COMPARATIVE STUDY OF TECHNOLOGIES RELATED TO COMPONENT-BASED APPLICATIONS BASED ON THEIR RESPONSE TIME PERFORMANCE

Richa Balauria[1], **Arvind Kalia**[2]

Department of Computer Science, H.P University, Shimla (H.P), India

richabalauria.21@gmail.com, arvkalia@gmail.com

Abstract: *Component Based software development is an approach to develop the software system by selecting appropriate off-the-shelf (COTS) components and assemble them into a target software system under a well-defined software architecture to perform a well defined function [4].Various component technologies such as Component Object Request Broker Architecture (CORBA), Component Object Model (COM) and Enterprise JavaBeans (EJB) are used to develop these Component-Based applications which should have high performance.*

So appropriate component technology should be selected to develop the Component-Based application. This study basically uses a tool to measure the response time of Component-Based application based on various technologies. By comparing these response times, we can find out that COM-Based application has less response time than EJB-Based application and thus can provide the required performance.

Keywords: *Component-Based application, Response time, Common Object Request Broker architecture (CORBA), Component Object Model (COM), Enterprise JavaBean (EJB).*

1. Introduction

With the increase in development in the software industry, there is a great demand for large-scale software systems but this led to increase in complexity, high development cost and low performance [7].So there is a need of new methodology to develop the large-scale software systems that led to Component-Based Software Development (CBSD). CBSD can significantly reduce development cost and improve maintainability, reliability and overall quality of software systems [5] [6]. A component can be defined as an independent and substitutable part of a system which performs a well-defined function under a well-defined architecture [2]. CBSD is used to develop the Component-Based application that performs a well defined function.

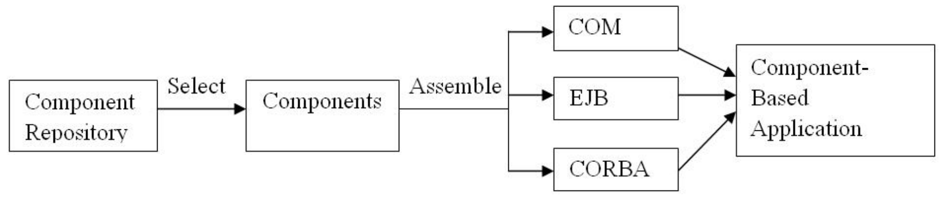

Fig. 1: Development of Component-Based application

Development of Component-Based application is illustrated in Figure 1 where appropriate components are selected from the component repository. Then these components are assembled either through CORBA, COM or EJB to develop Component-Based Application to perform a set of well-defined functions. Various implementation issues arise when developing a Component-Based Application. The following are the implementation issues:-

- Technology related to Component Based Software Engineering (CBSE).
- Selection criteria for components and assemblies of components.
- Predictability of component composition and configuration management of components.
- Verification of applications based on component attributes. [3]

Recent developments in Component-Based standards and technologies have led to the increase in development of Component-Based software systems [1].Technology is basically a standard to develop Component-Based application. While developing a Component-Based application following points should be considered:

- The syntax of components.
- The semantics of components.
- The composition of components [8].

Among the Component infrastructure technologies that have been developed, three have become somewhat standardized: Microsoft's COM, Sun's EJB and OMG's CORBA [12].These technologies are explained below.

2. Review Literature

2.1 A Brief Overview of Component Object Model (COM)

COM is introduced by Microsoft in 1993.It is a general architecture for Component-Based application. In COM model, interfaces are kept separate from the implementation details so COM allows reuse of components with no knowledge of their internal implementation [16] [2]. The main purpose of COM is to provide the solutions to the distributed Component-Based applications so that they can communicate among different programming languages and platforms [13].

COM uses system-level code that is implemented in the form of dynamic link libraries, collectively called COM library. Client sends the request to the application programming interface which in turn instantiate the objects that client wants to use. COM library is responsible for loading and activating server objects. Then these server objects respond to the client request [16].

The features of COM are beneficial in a way: COM provides a standard to develop a Component-Based Application.COM enhances code reusability. It means that the component which can be used by one application can be reused by another application.COM also keeps the implementation details of an object to be hidden, thus it supports encapsulation. It also supports polymorphism and inheritance. Therefore COM is an extension of OO technology and it is acceptable in today's world. COM also provides language independence.COM is supported by a wide range of strong development environments [13].A binary standard for component interaction is the heart of COM [2].

However, COM needs extra modification and maintenance [2].

2.2 A Brief Overview of Enterprise JavaBeans (EJBs)

It was originally developed in 1997 by IBM and later adopted by Sun Microsystems. EJBs is a server side model that basically intends to provide a standard way to implement the back-end 'business' code typically found in enterprise applications [18].

