ijecs open access www.ijecs.in

International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 3 Issue 10 October, 2014 Page No.8835-8838

Component-Based Development Technologies and Limitations

Jyotsna

Assistant Professor, Department of Computer Science and Engineering
Sharda University, G.Noida,U.P-201306
jyotsna.seth@gmail.com

Abstract- ***The primary role of component-based software engineering is to address the development of systems as an assembly of parts (components), the development of parts as reusable entities, the maintenance and upgrading of systems by customising and replacing such parts. This paper defines current component-based software technologies used and their advantages and disadvantages***

Keywords- Software reuse, Software components, COM, DCOM, CORBA.

1. Introduction

Software reuse is one of the most effective ways of increasing productivity of the organization and improving quality of software components and projects. Component-based software engineering (CBSE) is an emerging software engineering paradigm in which applications are developed by integrating existing components. Component-based software engineering is a special case of software engineering which includes development for reuse and development with reusable assets. Reusable assets have special packaging, and are referred to as components [1]. The components which can be reused can be any units of integration, including computational components, interface components, communication components, designs and architectures [2]. There are many techniques for developing software with reuse of components.

2. Purpose

The purpose of component based development (CBD) is to develop large systems, incorporating previously developed or existing components, thus cutting down on development time and costs. It can also be used to reduce maintenance associated with the upgrading of large systems. It is assumed that common parts (e.g. classes or functions) in a software application only need to be written once and re-used again and again rather than being re-written every time a new application is developed [3].

The CBSE process is quite different from that of the traditional waterfall approach. CBSE not only requires focus on system specification and development, but also requires additional consideration for overall system context, individual components properties and component acquisition and integration process[3]. Component-based development approach is based on the idea to develop software systems by selecting appropriate off-the-shelf components and then to assemble them with a well-defined software architecture. The term component-based software development can be

referred to as the process for building a system using components

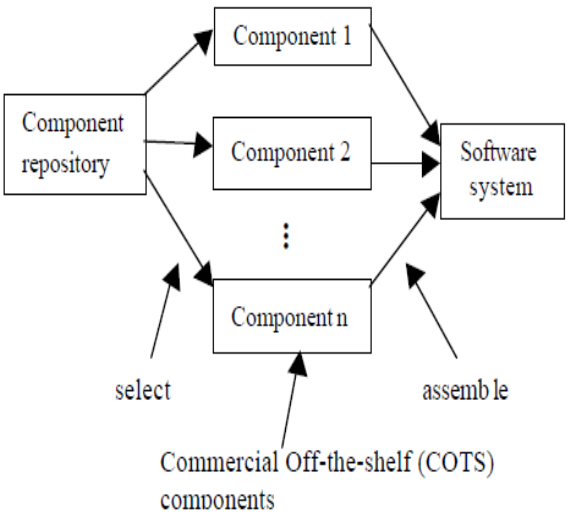

Figure1: Component-based software development[3]

3. Life Cycle of a Reusable Software component

The link between develop for reuse and develop by reuse is very important in the success of the new project development. The link between a build-for-reuse and build-by-reuse is a process consists of three phases, pre-store process, pre-use process, and reusable software component storage. In order to decide whether the intended components able to achieve their functions properly within other system, project manager evaluates the ability of the components to work and communicate with other components within other systems in different platforms. The reusability process considers the ability of the reusable components to achieve certain conditions that are required by reusable component library or new project. Reusable component library is a repository of software components able to be used in different systems Components exist in this storage have to follow certain standards and conditions. Hence, reusability test evaluates whether new component that needed to be stored in the library achieves these conditions. If the component passes this test, it will be kept. Otherwise, the process is required to modify this component in order to achieve a required standards and conditions in the library.

After component adoption, the required components returned from reusable component library have to be sent to reusability test(R-Test). This test is different from reusability test in the extraction process. The components are returned from a library have been verified according to the library conditions. But in this test, it evaluates whether the component is suitable for a new system[6].

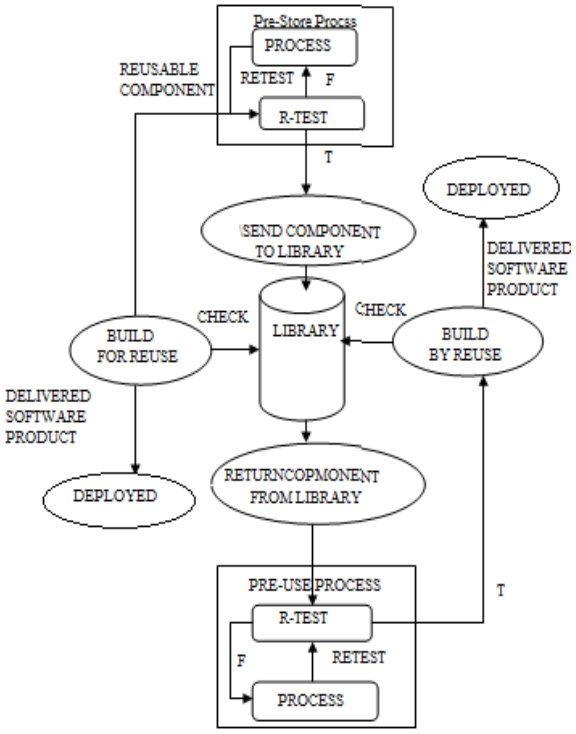

Figure 2: Life cycle of a Reusable software component [6]

4. Component Technologies

Some approaches, such as Visual Basic Controls(VBX), ActiveX controls, class libraries, and JavaBeans, make it possible for their related languages, such as Visual Basic, C++, Java, and the supporting tools to share and distribute application pieces. But all of these approaches rely on certain underlying services to provide the communication and coordination necessary for the application. The infrastructure of components sometimes called a *component model* acts as the "plumbing" that allows communication among components [1]. Among the component infrastructure technologies that have been developed, three have become somewhat standardized: OMG's CORBA, Microsoft's Component Object Model and Distributed COM, and Sun's JavaBeans and Enterprise JavaBeans[4].

