Comparative Analysis of Business Object Approaches

Alina VLĂŞCEANU, alina.vlasceanu@ccr.ro
The Constitutional Court of Romania, Chief of the IT Department
Roxana-Adina IRIMIA, Roxana.Irimia@ro.ibm.com
IBM Romania, IT specialist

This paper presents a comparison of several technologies for developing distributed applications. The specific technologies into consideration are: one focused on COM/DCOM/COM + Microsoft technologies, Internet Explorer and ActiveX – and the other focused on Netscape, CORBA, JAVA/J2EE solutions.
Rapidly changing business processes require quick adaptation of supporting information systems. Component technologies in general and business objects, in particular seem a promising approach.
In this paper, we survey, analysis and compare objects approaches. We develop a comparison model covering concepts, distribution infrastructure, object facilities and object solutions. We then use the model to analysis the Combined Submission to the OMG Business Object Domain. Each of the approaches allows us to compare them analytically.
Keywords: *integrated technologies, interoperability, distributed systems, components, distributed architecture*

MICROSOFT versus SUN vision related to components technologies

Web industry is divided in two polls: one focused on COM/DCOM/COM + Microsoft technologies, Internet Explorer and ActiveX – and the other focused on Netscape, CORBA, JAVA/J2EE solutions.
In order to interact, the components must affiliate to a binary structure specified by Microsoft. As long as the components affiliate to this binary structure, the components written in different languages can interoperate.
COM/DCOM/COM+ represent a part of the possible technologies that allow distributed applications. Some technologies, like RPC (Remote Procedure Call) permit distribution at low level.
Usually, COM and DCOM are associated along with OLE, ActiveX, MTS and COM+. Indeed, these, as well as other technologies constitute the Microsoft's object oriented and distributed strategy. This strategy is called DNA (Distributed InterNet Architecture) and has a complete set of products and specifications for implementing networking central applications.
CORBA (Common Object Request Broker Architecture) and J2EE (Java 2 Enterprise Edition) technologies can be considered directly competitors of COM/DCOM.
COM objects can be created and manipulated by Java code.
The tools are supplied for creating Java classes which contains COM information libraries.
Generally, the Microsoft approach regarding Java support implies a close joining with existing Internet strategy (Internet Explorer, COM/DCOM, and ActiveX).
The main differences between EJB and COM are:
- COM components can be written in many languages (Visual Basic, C++, Java, Delphi), while EJB (Enterprise JavaBeans) can be written only in Java code;
- COM components work only on Windows platforms, while EJB and EJB associated servers are portable and, as a result, are working on a large variety of platforms.
While COM and DCOM represents the low level of technology that allows components to interact, OLE, ActiveX and MTS represent the high level of the applications services which are built on the COM and DCOM technologies.
OLE deliver services such as linked and embedding objects which are used in creating compound documents.

ActiveX extend the basis capacities of COM related to organization services, such as transactions and security which allow Enterprise Information System's to be built using COM components.
Distribution support and services quality delivered by COM+ can help to overcome the complexity involved in these architectures.
COM+ integrates MTS services and messages which form a queue to COM and make COM programming much easier through an approach integration with Microsoft languages, for example Visual Basic, Visual C++ and J++.
Directly competitors of CORBA's are DCOM/ActiveX, WebObjects and RMI.
The joining between JAVA and CORBA brings a lot of advantages for CORBA, such as: supplements CORBA services regarding the objects life cycle control, simplifies code distribution in large CORBA systems, and it is the ideal language for writing CORBA objects.
DCOM, just as CORBA, separates implementation interface, and all the interfaces must be described using an IDL. Microsoft IDL depends on DCE and it is not compatible with CORBA. While CORBA depends on classic object model, DCOM doesn't. A DCOM component doesn't support multiple heritance, but it can implement several interfaces, and thus code reutilization doesn't result from heritance, but from aggregation.

CORBA - JAVA RMI comparative analysis
The relation between JAVA and CORBA is more about complementary than competition. JAVA is an excellent language for describing CORBA objects. It implements components service from CORBA, based on OpenDoc.
While CORBA defines visual containers for components and mobile recording containers, JAVA can deliver the content of such containers (Java Beans).
More than that, the mobile code facilities from JAVA allow the partition of an application between client and server at the execution moment. JAVA simplifies the code distribution in large CORBA systems through a centralized management of code on the server and distribution to the clients, when and where is necessary.
In compensation, CORBA brings three immediately benefits to Web applications: allow the avoiding of the CGI neck less through direct invocation by the client of servers' methods, facilities a scalable infrastructure of servers (server objects can communicate using CORBA ORB), extends JAVA with a distributed objects infrastructure (the possibility of communication between different addresses spaces).

