International Journal of Software Engineering.
ISSN 0974-3162 Volume 6, Number 1 (2015), pp. 1-7

http://www.irphouse.com

A Survey on Component-Based Software Development Technologies and Quality Assurance Schemes

Neha Prajapati
Department of Computer Science & Engineering
Galgotias University, Greater Noida
Greater Noida, India

Abstract

Component-based software development technique is based on the concept where we develop software systems by selecting appropriate off-the-shelf components and then assemble them along with well-defined software architecture. This approach is used because the new software development paradigm is much different from the traditional approach; quality assurance for component-based software development is a new topic in the software engineering community.
In this review paper survey of current component-based software technologies and frameworks along with their advantages and disadvantages are described as well as features they adopt. [6]

Keywords: Components, Technologies, Frameworks, Software Engineering

Introduction

Now-a-days software systems have become more and more large-scale, complex and difficult to control. This results in high development cost, low productivity, and unmanageable software quality. Thus, there is a demand of searching for a new, efficient, and cost-effective software development paradigm. Here we present one of the most promising solutions that is the component-based software development approach. This approach is based on the idea that software systems can be developed by selecting appropriate off-the-shelf components and then assembling them with well-defined software architecture. This new software development approach is very different from traditional approach in which software systems can only be implemented from scratch. These commercial off-the-shelf (COTS) components can be developed by different envelopers using different languages and different platforms. This can be shown in Figure 1, where COTS components can be checked out from a component repository, and assembled into a target software system.[5]

Component-based software engineering (CBSE) is currently in a period of rapid growth and change. Component-based software development (CBSD) can significantly reduce development cost and time-to-market, and improve maintainability, reliability and overall quality of software systems. Component-based software engineering (CBSE) is a branch of software engineering that emphasizes the separation of concerns in respect of the wide-ranging functionality available throughout a given software system. It is a reuse-based approach to defining, implementing and composing loosely coupled independent components into systems.[1] This practice aims to bring about an equally wide-ranging degree of benefits in both the short-term and the long-term for the software itself and for organizations that sponsor such software. Software engineering practitioners regard components as part of the starting platform for service-orientation. Components play this role, for example, in web services, and more recently, in service-oriented architectures (SOA), whereby a component is converted by the web service into a *service* and subsequently inherits further characteristics beyond that of an ordinary component. [1]

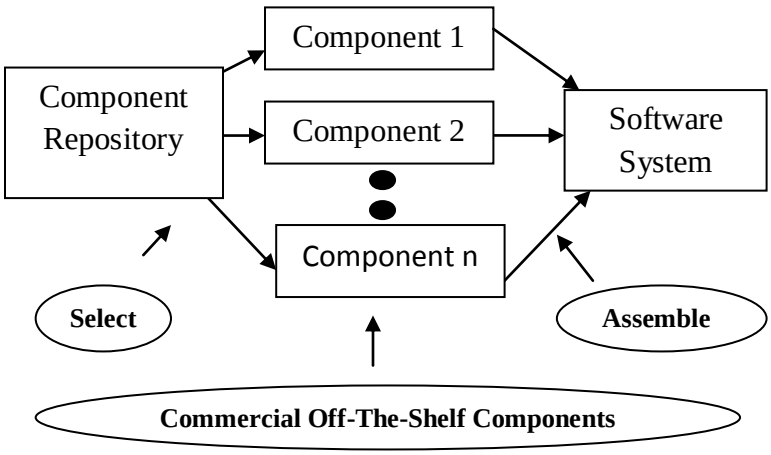

Figure 1: Component-based Software Engineering

other approaches. Software component technologies are just an emerging technology, which is far from being matured and practically implemented. There is no existing standards or guidelines in this new area, and we do not even have a unified definition of the major item "component". [7] However, we can say that a component has three main features:

Current Component-Based Technologies

Amongst the various existing component-based technologies, three of them have become somewhat standardize. They are as follows:

Common Object Request Broker Architecture (Corba)

Common Object Request Broker Architecture (CORBA) is an architecture and specification for creating, distributing, and managing distributed program objects in a network. It allows programs at different locations and developed by different vendors to communicate in a network through an "interface broker." CORBA was developed by a consortium of vendors through the Object Management Group (OMG), which currently includes over 500 member companies. Both International Organization for Standardization (ISO) and X/Open have sanctioned CORBA as the standard architecture for distributed objects (which are also known as components). CORBA 3 is the latest level.