Enterprise JavaBeans component model basically consists of two parts: JavaBeans for client-side development and Enterprise JavaBeans for server-side development. The main purpose of EJBs is to support the applications of multiple platforms [19].

EJB mainly holds two major types of beans:

1. Session Beans- These beans can be accessed via interface (Local or Remote) or without interface (local semantics apply in this case).Session beans are further divided into: Stateful Session Beans: It keeps track of client with which they interact throughout a session. Access to single bean instance is prohibited. Stateless Session Beans: It doesnot keep track of the client with which they interact during a session. However, concurrent access to the single bean instance is prohibited. Singleton session Beans: It keeps track of a global shared state with JVM. Only one bean instance can be concurrently accessed at a time.

2. Message Driven Session Beans: These beans also support asynchronous execution through messaging paradigm [18].

The features of EJBs are beneficial in a way: EJBs provides component portability. It also supports platform independence. EJBs application can be customized without access to the source code [14].It provides a binary standard for component interaction [2].

EJB also have some limitations which are as: Enterprise JavaBeans specification is too large and complicated to understand [10]. It has poor object/relational mapping [9].

2.3 A Brief Overview of Common Object Request Broker Architecture (CORBA)

CORBA is a standard managed by Object Management Group (OMG). CORBA enables software components written in multiple computer languages and running on multiple computers to work together and thus support component interoperability. CORBA is used in object-oriented distributed systems [11]. CORBA's first version was introduced in 1991 [17][2].

In CORBA, Object Request Broker (ORB) is the most important part of CORBA that establishes the client server relationships between components. It basically acts as middleware between components [2]. The function of ORB is explained in given Figure 2.

The client invokes a method on server object. ORB intercepts the call and finds the server object that can implement the request and return the request [2]. The features of CORBA are beneficial in a way: CORBA provides language independence. It also provides OS independence [17].

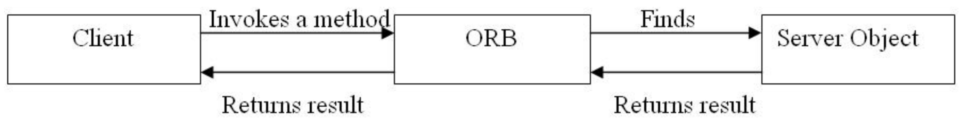

Figure 2: The ORB function

There is no doubt that CORBA is the first step towards Component-Based technologies but it is also true that it has number of limitations. CORBA is basically underdeveloped and doesnot support strong development environment [2] and is not considered. That is why in our study we are not considering CORBA based applications.

Now-a-days, COM and EJB technologies are widely used. Therefore we focused on COM and EJB based applications to do our analysis.

3. Data Analysis

In our study we analyzed COM based application i.e. Tube catcher 2.0 and EJB based application i.e. Google Video Studio enterprise 6.1.05 by using MyARM tool. Tube catcher 2.0 (downloaded from [22]) and Google Video Studio enterprise 6.1.05 (downloaded from [21]) are used to download the FLV files.My Application Response Time Measurement (MyARM) tool (downloaded from [20]) is used to measure the response time of an application. It was jointly introduced by Tivoli software and Hewlett Packard in 1996 [15].

For analysis, the files are classified according to their size as:-Very Small (less than 512 KB), Small (greater than 512KB and smaller than 1MB), Medium (greater than 1MB and smaller than 10MB), Large (greater than 10MB and smaller than 50MB) and Very Large (size greater than 50 MB).The method adopted for the purpose of analysis is to download the files by both technologies i.e. COM and EJB. During the process the performance of both technologies was calculated by using MyARM tool. The internet connection used was broadband. Since the speed of internet connection for the purpose of downloading may vary therefore the process was repeated a number of times on the same file in the same time period. The average of these readings was noted to evaluate response time of both the applications when these applications were downloading the same FLV file a number of times. This procedure would be repeated for other categories of files a number of times.

Very small files: The FLV file of size less than 512KB (say 228KB) was downloaded a number of times and the readings for both applications was observed in Table 1.

