International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-2 Issue-1, March 2013

Component Based Development in Software Engineering

Amandeep Bakshi, Rupinder Singh

Abstract--In today's world, Component Based development is an active research area for more than a decade in software engineering. As they provide automated or semi-automated support for the various processes and the methods. There are three main approaches in Software Engineering world such as Structured, Object-oriented and Component-based approach. The last Component-based approach introduces more benefits to this world in terms of reusability, flexibility and maintainability. This paper basically provides the background of various software engineering approaches and compares the Component based development to the object oriented development approach and to the various other traditional approaches.

Index Term– Component Based Development, Commercial off the shelf (COTS), Object-oriented Development, Process Models.

I. INTRODUCTION

Software engineering is defined as the application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software that is the application of engineering to software. Software engineering process is the glue that holds the technology layers together and enables rational and timely development of computer software [1].

The history of software engineering begins from traditional approach that is the first methodology in the software world. It is meant that software development using a set of mature and stable technologies, which often include mainframe-based technologies. Then object-oriented approach came into this world and brought many new useful features. It covered most of the aspects of software development, including testing and project management. And the last one is-like a return to the correct mean of engineering- component-based approach. One of the essential characteristics of engineering disciplines is to build a product by assembling pre-made, standard components. The component-based approach is the most recent. Right now, component-based development (CBD) is in the leading edge phase. Indeed, there are now a number of technologies appropriate for and people with experience in the application of CBD [2].

Revised Manuscript Received on 30 March 2013.
* Correspondence Author
Amandeep Bakshi, Computer Science & Engineering Department, Thapar University, Patiala, India.
Rupinder Singh, Computer Science & Engineering Department, Thapar University, Patiala, India.

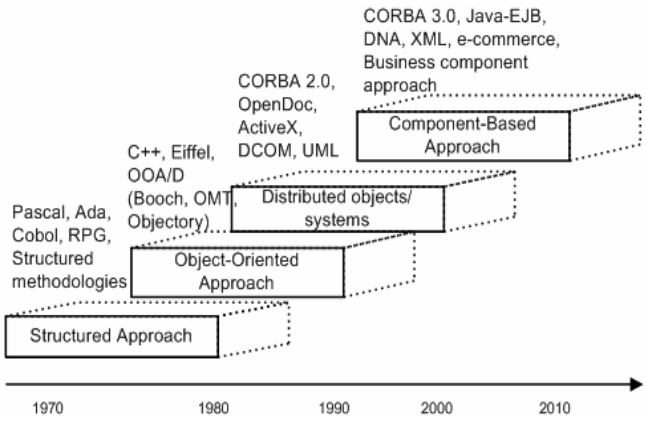

Figure 1- The evolution to components in the industry [2].

One of the main contributions that CBD has is the reuse of software components in multiple systems. In this way a software component is developed only once, and can save out development effort multiple times. This approach towards saving effort does of course also work for other (intermediate) products that are made in the course of software development. An approach puts this idea central is called reuse-based software engineering [3].

II. VARIOUS APPROACHES FOR SOFTWARE ENGINEERING

There are three major approaches in this world such as structured, object-oriented, and component-based [5].

A. Structured Approach

This approach contains basic steps of a software development process such as analysis, design, implementation, testing, and maintenance. This report focuses on only analysis and design steps and their models. The design phase produces a data design, an architectural design, an interface design, and a procedural design with the help of various methods and techniques [5].

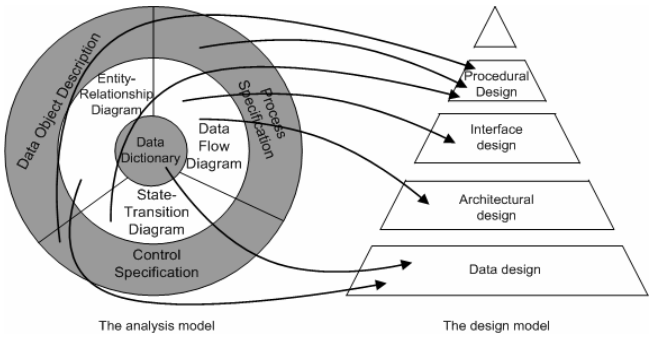

Figure 2- Analysis and design model for structured approach [1].