The essential concept in CORBA is the Object Request Broker (ORB). ORB support in a network of clients and servers on different computers means that a client program (which may itself be an object) can request services from a server program or object without having to understand where the server is in a distributed network or what the interface to the server program looks like.[1][4]

CORBA and Microsoft have agreed on a gateway approach so that a client object developed with the Component Object Model will be able to communicate with a CORBA server (and vice versa).It is just an emerging technology as compared to EJB and COM/DOM and still needs extra modification.

Common Object Model (Com) and Distributed Com (DCOM)

COM is a framework for creating and using objects. COM, the Component Object Model delivers on the long promised benefits of object technology: [10]code reuse and off the shelf components. COM services are provided in a standard way, whether those services are required within a single running process, within two different processes on the same machine, or on two different processes across a network using DCOM. COM is about choice; it provides the choice of the highest volume languages and tools available, as well as the largest base of applications. COM also provides choice in the area of security, as it provides a common interface (SSPI) where various security providers can be plugged in. COM also provides choice of network transport.[8]

The COM specification has been complete since the end of 1992. Since then additions have been made, such as DCOM, but applications that worked then still work now. Unlike CORBA, COM provides the major elements necessary for a technology to succeed:

1. A solid specification,
2. A single reference implementation which has been ported to multiple platforms.

COM is ubiquitous; it is found on millions of systems worldwide and is a key part of most Microsoft software. Now that major systems Vendors such as HP, DEC and SNI have announced plans to ship COM on their systems within the year, COM will be used to create and use three-tier applications in many environments.[10]

Also, Component Object Model is:

- Specification
- Philosophy of how software is constructed
- Binary Standard

DCOM (Distributed Component Object Model) is a set of Microsoft concepts and program interfaces in which client program object s can request services from server program objects on other computers in a network. DCOM is based on the Component Object Model (COM), which provides a set of interfaces allowing clients and servers to communicate within the same computer (that is running Windows 95 or a later version).[1][4]

For example, you can create a page for a Web site that contains a script or program that can be processed (before being sent to a requesting user) not on the Web site server but on another, more specialized server in the network. Using DCOM interfaces, the Web server site program (now acting as a client object) can forward a Remote Procedure Call (RPC) to the specialized server object, which provides the necessary processing and returns the result to the Web server site. It passes the result on to the Web page viewer.

DCOM can also work on a network within an enterprise or on other networks besides the public Internet. It uses TCP/IP and Hypertext Transfer Protocol . DCOM comes as part of the Windows operating systems. DCOM is or soon will be available on all major UNIX platforms and on IBM's large server products. DCOM replaces OLE Remote Automation.[10]

DCOM is generally equivalent to the Common Object Request Broker Architecture (CORBA) in terms of providing a set of distributed services. DCOM is Microsoft's approach to a network-wide environment for program and data objects. CORBA is sponsored by the rest of the information technology industry under the auspices of the Object Management Group (OMG).COM/DOM technology is supported by a wide range of strong development environments as compared to CORBA and EJB. [1][4]

Enterprise Java Beans(EJB)

Enterprise JavaBeans (EJB) is an architecture for setting up program components, written in the Java programming language, that run in the server parts of a computer network that uses the client/server model. Enterprise JavaBeans (EJB) is an architecture for setting up program components, written in the Java programming language, that run in the server parts of a computer network that uses the client/server model.[10]

Enterprise JavaBeans is built on the JavaBeans technology for distributing program components (which are called Beans, using the coffee metaphor) to clients in a network. Enterprise JavaBeans offers enterprises the advantage of being able to control change at the server rather than having to update each individual computer with a client whenever a new program component is changed or added. EJB components have the advantage of being reusable in multiple applications. To deploy an EJB Bean or component, it must be part of a specific application, which is called a container. [1][4]

EJB is an emerging technology as compared to CORBA and COM/DOM.[10]

Quality Assurance For Component-Based Software Systems

Quality assurance for component-based software systems should be such that it addresses the life cycle and its key activities to analyze the components and achieve high quality component-based software systems. QA technologies for component-based software systems are currently premature, as the specific characteristics of component systems differ a lot from those of traditional systems in various ways. The identification of the QA characteristics, along with the models, tools and metrics, are all the need of the hour today. The main practices relating to Components and systems in the model contain the following phases:

Component Requirement Analysis:-
In this phase the requirements of the user are discovered and understood. The requirement of the component is documented accordingly. Validation of component requirement is also done here.