Comparative analyze of CORBA – SOAP technologies
In order to understand the present problems in the framework of Web services technologies, a comparative analysis regarding SOAP and other distributed technologies is absolutely necessary.
CORBA offers an object oriented distributed technology. Moreover, the Enterprise Java Beans technology promoted by the Sun Microsystems and adopted by all the big companies concentrated on JAVA development, became compatible with OMG platform. The SOAP goal isn't to replace or to remove CORBA from the market, but, on the contrary, OMG recently launched SOAP – CORBA interoperability specification.
Also, there are certain projects that try to create interoperability between technologies based on SOAP and CORBA platforms.
Further, it can be identifies a set of CORBA services that doesn't have an equivalent in SOAP technology.
SOAP contains specifications which exceeded the level of a protocol. In basis of the SOAP facilities and considering the W3C recommendations on the invocation modes, higher level services are built (for example WSDL and UDDI).
Interoperability between different platforms
CORBA contains components that work on any computer, called ORB (Object Request Broker) components.
In version 1.0 CORBA didn't offer the implementation details of these components, and therefore the interoperability between clients and server was much reduced.

CORBA 2.0 has introduced the IIOP protocol which offers high interoperability between applications from different platforms.
On the other side, SOAP runs on the WEB protocols known as HTTP or SMTP. In this way, it can be used on any existing platform.
Data transmitting format
The CORBA platform uses the binary encoding format and in this way it brings a plus in the system's performance.
IIOP protocol implies that both the transmitter and the receiver have complete knowledge regarding the message and it doesn't include meta-information. This enhances the performance, but excludes the possibility of intermediary nodes for processing and transforming messages. In this way it is easier to repair programs, because the data are from the beginning within a system legible for the developers.
SOAP uses the XML format for data encoding both in text and binary form.
Scalability
CORBA is an object oriented platform. Therefore, most calling are stateful. Nevertheless, for stateless callings (pure RPC), CORBA offers a very simple mechanism through ORG components of every computer. The choice between stateful and stateless invocation is at the developers' option.
In most cases, SOAP functions over HTTP or SMTP protocols. HTTP is a stateless protocol, and therefore the stateful session between successive invocations can be achieved either through cookies or by the transmission of certain objects IDs into messages.
Objects identity and life cycle
The identity of an object is maintained through objects references of "substitute" type (stub) which work on the client's computer. CORBA can be used for transparent communications between application's objects.
SOAP doesn't require an identity mechanism of objects. The Web services are, generally, identified by an URL. If the identity object is missing, we can notice that SOAP is an RPC system and not an ORPC system, because it isn't based on objects (classes, instances), and even less oriented on objects (heritance, polymorphism, etc.). If the identities are not maintained, the objects are destroyed after an inactivity period.
Transport protocols
CORBA in version 2.0 defines IIOP, as a different reading of GIOP (General Inter-ORB Protocol) based on TCP/IP protocol.
There is the possibility to use another protocol called DCE CIOP (DCE Common Inter-ORB Protocol).
In case of SOAP, the protocol defined for the methods calls is HTTP, while SMTP can be used for other types of messages.
There is also the possibility to implement the support for other protocols (such as FTP protocol).
Security
CORBA security service offers a variety of security politics for different scenarios. This service refers to the authentication, authorization and encoding of messages.
SOAP allows the use of different services that refer to different levels. However, there is no standard regarding the authentication and authorization, each web service developer implementing his own security method.
Easy to use
CORBA is a platform where implementation is quite complex from many points of view:
- use of two different languages (client/server language and interfaces description languages – IDL);
- a very large set of services (security, transactions, objects life cycle, publishing and finding interfaces etc.);
- the distribution system of clients and servers with ORB components require the presence of these components on the client computers.

On the other hand, the most important advantage of SOAP technology is its simplicity.
The basic technologies are HTTP and XML, so the development is quite simple and easy to understand. However, the high level services such as WSDL and UDDI have complicated a bit the SOAP world.
Limitation of SOAP technology
As it was already mentioned, CORBA offers a quite large set of services. For example,

when Enterprise Java Beans appeared, Sun Microsystems related to the CORBA list of services in order to implement a subset of such services.
Yet, SOAP offers a larger simplicity both in message systems and in data representation. But the lack of certain services bothered the developers: there are no objects identities, there isn't an event service; the procedure calls are only synchronous; there isn't a standard modality of authentication and authorization.

JAVA VERSUS .NET
Nowadays, there are two important platforms for new application: Java 2 Enterprise Edition (J2EE) and Microsoft .NET.
.NET has certain advantages, because it has used from the beginning modern technology, such as XML and web services. By developing its own virtual java machine, Microsoft solved the problems of JAVA interpreter.
.NET and J2EE are very similar, but Microsoft offers a more modern technical solution, by implementing web technologies and XML language. Likewise, the C# language and the virtual machine (CLR) are ideas deriving from JAVA.
Several functionalities of Windows operating systems can be used directly, such as IIS (Internet Information Services) web server, Active Directory, OLEDB and Windows Load Balancing. The efficient coupling with the operating system is the cause of the improved performances of .NET applications, as compared with J2EE applications, although it is difficult to assess the objectivity of the tests.
In order to take the right decision in the future, the users have to consider two important criteria:
- the potential of development and comprehensibility maintained by a certain platform;
- the platform's own solutions offer, which are necessary to remain on the market.