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

B. Object-Oriented Approach

Object-oriented approach promises a way for implementing real-world problems to abstractions from which software can be developed effectively. It is a sensible strategy to transform the development of a large, complex super-system into the development of a set of less complicated sub-systems. Object-orientation also aims to provide a mechanism to support the reuse of program code, design, and analysis models. Therefore, analysis and design phases are inevitable for object-oriented approach, as well. In this approach, design is divided into four different steps as illustrated in Table 1 [5].

Table1- Analysis and design phases for object-oriented approach[5].

Phase	Techniques	Key Deliverables
Analysis	Collaboration Diagrams Class and Object Models Analysis Modeling	Analysis Models
System Design	Deployment Modeling Component Modeling Package Modeling Architectural Modeling	Overview design and implementation architecture
Class Design	Class and Object Modeling Interaction Modeling State Modeling Design Patterns	Design Models
Interface Design	Class and Object Modeling Interaction Modeling State Modeling Package Modeling Prototyping Design Patterns	Design Models with interface specification
Data Management Design	Class and Object Modeling Interaction Modeling State Modeling Package Modeling Design Patterns	Design Models with database specification

In structured approach, the end user would receive a monolithic application. When it is developed using object-oriented approach, the end user would still receive a monolithic application. Shortly, the object-oriented approach is the service of the functional developer, not the end user [5].

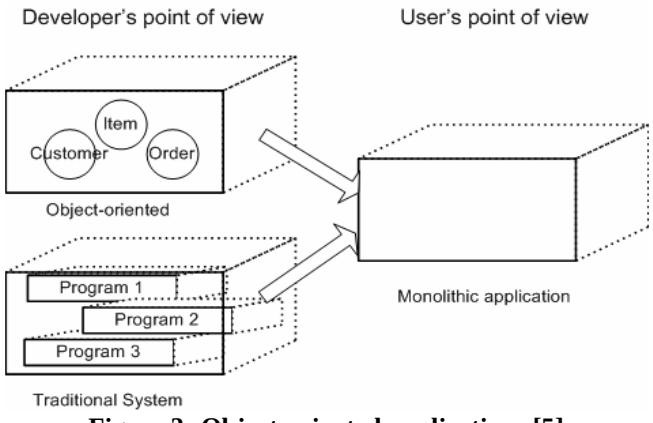

Figure 3- Object-oriented applications [5].

C. Component Based Approach

Component-based development is a CBSE activity that occurs in parallel with domain Engineering. Once the architecture has been established, it must be populated by components that (1) are available from reuse libraries and/or (2) are engineered to meet custom needs. Hence, the task flow for component-based development has two parallel paths.

When reusable components are available for potential integration into the architecture, they must be qualified and adapted. When new components are required, they must be engineered. The resultant components are then "composed" (integrated) into the architecture template and tested thoroughly [1]. The resulting increase in reuse should dramatically improve time-to-market, software lifecycle costs, and quality [5].

The *distributed object approach* extends the object-oriented approach with the ability to call objects across address space boundaries, typically using an "object request broker" capability. The *distributed system approach* means a development approach for building systems that are distributed, are often multi-tier [5]. Figure 4 depicts the transformation that occurs after object oriented approach.

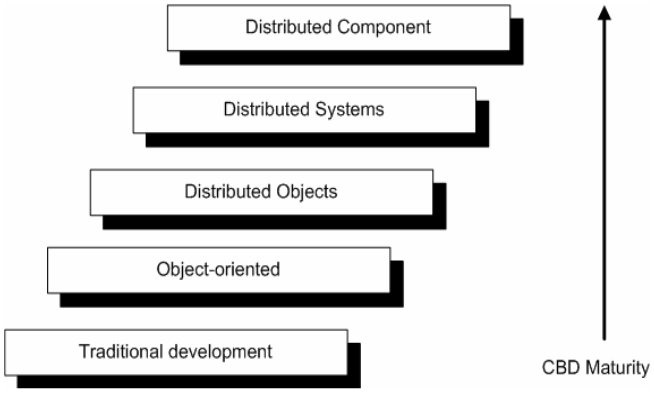

Figure 4- CBD maturity phases [5].

III. VARIOUS PROCESS MODELS

There are various process models called life cycle, which are developed for different circumstances. Traditional process models are the most mature ones, and mostly suit structured approach. However, they are also used for recent approaches such as object-oriented and component-based approaches in various phases and in somehow modified form. Object-oriented approach does not offer any original process model, which provides for its needs completely. Therefore, this approach complies with traditional models and their appropriate combinations. However, it is different when it comes to component-based approach because it changes the nature of software [5].

