A Component-Based Framework for Software Reusability

Adnan Khan, Khalid Khan, Muhammad Amir and M. N. A. Khan
Shaheed Zulfikar Ali Bhutto Institute of Science and Technology,
Islamabad, Pakistan
adnan_online47@yahoo.com, khalid.sarhadi@gmail.com,
amir_amirkhan94@yahoo.com, mnak2010@gmail.com

Abstract

Component-based development allows us to develop and integrate product components which facilitate software reusability, high quality and simplification for testing. Component-Based Software Engineering makes use of approaches which are based on architecture definition languages, object oriented design and software architecture. These strategies assist in the development of both domain-specific and generic software products. Reusability approach speeds up software development by using already developed components, thus software development cost and time is substantially reduced. In this paper, we present a Component-Based Software Engineering framework for software reusability.

Keywords: Component-Based Software Engineering, Reusability, Component-Based Software Architectures, Software Component Models, Service Oriented Software Engineering, Software Development Life Cycle

1. Introduction

Developing cost-effective and quality products is an important and challenging aspect of software development. Component-based software development can help developers to efficiently produce software within the time and budget constraints. The concept of component-based software engineering (CBSE) is based on the development of independent and loosely coupled components of the system, by avoiding irrelevant dependency among system components. CBSE focuses on linkage among different components in a way that one component can provide services to another through different interfaces. This way of development of normalize components facilitates rapid software development.

The field of software systems has become increasingly intricate and performance specific. To produce cost-effective systems, organizations often make use of component-based technologies as an alternative to developing the whole system from scratch. The objective behind using component-based technologies is to reduce the cost associated with the development. Nevertheless, this field later evolved into a more critical area to reduce dependence on the current market and to match rapidly emerging consumer requirements. Currently, the usage of component-based technologies is more frequently motivated towards curtailing the development costs. More functionality can be produced by employing this technology and with lesser investment of time and money [1]. However, novel issues pertaining to dynamic configuration and scalability crop up when new components are introduced into a system. Some issues are usually addressed using CBSE. It provides approaches, prototypes and principle for the programmers who are somewhat associated with the component-based techniques. Component-based development (CBD) emphasizes on improving the techniques by
creating significant components of utilization. Although, CBSE research area has been there for the last two decades, but still several issues remain unresolved. Several CBSE solutions have been achieved by utilizing rules and techniques from different design procedures, including component-based techniques.

CBSE is a recognized field of software engineering. Its strategies and approaches are based on architecture definition languages (ADLs), middleware, object-oriented design, software architectures. Nevertheless, the nature of software is different from industrial products. Therefore, an immediate interpretation involving rules on the conventional architectural disciplines into software architectural is not achievable. For instance, understanding the component is not an issue in the conventional architectural disciplines because a component is generally intuitively realized and fits efficiently with the fundamental concepts/principles as well as engineering design and model. However, the same situation is not with the software components. A common definition states, “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 party” [2].

Many prototypes depend on specific platforms such as Enterprise Java Beans, DCOM, Microsoft .Net etc. The CBS-specific risks are created as features of the selected components do not completely fulfill system requirements to provide more features that are undesirable in a given system [3]. The particular connection among components performs a vital role throughout the requirements analysis. Adapters and wrappers are the earliest part of integration methods.

CBSE is considered by two development processes: the development of reuse components and integrated components. The ability to measure quality attributes of CBSE facilitate better understanding, assessment and control the individual risk of the whole software development lifecycle. Component producers are involved using implementation, design and maintenance of the components while component consumers are interested in certain components. CBSE metric approaches not only measure quality of the component, but also include managing the ambiguity descriptions and appropriate mathematical property that can fail quality metrics.

Information technology is facing enormous challenges such as high demands to meet the product deadlines with minimum development time and cost. Reusability approach of the software component is required to optimize software development cost and time. Many software organizations are adopting CBSE methodologies in order to meet requirements of the customers to deliver products at competitive cost. Incorporating the existing components in the software development process can improve software productivity. Similarly, employing state of the art tools are also very useful in this scenario. In addition, risk associated with the software products can be reduced as the use of latest software and other domain-specific tools increases the possibility of identifying and fixing errors. CBD approaches are composed of implementing a component into a system through a set of well defined interfaces. These components interact with different components to perform a function of the system. However, the internal framework of the components and implementation of the interfaces are always hidden to the outside world.