Table 1: Response Time of very small files

Component Based application	Start Date	Start Time	Duration (milliseconds)	Status	Average (milliseconds)
Tube catcher	28.08.2011	10:16:00.743	22881.000	GOOD	16905.600
	28.08.2011	10:21:49.865	14145.000	GOOD	
	28.08.2011	10:26:44.108	15423.000	GOOD	
	28.08.2011	10:30:16.500	15306.000	GOOD	
	28.08.2011	10:33:13.565	16773.000	GOOD	
Google Video Studio Enterprise	28.08.2011	10:20:09.088	38103.000	GOOD	25916.200
	28.08.2011	10:23:09.885	21735.000	GOOD	
	28.08.2011	10:27:08.571	22629.000	GOOD	
	28.08.2011	10:30:42.176	25305.000	GOOD	
	28.08.2011	10:33:37.822	21809.000	GOOD	

Small files: The FLV file of size greater than 512KB and less than 1MB (say 838KB) was downloaded a number of times and the readings for both applications was observed in Table 2.

Table 2: Response Time of small files

Component Based application	Start Date	Start Time	Duration (milliseconds)	Status	Average (milliseconds)
Tube catcher	27.08.2011	14:45:48.898	49121.000	GOOD	58658. 600
	27.08.2011	14:52:13.598	57868.000	GOOD	
	27.08.2011	14:58:25.858	61389.000	GOOD	
	27.08.2011	15:04:00.822	65622.000	GOOD	
	27.08.2011	15:08:00.503	59293.000	GOOD	
Google Video Studio Enterprise	27.08.2011	14:42:47.099	70854.000	GOOD	62745. 400
	27.08.2011	14:50:44.344	60717.000	GOOD	
	27.08.2011	14:57:00.921	56515.000	GOOD	
	27.08.2011	15:02:48.642	60126.000	GOOD	
	27.08.2011	15:06:43.104	65515.000	GOOD	

Medium size files: The FLV file of size less than 1MB and greater than 10MB (say 1.38MB) was downloaded a number of times and the readings for both applications was observed in Table 3.

Table 3: Response Time of medium size files

Component Based application	Start Date	Start Time	Duration (milliseconds)	Status	Average (milliseconds)
Tube catcher	27.08.2011	21:47:56.655	225770.000	GOOD	185353.000
	27.08.2011	21:59:52.625	152884.000	GOOD	
	27.08.2011	22:21:34.187	177405.000	GOOD	
Google Video Studio Enterprise	27.08.2011	21:53:41.375	226936.000	GOOD	249110.300
	27.08.2011	22:05:38.828	232807.000	GOOD	
	27.08.2011	22:16:16.020	287588.000	GOOD	

Large files: The FLV file of size less than 10MB and greater than 50MB (say 10.2MB) was downloaded number of times and the readings for both applications was observed in Table 4.

Table 4: Response Time of Large files

Component Based application	Start Date	Start Time	Duration (milliseconds)	Status	Average (milliseconds)
Tube catcher	27.08.2011	23:30:10.371	167675.000	GOOD	132741.800
	27.08.2011	23:43:05.331	136029.000	GOOD	
	27.08.2011	23:50:30.773	120036.000	GOOD	
	27.08.2011	23:58:07.105	113128.000	GOOD	
	28.08.2011	00:05:44.454	126841.000	GOOD	
Google Video Studio Enterprise	27.08.2011	23:37:41.585	205625.000	GOOD	158675.800
	27.08.2011	23:45:36.933	196467.000	GOOD	
	27.08.2011	23:54:23.240	134695.000	GOOD	
	28.08.2011	00:02:22.937	132316.000	GOOD	
	28.08.2011	00:08:50.053	124276.000	GOOD	

Very Large files: The FLV file of size greater than 50MB (say 75.5MB) was downloaded a number of times and the readings for both applications was observed in Table 5.

Table 5: Response Times of very large files

Component Based application	Start Date	Start Time	Duration (milliseconds)	Status	Average (milliseconds)
Tube catcher	28.08.2011	00:32:17.871	1871398.000	GOOD	1525454.000
	28.08.2011	01:37:28.145	1585246.000	GOOD	
	28.08.2011	02:31:32.757	1187854.000	GOOD	
Google Video Studio Enterprise	28.08.2011	01:04:24.743	1871398.000	GOOD	1567991.300
	28.08.2011	02:06:09.878	1419668.000	GOOD	
	28.08.2011	03:04:47.572	1412908.000	GOOD	

On the basis of above analysis, finally it is observed that Tube Catcher based on COM technology has a far better performance level than Google video studio enterprise based on EJB technology. In all circumstances i.e. for very small files, small files, medium size files, large files and very large files, it is clear that COM based application is better than EJB based application.

The same has been depicted in Figure 3.