A. Traditional Process Models

A number of different life-cycle models are used and three of them are most widely used i.e. waterfall with iteration, rapid prototyping and the spiral model.

B. Object-Oriented Process Models

The critical sense of object-oriented models is iteration, which is inevitable. One of the most favored models is *fountain model*, which contains circles, defining overlapping phases. In addition, there are arrows, describing the iteration within those phases. In this model, maintenance is reduced considerably. Besides fountain, there are other models, which can be considered within object oriented approach. These models are iterative and, incorporate some form of parallelism (overlap of activities), and support incremental development. One essential problem with these types of models, that is team members moves almost randomly between phases due to parallelism [6].

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

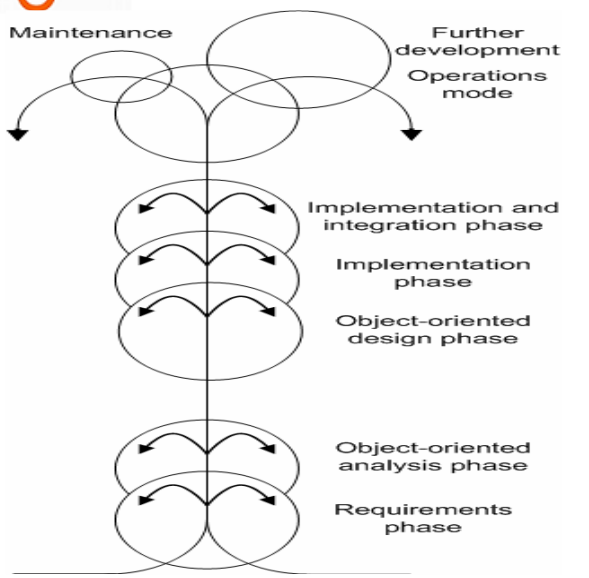

Figure 5- Fountain model [6].

C. *Component Based Stojanovic Process Model*

A component-oriented development process model, shown in Figure 6, has been introduced by Stojanovic, focusing on the component concept from business requirements to implementation. This process will be called by its owner's name. The phases of requirements, analysis, design and implementation in a traditional development process has been substituted by service requirements, component identification, component specification, component assembly and deployment. After the components of the system are fully specified, a decision can be made to build components, wrap existing assets, buy COTS (Commercial Off-the-Shelf) components or invoke web services over the Internet [5].

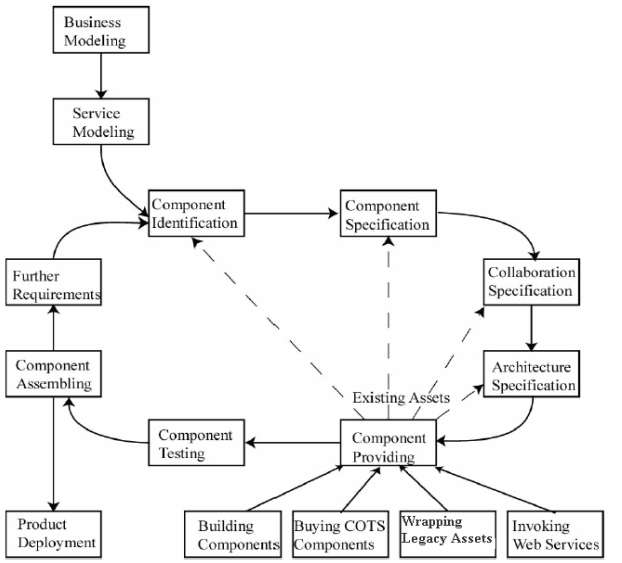

Figure 6 - Stojanovic Process Model [5].

D. *Component Based COSE Process Model*

COSE Process Model is an activity which starts from top-down to introduce the building blocks of the system. As the activity continues towards lower granularity blocks, interfaces between the blocks are also defined. At an arrived level where the module is expected to correspond to a component, a temporary bottom-up approach can be taken, if desired capability can only be achieved by a set of components, their integration into a super-component should be carried out.