2. Literature Review

CBSE is implemented on reusable components as per their functionality, behavior and interfaces. Zhang et al. [4] proposed a research for CBSE based on local projects. The authors proposed a solution for CBSE and validated Trustee components system by practicing Cushion and Eclipse tools. The two strategies proposed by the authors
include component-based software engineering and service oriented software development. Open Services Gateway Initiative (OSGI) provides an interface for CBSE and Software Oriented Architecture (SOA).

Chouambe et al. [5] focus on reverse engineering software models of component-based systems. The authors argue that a large number of software systems use component-based technologies such as Common Object Request Broker Architecture (CORBA), Enterprise JavaBeans (EJB) and Component Object Model (COM). However, there is a lack of efficient tools and service in order to reversing such techniques. Current approaches employed for such purposes mostly depend on components of reverse engineering, but these approaches do not resolve external dependencies. In addition, dependent and existing components description is also used to interchange classes and modules. Developers who want to use iterative reverse engineering strategies and approaches need to apply component-based software architectures and compare different tools, methodologies, approaches and techniques.

Crnkovic et al. [2] looked into different software component-based models which were developed by employing diverse themes, objectives, principles and technologies. All the models produced similar results but the principles they followed were different. While some models do not explain the concept clearly; yet they help identify, characterize and provide easy to understand concept of component-based model framework.

The majority of component-based model utilize high level programming development. In such cases, some component-based models execute a specific language for implementation that requires some specific predefined rules. Enterprise Java Beans component-based model uses Java with some additional requirements for implementation and some component-based model use translator for their specific language or multiple languages such as Corba component model. The main advantage of CBSE is that it is independent for the development process of individual components. These processes work completely independent such as development of COTS (Commercial Off-the-Shelf) and COTS-based systems. Until other aspects of the component are integrated into a system, some specific stages are involved in the development of an individual component such as deployment, implementation, execution, requirements and design.

Hunt and McGregor [6] argue that CBSE is a technique for developing as well as assembling system from recent components containing essential implications for assorted computer software engineering practices. CBSE approaches developed from time to time and this area of research is rapidly evolving to solve fundamental difficulties, problems and the convenience connected with CBSE. If CBSE course is included in the curriculum, then students would get the opportunity to gain practical knowledge of applying the CBSE techniques despite limitations of the existing software tools.

Abdellatief et al. [7] studied how to integrate CBSS with different components to deploy them independently. Though researchers used several proposed CBSE attributes, nevertheless, implementing the CBSE metrics practically is a difficult task because some metrics either overlap with other metrics or are not well defined. The authors described the interface complexity metrics for parameters and methods in the interface. The proposed collection of metrics is exclusively meant for reusability component of JavaBeans evaluation.

Software development depends on three major aspects which have a direct impact on software business such as time, cost and quality of product. Khan et al. [8] argue that
information technology is facing enormous challenges such as customer demands to
meet the product deadlines with minimum development time and cost. Reusability
approach of the software component could be beneficial in this regards to optimize
software development cost and time. Many software organizations are adopting CBSE
methodologies in order to meet the requirements of the customers to deliver the product
at a very low cost as well as to reduce the development time. Moreover, the already
existing components in the software development process can also improve software
productivity to a greater extent. Similarly, employing state of the art tools are also very
useful in those scenarios where software development cost and time are required to be
kept to minimum. In addition, risk associated with the software products can be reduced
as use of the latest software and other domain-specific tools increases the possibility of
identifying and fixing errors at the earlier stages of the software development process.
Component-based development approaches composed of implementing a component
into a system through its well defined interfaces. These components interact with
different components to perform the overall function of the system.

