct and indirect costs may also prove the process selected to be costly for the development of current application. So it is important to compare the two processes. A number of criteria are used for differentiating between CBSE and COTS based development. Table 1 summarizes these differences:

<table>
<thead>
<tr>
<th>Criteria</th>
<th>CBSE</th>
<th>COTS Based Development</th>
</tr>
</thead>
<tbody>
<tr>
<td>Place of development</td>
<td>In-house development</td>
<td>Developed by third parties</td>
</tr>
<tr>
<td>Maintenance</td>
<td>Responsibility of system developer</td>
<td>Vendors maintains COTS components</td>
</tr>
<tr>
<td>Type of components</td>
<td>Design, SRS, test plans, test data, executable, source code</td>
<td>Only executables</td>
</tr>
<tr>
<td>Type of reuse permitted</td>
<td>White box reuse</td>
<td>Black box reuse</td>
</tr>
<tr>
<td>Integration efforts</td>
<td>High due to source code</td>
<td>Low ; only through interfaces because of executables</td>
</tr>
<tr>
<td>Benefits to Developer</td>
<td>Developer acquires source code</td>
<td>System developer acquires the services</td>
</tr>
<tr>
<td>Shared costs</td>
<td>Only development costs are shared between developer and reuser</td>
<td>Both development and maintenance costs are shared between vendor and reuser</td>
</tr>
<tr>
<td>Retrieval criteria</td>
<td>Retrieved by exact or approximate</td>
<td>Exact retrieval criteria</td>
</tr>
<tr>
<td>Certification</td>
<td>No need of certification</td>
<td>Certification is required</td>
</tr>
<tr>
<td>Usability</td>
<td>Simple to reuse</td>
<td>May lack of integrity and simplicity</td>
</tr>
<tr>
<td>Risks</td>
<td>Less risk of reusing, support within organization</td>
<td>High risk is involved when vendor may go out of business</td>
</tr>
<tr>
<td>Testing</td>
<td>Little testing is required</td>
<td>Rigorous testing is required</td>
</tr>
<tr>
<td>Verification and Validation</td>
<td>Verification and Validation is done in-house</td>
<td>Verification and validation by vendors and re-users</td>
</tr>
<tr>
<td>Cost of future reuse</td>
<td>Less as may be reused as and when required</td>
<td>More as again may need purchasing or maintenance cost</td>
</tr>
</tbody>
</table>

© 2012, IJARCSSE All Rights Reserved
From the above table of comparison it is clear that COTS based software development may prove to be costly as well as risky. This is due to the involvement of vendor from which the COTS component is to be purchased. The dependency on vendor for maintenance of the COTS component again adds a difficulty for reuse in future. The risk of vendor going out of business can surely waste all the investments in reuse. These costs and risks are not with the components developed in-house. However, there are many attributes of a COTS component which may prompt us to use the COTS based software development. It provides us the direct executables, which are thoroughly tested and are of high quality. With less integration cost these can be simply added in the required place instead of developing it from scratch. The development time of software application may be greatly reduced with COTS based development but future reuse may be limited which is not a limitation with in-house developed software components.

VI. CONCLUSION

The life cycle phases of CBSE and COTS based software development are discussed in this paper in a simplified way. Many parameters are identified for comparing the two modes of software development with reuse. The paper highlights the benefits and limitations of COTS based software development versus CBSE. These can be useful for the software developers who want to develop the current application with reuse.

REFERENCES