Searching New Component:-
In the second phase required component is searched from the component repository or outsourced from a third party. If it is not found then a new component is developed.

Develop New Component:-
If the required component is not in the repository then it is developed according to the requirement. After the new component has been developed, it is added in the component repository for future use.

Component Certification Process:-
Here, the components are audited to confirm its reliability and functionality. The component should satisfy the user requirement. Only then can a component be certified.

Component Customization:-
Components can also be modified as and when required. Sometimes, instead of developing an entire new component, an already existing component can be customized to fulfil a specific purpose. The customized component should be compatible with the other components also.

System Architecture Design:-
At this stage, the design the design requirement of the system is collected and an appropriate architecture is selected. Then the platform, programming languages, etc. are determined.

Testing on Each Component:-
Here each component is tested for its functionality individually and then by integrating them. The defects in the system implementation are detected and eliminated.

System Maintenance:-
This phase provides service and maintenance activities required to use the software effectively. The designed system should be adaptive to changes in the environment and market so that it can be modified from time to time.

Advantages and Disadvantages of the Technologies, One over the Other
After the survey certain conclusions have been drawn after comparing the above mentioned technologies. COM/ DCOM in a way is better technology than CORBA and EJB because they are the underdeveloped technologies while COM/ DCOM is supported by various development environments.

Also, COM/DCOM is platform dependent technology while CORBA and EJB are platform independent technologies. Generally, on a multi-platform environment we need an additional level of abstraction between language and the machine. Additional level tells the specific machine how to run the code in its environment and brings more code that the system has to run to handle a given set of instructions. So where platform independency is a necessity only there the other technologies should be used. COM/ DCOM is language independent, which adds an extra advantage to it.

Conclusion and Future Work

In this survey paper, we have reviewed existing component-based software technologies and the features they adopt and also compare them on various parameters. The following technologies have been described and compared:

- Sun Microsystems's JavaBeans and Enterprise JavaBeans (EJB)
- Common Object Request Broker Architecture (CORBA)
- Component Object Model (COM)

Distributed COM (DCOM)

As far as the future work is concerned, various new features can be added to the technologies and frameworks and applied to various softwares. Further some new major activities in the component-based systems, technical and nontechnical issues that need to be resolved for widespread adoption of this approach can be discussed. [9]

References

[1] "Component-Based Software Engineering: Technologies, Development Frameworks, and Quality Assurance Schemes"- Xia Cai, Michael R. Lyu, Kam-Fai Wong Roy Ko The Chinese University of Hong Kong Hong Kong Productivity Council, 2000.

[2] "A Generic Component Framework for System Modelling" by Braatz1, Markus Klein1, and Martti Piirainen Technische University at Berlin, Germany, 2004.
[3] "A Framework for Formal Component-Based Software Architecting" by M.R.V. Chaudron, E.M. Eskenazi, A.V. Fioukov, D.K. Hammer Department of Mathematics and computing Science, Technische Universiteit Eindhoven,2001.
[4] "A Comparison of Distributed Object Technologies" by Carl-Fredrik Sørensen, The Norwegian University of Science and Technology, 2014.
[5] "A Case Study Approach to Teaching Component Based Software Engineering" by Allen Parrish and Brandon Dixon (Department of Computer Science) David Hale and Joanne Hale (Area of Management Information Systems),The University of Alabama The University of Alabama,1999
[6] "A Comparison of Distributed Object Technologies" by Carl-Fredrik Sørensen, The Norwegian University of Science and Technology, 2014.
[7] Component Software: Beyond Object-Oriented Programming. 2nd Edition (2002) by C Szyperski
[8] "Software Reuse Architecture, Process, and Organization for Business Success," by M. L. Griss Proceedings of the Eighth Israeli Conference on Computer Systems and Software Engineering, 1997.
[9] "New Age of Software Development: How Component-Based Software Engineering Changes the Way of Software Development?" by Mikio Aoyama
[10] "Component-based Development Process and Component Lifecycle" by Ivica Crnkovic, Stig Larsson, Michel Chaudron, 2006