Common Object Request Broker Architecture(CORBA)

CORBA is an open standard for application interoperability that is defined and supported by the Object Management Group (OMG). CORBA component model started out as a component model for distributed objects. The most important part of a CORBA system is the Object Request Broker (ORB). The ORB is the middleware that establishes the client-server relationships between components.

Component Object Model (COM) and Distributed COM (DCOM)

COM stands for component object model which defines how components and their clients interact. This interaction is defined such that the client and the component can connect without the need of any intermediate system component[4]. It evolved in two directions, with COM+ toward a simpler component model for in-process components and DCOM for distributed components[1]. As an extension of the Component Object Model (COM), Distributed COM (DCOM), is a protocol that enables software components to communicate directly over a network in a reliable, secure, and efficient manner.

Sun Microsystems's JavaBeans and Enterprise JavaBeans

JavaBeans component model is a component model for customizable reusable components [1]. Sun's Java-based component model consists of two parts: the JavaBeans for client-side component development and the Enterprise JavaBeans (EJB) for the server-side component development. The JavaBeans component architecture supports applications of multiple platforms, as well as reusable, client-side and server-side components[4].

5. Comparison

There are certain disadvantages of component-based software Engineering.

Time and effort required for development of components:- Among the factors which can discourage the development of reusable components is the increased time and effort required, the building of a reusable unit requires three to five times the effort required to develop a unit for one specific purpose.

Unclear and ambiguous requirement:-. Reusable components are, by definition, to be used in different applications, some of which may yet be unknown and the requirements of which cannot be predicted. This applies to both functional and non-functional[6].

Component maintenance costs:- component maintenance costs can be very high since the component must respond to the different requirements of different applications running in different environments, with different reliability requirements and perhaps requiring a different level of maintenance support.

Reliability and sensitivity to changes:- As components and applications have separate lifecycles and different requirements, there is some risk that a component will not completely satisfy the application requirements or that it may include concealed characteristics not known to application developers. When introducing changes on the application level, there is a risk that the change introduced will cause system failure.

6. CBD Techniques: Advantages & Disadvantages

	CORBA	COM/DCOM	EJB
Language binding	Language independent	Language independent	Language dependent
Communication Type	Synchronous, Asynchronous	Synchronous	Synchronous, Asynchronous
Platform dependency	Platform independent	Platform dependent	Platform independent
Modification & maintenance	Need extra modification & maintenance	Need extra modification & maintenance	Easier modification & maintenance.
Interface Language	CORBA IDL	Microsoft IDL	Java Programming Language + Annotations
Process	Top-down process	Bottom-up process	Top-down process
Deployment Environment	At run-time	At compilation and at run-time	At run-time
Compatibility & portability	Particularly strong in standardizing language bindings; but not so portable	Not having any concept of source-level standard of standard language binding	Portable by Java language specification; but not very compatible.
Implementation	Strongest for traditional enterprise computing	Strongest on the traditional desktop applications	Strongest on general Web clients.

Table 1: Comparison between different components technologies

7. Conclusion

CBD is a good solution that has potential to improve time – to-market and man power/cost trends that have been ongoing. CBD is best implemented using more modern software technologies like: COM, EJB, CORBA, ActiveX. There are certain drawbacks also which should be taken into consideration. To enjoy the advantages and avoid the problems and risks, we need systematic approach to component-based development at the process and technology levels.

8. Scope and proposed work in CBSD

The above discussion leads to the following topics that can be worked upon in future:
1) Generation and adaptation of component-based systems.
2) Components and model-driven development
3) Specification, verification, testing and checking of component systems.
4) Compositional reasoning techniques for component models.
5) Measurement and prediction models for component assemblies.
6) Patterns and frameworks for component-based systems.
7) Extra-functional system properties of components and component-based systems.
8) Static and execution-based measurement of system properties.
9) Assurance and certification of components and component-based systems.
10) Components for service-oriented architectures, web services and grid systems.
11) Development environments and tools for building component-based systems.
12) Components for real-time, secure, safety critical and/or embedded systems

9. References

[1] Hafedh mili, Ali mili, Sherif Yacoub and Edward Addy, "Reuse based Software engineering", ISBN 0-471-39819-5, 2002

[2] Kyo C. Kang **"**Issues in Component-Based Software Engineering" pp.784-790, 1999.

[3] Iqbaldeep Kaur, Parvinder S. Sandhu, Hardeep Singh, and Vandana Saini "Analytical Study of Component Based Software Engineering" pp.437-442, 2009.

[4] Xia Cai, Michael R. Lyu, Kam-Fai Wong "Component-Based Software Engineering: Technologies, Development Frameworks, and Quality Assurance Schemes"

[5] Anas bassam Al-badareen, Mohd hasan selamat, Marzanah a. jabar, Jamilah din,Ssherzod turaev "Reusable software component life cycle" international journal of computers issue 2, volume 5, 2011.

[6] Ivica Crnkovic Mälardalen University, Department of Computer Engineering, Västerås, Sweden "Component-based Software Engineering – New Challenges in Software Development"

[7] George T. Heineman and William T. Councill, "Component-Based Software Engineering Putting the Pieces Together", Addison-Wesley, Boston, MA ,880, June 2001.
[8] Ivica Crnkovic, Séverine Sentilles, Aneta Vulgarakis and Michel Chaudron," A Classification Framework for Component Models".