Centre for SOFTWARE Engineering

# Enterprise Computing

## Introduction

The vision of Enterprise Computing is one of high volume, complex transactional systems interacting seamlessly with each over the Internet to deliver business benefit to their respective organisations. Up until now constraints imposed by legacy systems and technology immaturity has made this goal seem almost unrealisable. However, we are now at a point in time where we have a near complete distributed component model in the CORBA/EJB model and several partial implementations of this model already completed or in the pipeline. These implementations, known as application-servers, enable the development of large-scale enterprise applications.

In this briefing we will look first at the requirements of enterprise computing. We will then discuss the two models that try to address these requirements CORBA/EJB and COM+. We will also look at different implementations of the CORBA/EJB model. Finally, we will discuss the issues surrounding developing and deploying business components that will utilise application-server technology.

## Requirements of enterprise applications

When an organisation tries to map clean logical enterprise models onto possible IT solutions it is primarily constrained by two things: the limitations and risks associated with current IT technology solutions, and it's existing IT architecture. The Internet is the basic enabling technology behind solutions that try to address these problems. Current Internet technologies such as HTML, JAVA applets, and CGI allow an organisation to make functionality embedded in its legacy systems directly available to customers. However these solutions do not scale up to address the following basic needs of an organisation that wishes to implement large scale inter enterprise computing:

- **Component Availability**: Allow a business to clearly expose and advertise its legacy systems functionality as a series of components in an electronic market place.
- **Component Interoperability:** Allow an organisation's components to seamlessly plug and play with other organisation's components to meet customer requirements.
- **Transaction Management:** Ensure transaction correctness, fault tolerance, and appropriate performance levels for transactions that may span several heterogeneous enterprise servers.
- **Support Rapid Development:** Allow an organisation to quickly develop and deploy new components and easily modify existing ones.

## Enterprise Application Models

An example of a component model which attempts to fully address the requirements of large scale Internet applications, is the Common Object Request Broker Architecture (Corba) /Enterprise Java Bean (EJB) model. Corba can be viewed as a series of specifications that define an infrastructure that allows heterogeneous components to interact with each other over the Internet. At the heart of the Corba is the Object Request broker (ORB). An ORB is an object bus that allows objects to interact with each other. For a Corba implementation to support large complex Internet applications it needs to also have the following two key components:

- Corba Internet Inter Orb Protocol (IIOP): This is the protocol that allows objects created using different Corba implementations to interact with each other.
- Object Transaction Manager (OTM): this is basically a Transaction Processor (TP) monitor, which is a piece of software that manages transactions on a server, linked to a Corba implementation. An OTM provides a CORBA implementation with the transaction services necessary to support complex, large transaction volume applications.

Corba provides the basic plumbing for enterprise computing, but it falls short of supplying a framework for organisations to use its technologies easily and appropriately. This is where Java steps in. Java is an object oriented language with the distinguishing characteristic that a piece of pure Java Code can be guaranteed to run on any machine that has a Java Virtual Machine (JVM) installed on it. The Corba specification allows CORBA objects to be created from Java classes. Java therefore brings implementation independence to CORBA objects, in that Corba/Java objects are able to run on any platform that has a JVM installed on it.

The JavaBean component model takes Java a step further in that it adds rules that allow Java developers to build classes that are toolable and reusable. Visual assembly tools are available that allow application developers to rapidly build applications by customising and composing JavaBeans. With the advent of JavaBeans much of the complexity of Corba programming has been removed and distributed object programming has been made accessible to corporate IT departments.

The JavaBean component model is primarily a client component model in that it does not scale up to high volume transactional systems. This is where Enterprise JavaBeans (EJBs) come into play; the EJB spec defines the interfaces between a server side component and its container. The container provides the infrastructure and services required to run a large transactional system. The developer of an EJB only has to concern themselves with building the business process logic and possibly some data access code.

Microsoft's Windows 2000 operating system, currently in beta, will contain COM+ a component model that will offer similar functionality to the Java component models. The COM+ component model is however not an open standard and its scalability will be open to question given that it is tied into the Windows 2000 operating system.

## Application Servers

An Application Server can be defined as "a set of utilities, services, frameworks, and ORBs designed to facilitate the execution of an internet-based component application." Most application servers are implementations to a greater or lesser extent of the CORBA/EJB model. To illustrate how leading software vendors are gravitating towards the CORBA/EJB model we give the following three examples of CORBA/EJB application servers:

- IBM's Component Broker is a product that has it roots in CICS, a mainframe TP monitor.
- Oracle application server 4.0 provides a CORBA/EJB implementation that will allow traditional client/server applications be transformed into components that will interact over the Internet.
- IONA's OrbixHome provides a good example of a pure CORBA implementation becoming more toolable and accessible to corporate IT departments.