JAVA language had an important development in the last five years, due to the associated technical platform power.
Recently, Microsoft offered a similar technological platform.
The concept of Component Object Model (COM) became too complex because of the support for different languages. Likewise, the distribution processing with DCOM solution based on Microsoft RPC concept and on Windows registry didn't prove to be compatible with Internet.
J2EE products acquired, in time, an acceptable level of maturation.
If we measure the productivity only according to the "code numbers", .NET has certain advantages as against J2EE.
Also, there are other differences of strategic importance:
A. J2EE isn't a product, but a specification for which different companies offer several products.
The applications are independent of the middleware support owner. In this way, the companies not only obtain independence as regards certain providers, but they can develop their own technological platforms.
.NET is a products collection of a single producer and it works only with Windows system. In this way is assured the integration of different components using some special characteristics of the Windows operating systems.
B. J2EE is independent of the operating system concept.
The portability is provided by Java Runtime Environment, and the application server and other middleware products can be programmed according to the operating system.
Beside these aspects, there are other important criteria to take into account while making a decision regarding these two technologies, such as the comprehensibility and development level. The J2EE advantage is the existence of API interfaces -Application Programming Interface), which creates a technological independence of applications.
This facilitates the future development of technology with reduced secondary effects. JAVA components model is more methodic and more elaborated, and the architecture based on connectors offers the basis for a larger interoperability than the correspondent facilities in the .NET technology.

Conclusions

The approaches we assessed are at different stages of their development and they are based on different technologies.

Business applications composed from business objects will be different from conventionally developed ones. Business applications will consist of heterogeneous and distributed business objects that encapsulate certain functionality. Business objects exist independently and autonomously. They are integrated into coherent software architectures by business object facilities.

Software processes that deploy business objects will be different, too.

They will have to be more flexible and adaptable. In particular, they will be less geared towards the development of objects. Future software processes will involve explicit make-or-buy decisions. These have to be supported by tasks that research the market for suitable objects and activities that evaluate candidate objects.

References

[1] Support for comprehensive reuse. Software Engineering Journal, *V. R. Basili and H. D. Rombach*, September [1991]

[2] Essays on Object-Oriented Software Engineering, *E. Berard*, Prentice Hall, [1993]

[3] Object-Oriented Languages, Systems and Applications, *G. S. Blair, J. J. Gallagher, D. Hutchinson and D. Shepard*, Halsted Press, [1991]

[4] Middleware Isolates Business Logic, *K. Bohrer*, Object Magazine, November [1997]

[5] Microsoft Transaction Server Programming, *S. Hillier*, Microsoft Press, [1998]

[6] Software reuse, *C. W. Krueger*, ACM Computing Surveys, 24(2), [1992]

[7] Instant UML, *P. A. Muller*, Wrox Press, [1997]

[8] OMG. COMMON Facilities RFP-4: Common Business Objects and Business Object Facility. Technical Report TC Document CF/96-01-04, Object Management Group, 492 Old Connecticut Path, *Framingham*, MA 01701, USA, [1996]

[9] CORBA services: Common Object Services Specification, Revised Edition. 492 Old Connecticut Path, *Framingham*, MA 01701, USA, March [1996]

[10] OMG. Combined Business Object Facility: Business Object Component Architecture. Technical Report TC Document BOM/98-01-07, Object Management Group, 492 Old Connecticut Path, Framingham, MA 01701, USA, JAN [1998]

[11] OMG. Combined Business Object Facility: Common Business Objects. Technical Report TC Document BOM/98-01-06, Object Management Group, 492 Old Connecticut Path, Framingham, MA 01701, USA, JAN 1998.

[12] OMG. Combined Business Object Facility: Interoperability Specification. Technical Report TC Document BOM/98-02-03, Object Management Group, 492 Old Connecticut Path, Framingham, MA 01701, USA, JAN [1998]

[13] Developing Java Web Services, *Ramesh Nagappan, Robert Skoczylas and Rima Patel Sriganesh*, John Wiley & Sons, [2003]

[14] SOAP Programming with Java, *Bill Brogden*, Sybex, [2002]

[15] Component-Based Software Development: Case Studies, *Kung-Kiu Lau*, World Scientific Publishing Co, [2004]

[16] Developing Secure Distributed Systems with CORBA, *Ulrich Lang and Rudolf Schreiner*, Artech House, [2002]