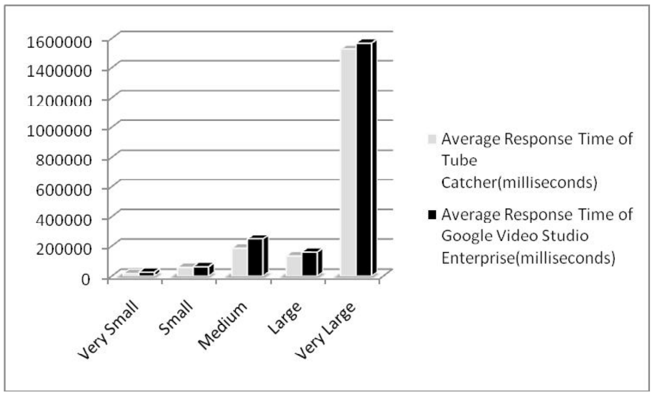

Fig. 3: Comparison of response time of Tube Catcher and Google Video Studio Enterprise

By observing the response time in above graph, it is clear that tube catcher based on COM technology is better than Google Video Studio Enterprise based on EJB technology as the tube catcher takes less response time than Google Video Studio Enterprise.

4. Conclusion

In his study, the researcher have analyzed the importance of implementation issues during development of a Component-Based Application by considering one of the important implementation issues i.e. Technology related to CBSE. The researcher have analyzed the technologies related to CBSE by comparing the response time of Component-Based Application based on both technologies i.e. COM and EJB and therefore concluded that the Application based on COM technology is better than the Application Based on EJB technology.

References

[1] Bose Prasanta, "Scenario-Driven Analysis of Component-Based Software Architecture Models", George Mason University, pp. 1-1.

[2] Cai Xia, R. Lyu, Wong Kam-Fai, Ko Roy, "Component-Based Software Engineering:Technologies, Development Frameworks, and Quality Assurance Schemes"**,** The Chinese University of Hong Kong, Hong Kong Productivity Council, pp. 1-4.

[3] Crnkovic Ivica, Larsson Stig, Stafford Judith, "Component-Based Software Engineering:Building systems from Components", 9th IEEE Conference and workshops on Engineering of Computer-Based Systems, pp. 1-1.

[4] G. Pour, "Component-Based Software Development Approach: New Opportunities and 375-Challenges," Proceedings Technology of Object-Oriented Languages, 1998. TOOLS 26, pp 383.

[5] G. Pour, "Enterprise JavaBeans, JavaBeans & XML Expanding the Possibilities for Web- Based Enterprise Application Development," Proceedings Technology of Object Oriented Languages and Systems, 1999, TOOLS 31, pp.282-291.

[6] G. Pour, M. Griss, J. Favaro, “Making the Transition to Component-Based Enterprise Software Development: Overcoming the Obstacles – Patterns for Success,” Proceedings Technology of Object-Oriented Languages and systems, 1999, pp.419 – 419

[7] G. Pour, “Software Component Technologies: JavaBeans and ActiveX,” Proceedings of Technology of Object-Oriented Languages and systems, 1999, pp. 398 – 398.

[8] Lau Kung-Kiu, Wang Zheng, “A Taxonomy of Software. Component Models”, Proceedings of the 2005 31st EUROMICRO Conference on Software Engineering and Advanced Applications, 05 2005,pp. 1-1.

[9] Neward, T. The Death of EJB As We Know It? http://www.oreillynet.com/pub/wlg/1922.

[10] Sheil,To EJB, or not to EJB? http://www.javaworld.com/javaworld/jw-1207 yesnoejb.html.

[11] S.S. Yau, B. Xia, “Object-Oriented Distributed Component Software Development based on CORBA, “Proceedings of COMPSAC’98. The Twenty-Second Annual International, 1997 pp. 246-251.

[12] W. Kozaczynski, G. Booch, “Component-Based Software Engineering,” IEEE Software Volume: 155, Sept.-Oct. 1998, pp. 34–36.

[13] http:www.peterindia.net/COMOverview.html, “Component Object Model (COM)”.

[14] https://publib.boulder.ibm.com/infocenter/cicsts/.../topic/com../dfhpj41.htm,”Benefits of EJB technology”.

[15] http://en.wikipedia.org/wiki Application_Response_Measurement,”Application Respons Measurement”.

[16] http://en.wikipedia.org/wiki/Component_Object_Model,”Component Object Model”.

[17] http://en.wikipedia.org/wiki/Common_Object_Request_Broker_Architecture, “Common Object Request Broker Architecture”.

[18] http://en.wikipedia.org/wiki/Enterprise JavaBean," Enterprise JavaBean".

[19] SUN http://developer.java.sun.com/developer, Mar. 2000.

[20] http://www.myarm.com.

[21] http://www.brothersoft.com/google-video-studio-enterprise-104663.

[22] http://www.brothersoft.com/atubecatcher-72345.html.