COSE process model consists of four main phases and a system test phase:

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

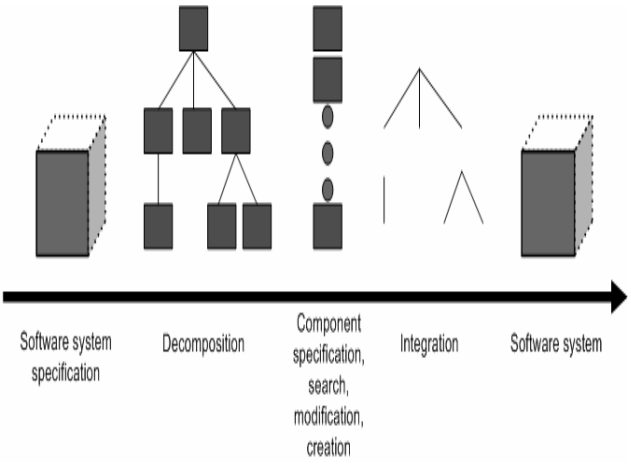

Figure 7- COSE process [5].

IV. CBD VS OBJECT-ORIENTED DEVELOPMENT (OOD)

The main difference between a component and an object is that, a component is meant to be a runtime entity, whereas an object is an instance of a class. Objects are usually not thread-secure, because the designer knows (or thinks he knows) how the objects are going to be used. In a component context, he cannot be sure about that, and therefore the components have to be secured. To build complex units, objects can also use inheritance, whereas when components build complex units, they are limited to composition. In other words, a developer connects several components to make a super component**.** Components are deployed independently, and it is impossible to predict how the component is going to be used. As a result, concerns about dependencies between components, call backs, components using each other - but belonging to different threads, safety and security, and so on - are much more important than they are in plain object-oriented modeling or design [5].

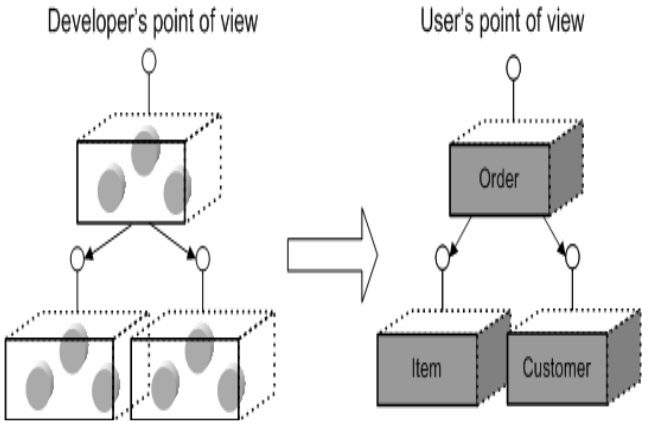

Figure 7- Component-based approaches [5].

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

Table2- A comparison of CBD and OOD [5]

Attributes	CBD	OOD
Interoperability	Provides communication between different technologies	Development is restricted with one technology on one platform
Reusability	Run-time	Development-time
Flexibility	More flexible in terms of hardware and software	Less flexible
Reliability	Thread safe and secure	Dependent on developers' ability
Deployment	Independent parts of software	Monolithic software application
Building strategy	Composition is major Inheritance is unnecessary	Inheritance and composition are both used

V. CBD VS TRADITIONAL DEVELOPMENT

In traditional approaches, a modification request may conceptually affect only one module, but implementation of the modification usually ends up with impacting several modules. Furthermore, in many cases the whole application needs to be redeployed as shown in Figure 8 [5].

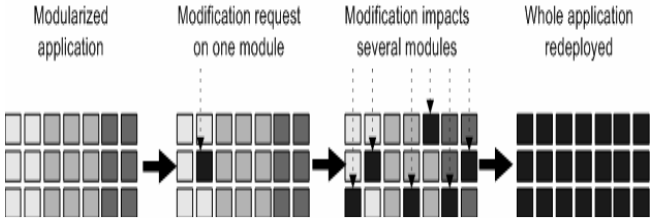

Figure 8- Traditional deployment method [5].

Using a component-based approach, the same cycle becomes as illustrated in Figure 9. In this case, the modification usually impacts only the component that is affected by the modification request, and it is possible to redeploy only the impacted component. This benefit is particularly important for large systems.