Lubair and Moiz [9] highlight the main goal of CBSD by describing that the
component-based technology changed from a simple component to the domain-specific
components over a period of time. Though the impact of time and cost is still a
challenging issue in the domain-specific components, the reusable design is among the
main advantages of CBSE as the reusable components save time and cost investment.
Component-based development models provide software reusability approach which is
highly useful to the software engineers. QSM Associates report mentions that 84%
decrease in project costs has been noticed due to the growing use of reusable
components in the software development process.

The development of model-driven engineering prototypes could have many positive
impacts on the overall performance of the projects. Particularly, the use of different
traditional methodologies and approaches for embedded system could be useful in this
regard. Reusable components are also used for quality improvement in CBSE. Bunse et
al. [10] investigated valuation of CBSD approaches and model-driven prototypes in the
construction of embedded systems. The authors used Marmot method for micro-
controller subsystem of automotive. Development efforts and size of the model were
compared and measured by applying two main methods, namely, agile software
development and unified process.

Breivold and Larsson [11] describe that CBSE and service oriented software
engineering (SOSE) are the most important paradigms adopted by the software
development organizations. Both the approaches are used for similar methodologies in
many senses, but their focus is different. The advantages of both the approaches are to
improve quality of attributes in software engineering and develop the software projects
rapidly. However, there could be escalating analysis required to comprehend and
implement the combined possibilities of both the models. Essential areas of research
within SOSE consist of service composition, service oriented engineering, monitoring
and service foundation (which is support service oriented technologies to provide run
time service oriented infrastructure), and perform connectivity to heterogeneous
systems.

CBSD is used to provide software development model with particular reference to
robustness, reliability and reusability of software components and assembling it into
appropriate software architecture. In case these components cannot perform properly, it
will impact on software quality and functionality [12]. All the testers are required to be
involved throughout the entire SDLC (software development life cycle) to reduced
defects and errors rates. Before implementing the procedures, the process adopted for system development needs to be approved through user acceptance, beta and alpha testing.

Koziolek [13] highlights the performance of evaluation measurement and prediction approaches for CBSS to evaluate the system component specification designed by component developers. Additionally, integrating conventional performance designs like stochastic process algebras, queuing networks or stochastic Petri nets techniques are used to benefit from the advantages of CBSE such as reusability. Even though several techniques and approaches have been proposed in this area during the last decade, yet they did not receive widespread business use. The functionality of reusable software components is complicated to specify because performance not only depends upon component implementation but also on the deployed context of the component. The performance of software component includes certain influence factors such as required services, usage profile, deployment platform, resource contention and component implementation.

Kahtan et al. [14] describe the future security challenges of the CBSD which concentrate on developing models by combining current software components. CBSD offers an enhanced capability to reuse existing components to develop high quality software at a lower cost and lesser time. Utilizing CBSD models in software engineering poses several challenges. During the integration phase, the software components can result in producing many issues, therefore, security features of the components analyzed in the previous CBSD lifecycle such as maintainability, reliability, safety, integrity and dependability must be considered to check the software. Such an approach results in providing good customer services as well as enabling system modification from time to time.

CBSE offers a large range of functionality throughout the development of a software system. Iqbal et al. [15] describe that CBSE focus on developing software system by reusing high quality of independent software components. With the help of OOT (Object Oriented Technology), reusable components have become key aspects of software development. The components modification is used in software development process which increases reusability approach at different levels such as at framework level, architecture level and modular design level. The development process of CBSE modifies the reusability approach into two different approaches, generation-based approach and composition-based approach, which are quite beneficial when programming components are reused.


Khan et al. [25, 26] proposed a machine learning approaches for post-event timeline reconstruction. Khan [27] suggests that Bayesian techniques are more promising than other conventional machine learning techniques for timeline reconstruction. Rafique and Khan [28] explored various methods, practices and tools being used for static and live digital forensics. In [29], Bashir and Khan discuss triaging methodologies being used for live digital forensic analysis.
3. Critical Evaluation

In this section we provide a critical review of the CBSE techniques discussed in Section 2 (see Table 1.).