## Business Objects

The CORBA/EJB model provides the framework we need to develop and deploy business components that will interact with each other over the Internet. We are however still left with the significant task of defining these components. This task is being tackled by the Object Management Group (OMG), the consortium which is responsible for the Corba specification, through special domain specific task forces. The OMG have defined the following domains: Business, Electronic Commerce, Finance, Life sciences, Manufacturing, Telecoms, and Transportation.

To use the Transport domain as an example, one of the goals identified in its task force mission statement is, "To establish a series of distributed object or component specifications based upon a Transportation Object Model that allow participation of system components constructed by different sources to interoperate in completing various transportation business domain scenarios." The product of such efforts as the OMGs domain task forces will be a series of domain specific frameworks. Companies that have components that play by the rules of a popular framework will gain competitive advantage. For example an airline, which uses components that play by the rules of a popular transport framework, will be able to:

- Create dynamic alliances with other transport organisations that use the framework. For example, instead of adding a new route to its schedule in order to compete with a rival on the route, it may alliance itself with a rail company to offer a feasible alternative.
- Make its components available to intelligent electronic agents: Frameworks will have associated intelligent agents that will be able to search through the components available to the framework and return solutions to particular problems. For example a person trying to produce a complex travel itinerary constrained by time and cost factors would use an agent to produce a list of possible solutions.
- Facilitate legacy data integration: Because organisations using the same framework will share a common data model, this should facilitate the integration of data from different organisations for data analysis purposes.

Other efforts at defining common business object include; the IBM San Francisco Model, and the Consortium for Business Object Promotion (CBOP) whose aim is define identify reusable business objects for Japanese companies.

## Application Development

So far we have discussed the models and implementations of distributed object architectures that support large applications, we will now look at the three basic types of application development that will be done in the future.

- Component Development: Object Oriented developers and domain specialists will come up with domain specific models. The models will be produced using an object modelling language such as UML. The models will then be implemented as JavaBeans and EJBs, that is customisable black box components.
- Component Assembly and customisation: Developers involved at this stage will customise and install the components into containers. They will be supported by visual tools which will allow them to treat the components as visual components at design time, even though at run time they may be invisible components, e.g. EJBs.
- Back-End Development: Development may need to be done at the legacy system end to allow the component to get the data it is looking for. For example, if the data in an operational system is in an old proprietary format, it might have to be transformed using some data transformation engine into a format that the component can be customised to handle.

Software Vendors must now be getting themselves in a position where they can handle the shift, that will happen in the next couple of years, from traditional client-server applications to Internet applications. They can do this by starting to reengineer their applications as components. If a vendor's components are to reach a wide market they must be able to plug and play with as many other components as is possible. This means that the components should be:

- compatible with the appropriate distributed object models: i.e. COM+ for windows based applications or CORBA/EJB for large Internet applications, and
- compatible with the appropriate domain specific models that are currently being developed by groups such as the OMG and CBOP.

Organisations who wish to be able to gain competitive advantage from the Internet should also be ramping up their knowledge of application-server technology. The scale of the work involved in exposing large transactional stand-alone systems to the Internet is not going to be trivial. However the payback in terms of competitive advantage, and the ability to establish far greater synergy between the business and IT make the move an imperative for most organisations.

## Further Information:

***Client/Server Programming with Java and Corba***, 2nd ed. Robert Orfali and Dan Harkey
John Wiley and Sons '98

***Programming with Enterprise JavaBeans***, *JTS and OTS*, Andreas Vogel, Madhavan Rangarao

***Objects, Components and Frameworks with UML***, *The Catalysis Approach*, Desmond F.D'Souza and Alan Cameron Wills, Addison Wesley '98

**EJB*:*** http://www.java.sun.com
**Corba*:*** http://www.omg.com
**EJB provider site:** http://www.mgm-edv.de/ejbsig/

Technical Briefing Notes are issued on a range of software engineering topics as an aid to software developers, project leaders and managers. The intention is to provide a 'status report' on the state of the art (and/or the state of practice) in relation to particular aspects of software engineering. In addition they aim to highlight, where appropriate, a likely roadmap on a time horizon for future developments of the technology.

Centre for Software Engineering Ltd, Dublin City University Campus, Dublin 9, Ireland.
Telephone: +353 1 7045750 Fax: +353 1 7045605 Email: admin@cse.dcu.ie