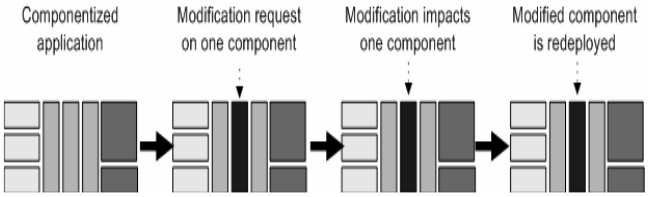

Figure 9- Deployment method in component-based approach [5].

VI. COTS VS CBD

Commercial off the shelf (COTS) based software development is the process of building software application from commercially available software components. A COTS product is an executable software product has the following characteristics [3].

- It is sold, leased or licensed to the general public.
- Buyers, lessees and licenses have no access to the source code; hence can only use the product as a black box.
- It is offered by a vendor who has created it and is typically responsible for its maintenance and its upgrades.
- It is available in multiple identical copies (with in the same version) on the market.

There are a number of advantages to use COTS products in software development [3].

- Gain in Cost
- Gain in Operational Quality
- Gain in Functionality
- Gain in Time to market.
- Gain in Maintenance Overhead

To make use of existing COTS components and existing integration technologies, the system designer often follows a bottom up approach [3].

Lifecycle follows sequence of phases:

1. Selection from wide range of candidate assets.
2. Integration in the host system.
3. Verification and validation of the COTS product.
4. Maintenance of the COTS product.

COTS development is a special case of CBSD

Table3- Comparison between COTS and CBSD [3].

Key areas	COTS	CBSD
White Box versus Black Box	Black Box	White Box
Copy right Privileges	Vendor	System/user Developer
Maintenance	Vendor	Developer
Retrieval Criteria	Exact Match	Approximate Match

VII. CONCLUSION

In software Industry for developing an application, mainly three process approaches and models are used i.e. structured approach, object oriented approach and component based approach. Out of these the component based approach and CBSE process model is the best because it help us to divide the application into two parallel activities i.e. component based development and domain engineering which helps to develop a components and reuse the old components from COTS for developing the new applications . By using CBD, we can easily develop components .CBD then qualifies, adapts and integrate these components for use in a new system. It also support integration and composition and COTS helps for selecting the component for reuse. So, as a result, CBSE process improves quality and productivity and hence reduces development schedule and effort. Now days two new process models are used i.e. Stojanovic Process Model and COSE process Model on the basis of component based approach. These Models helps to redeploy the whole application.

VIII. FUTURE SCOPE

CBD approach is especially sensitive for safety-critical domains, real-time systems, different process-control systems where the demands on reliability are higher than in a "standard" web based application. One biggest problem with CBD is low control of quality requirements of component-based systems and extensive work must be done in that direction.

CBSE is facing many challenges today, some of these are summarized in the following:-

- Trusted components
- Component certification
- Composition predictability

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

- Requirements management properly
- Long-term management of component-based systems
- Component configurations
- Tool support

REFERENCES

1. PRESSMAN Roger, "Software Engineering", McGraw Hill, 2006.
2. HERZUM Peter, SIMS Oliver, "Business Component Factory", Wiley, 1999.
3. Mili, H, Mili ,A.,Vacoub S., and Addy, E(2002) "Reuse Based Software Engineering", Wiley- Interscience Publication, USA.
4. G. Pour, "Component-Based Software Development Approach: New Opportunities and Challenges," Proceedings Technology of Object-Oriented Languages, 1998. TOOLS 26., pp. 375-383.
5. BAYAR Vedat," A Process Model For Component Oriented Software Development", Master Thesis, 2001.
6. RUMBAUGH James, BLAHA Michael, PREMERLANI William, EDDY Frederick, LORENSEN William, "Object Oriented Modeling and Design", Prentice Hall, 1991.
7. Somerville, Ian," Software Engineering", 7th Edition, Addison-Wesley, 2004.

AUTHOR PROFILE

Amandeep Bakshi is Post Graduate student in Computer Science & Engineering Department, Thapar University, Patiala, India and pursuing Master of Engineering in Software Engineering. He holds Bachelor of Technology (B.Tech) degree in Information Technology from Punjab Technical University, I.E.T. Bhaddal, Ropar, Punjab, India (2010).

Rupinder Singh is Post Graduate student in Computer Science & Engineering Department, Thapar University, Patiala, India and pursuing Master of Engineering in Software Engineering. He holds Bachelor of Technology (B.Tech) degree in Computer Science & Engineering from Thapar University Patiala, Punjab, India (2011).

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)