<table>
<thead>
<tr>
<th>Sr #</th>
<th>Focused Area</th>
<th>Tools and Techniques</th>
<th>Merits</th>
<th>Demerits</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>Software Component Model</td>
<td>Component Composition, Software Component Models, Component Lifecycle, Extra-functional Properties.</td>
<td>Component based model is developed for object oriented development and ADLs.</td>
<td>The proposed approach is used for a single model.</td>
</tr>
<tr>
<td>[2]</td>
<td>Prioritizing Component Based System</td>
<td>Component Integration Graph, Use Case Scenario.</td>
<td>A new technique to prioritize the component based model using CIG (cert in guidance).</td>
<td>The use case scenario is not helpful for complex procedures.</td>
</tr>
<tr>
<td>[3]</td>
<td>Cluster-based Component Systems</td>
<td>Matching Index, Signed Graph Clustering, Handling Dependencies.</td>
<td>Selection process of CBS is helpful to analyze the model interdependencies.</td>
<td>Proposed framework is based on lengthy and complex procedures.</td>
</tr>
<tr>
<td>[4]</td>
<td>Component Based Metrics Systems</td>
<td>Software Metrics, Component based Quality Systems.</td>
<td>For CBS, the software metrics system is a new approach to create CBS models.</td>
<td>To mapping mechanism for analyzing the CBS is not a perfect approach.</td>
</tr>
<tr>
<td>[5]</td>
<td>CBS Development Model</td>
<td>Software process. CBD, CBSD Models.</td>
<td>The traditional software process approach for CBSD models enhances reusability of CBS models.</td>
<td>Various models have been used to analyze a single process.</td>
</tr>
<tr>
<td>[7]</td>
<td>Testing in CBSE</td>
<td>Software Quality, CBSD, Functional Testing.</td>
<td>Testers are required to be involved throughout the entire SDLC to reduce defects and error rates.</td>
<td>Lack of guidance for testing system as a whole.</td>
</tr>
<tr>
<td>[10]</td>
<td>CBSE Process Modification</td>
<td>CBSE, Object-Oriented Programming (OOP), Domain Engineering, Java API library, Aspect-Oriented Programming (AOP), Interface Description Language (IDL), Information Hiding.</td>
<td>The components modification are used in software development process which increase reusability approach at different levels such as at framework level, architecture level and modular design level.</td>
<td>Lack of clear objective for distributed environment.</td>
</tr>
</tbody>
</table>

4. Proposed Framework for Reusability of CBSE

In this section, we propose a reusability framework for CBSE. The proposed framework addresses various aspects of reusability artifacts developed earlier. The framework is divided into two phases. In the first phase, we propose to store the newly developed components into a repository for future use. The second phase deals with searching and analyzing the stored components for reuse. The architecture of the proposed framework is provided in Figure 1.
Figure 1. Proposed Reusability Framework for CBSE

The detailed description of the framework is provided as under.

4.1. Software Development

Software development entails programming/coding which is the procedure of writing and keeping the source code. In a wider sense, it is concerned between idea of the preferred software through to the final manifestation of the software, usually in a organized and structured procedure. Software development may include modification, prototyping, research, maintenance, new development, code reuse, reengineering, or any other actions that result in software product.

4.2. Component Development

Software development pertains to development of individual components that offer specific functionality as per the software architecture. However, such functionality is integrated with the other components.

4.3. Reusability Domain Analysis

Domain analysis is the generic priority of software engineering and software reusability method is the key aspect of domain analysis for software recycling. Reusability domain analysis entails using different methodologies such as generic architectures, feature platforms,
feature tables, facet templates and domain specific languages which describe all of the
systems in a domain.

4.4. Reusable Tagging

Reusable tagging is created dynamically which tags artifacts in the ascending order to cater
for large amount of tools and techniques to support the expanding requirement for content
sharing, management and component creation. Reusable tagging techniques are used to share
all reusable information to the user and collaboratively interpret content such as images,
URLs and other information.

4.5. Reusable Component Repository

Reusable component repositories had been produced to be able to minimize difficulties due
to artifact growth as well as to get rid of the need for as a standalone file storage devices
solutions due to the contingency deployment involving varied storage devices technologies
running diverse operating systems. These will characteristic centralized operations for file
storage devices. In repository, we store information about reusable component such as unique
ID number, artifact description, type of artifact (i.e., software or design or architecture),
component name, physical path and programming language/tools used to develop it.

**Table 2. Reusable Component Repository**

<table>
<thead>
<tr>
<th>ID</th>
<th>Component Name/Path</th>
<th>Software/Design/Architecture</th>
<th>Description</th>
<th>Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Login Form</td>
<td>Design</td>
<td>User Registration Form</td>
<td>Vb.net</td>
</tr>
<tr>
<td>2</td>
<td>Searching Modules</td>
<td>Software</td>
<td>List of modules offered in a semester subject</td>
<td>Asp.net</td>
</tr>
<tr>
<td>3</td>
<td>CORBA</td>
<td>Architecture</td>
<td>Include support for FORTRAN programmers</td>
<td>Java</td>
</tr>
<tr>
<td>4</td>
<td>Data Adapter</td>
<td>Architecture</td>
<td>Object to connect to the database</td>
<td>Oracle10g</td>
</tr>
</tbody>
</table>

4.6. Search Engine Reusable / Artifact

Initiative for software reusability is better applied when there is an effective method to find
the reusable resources. However, retrieval and searching of such information is considered an
issue inside literature, as soon as we have a gap among precisely what the software engineer
would want to access along with what is located inside repository.

4.7. Component Integration

The interaction between software components is carried out after testing the component.
These components are specified by different groups in different time. The integration of all of
the items must work together. It is important to cover pessimistic cases as well because
component might create predictions according to the data.
4.8. System Validation

System validation is the procedure of ensuring that system does exactly what it is meant to do in a regular and reproducible approach. The approval procedure starts with the software requirements in a meaningful way and is carried out until system is abandoned.

5. Case Study based Validation

To validate our model, we used cases studies already published in the contemporary literature. We used two cases studies: one published by McCarey et al. [13] and the other by Matsumura et al. [14]. Both the case studies were proposed to evaluate reusable components in a software organization. The purpose of case study based validation is to determine the usefulness and relevance of the suggested procedures. The main aim of this research is to see how the existing components can be used in conjunction with the other components to develop large units of software system that bear tremendous commercial perspective. In this regard, our focus was to evaluate the cost, time, quality, reduced learning curve and knowledge sharing aspects of the reusability.

McCarey et al. [13] describe collaborative filtering strategy that allows developer using reusable components to achieve main objective of the on-demand learning; thus, improve quality of developer productivity at low cost and reduced development time. The case study proposed by McCarey et al. [13] mainly focuses on three aspects as mentioned earlier, whereas, our proposed model also takes into account two additional factors which include reducing the efforts required by the software designers and developers to learn working with new technologies as well as sharing the knowledge among the software development teams. We argue that reusability can reduce the learning curve time by providing solutions to the developers in the form of code snippets, design documents and other artifacts.

Matsumura et al. [14] focused on the quality of productivity during integration testing of the software reusable components. The occurrence of bugs was reduced about 20% to 30% in addition to reducing size of the reusable components which is of a great help for the developers. In this regard, we suggest using necessary documentation and search tags with the reusable components as it would be beneficial to determine what sort of customization would be required in the reusable component to make it work with the new requirement. Also, we suggest storing only those components into the reusable repository which are bug free so that their reuse could not result in compromising the software quality.

6. Conclusion

The reusability of CBSE is a common approach to improve quality, reliability and productivity as well as reduced time and cost for development of new software. The impact of time and cost is still a challenging issue in the domain-specific components. The reusable design is among the main advantages of CBSE as the reusable components save time and cost investments. In this research, we have reviewed CBSE-based methodologies which are dependent on different available approaches and techniques aimed at reusing different components. This study also compares different techniques and model-based approaches which can help understand different components of software engineering. Also, we look into how components are integrated to encourage reusability and reduce cohesion e.g., implementing a loosely coupled system where components are independent of each other.
References

