Component Technologies: Java Beans, COM, CORBA, RMI, EJB and the CORBA Component Model

Wolfgang Emmerich and Nima Kaveh
Department of Computer Science
University College London
Gower Street, London WC1E 6BT, UK
{w.emmerich|n.kaveh}@cs.ucl.ac.uk

ABSTRACT

This one-day tutorial is aimed at software engineering practitioners and researchers, who are familiar with object-oriented analysis, design and programming and want to obtain an overview of the technologies that are enabling component-based development. We introduce the idea of component-based development by defining the concept and providing its economic rationale. We describe how object-oriented programming evolved into local component models, such as Java Beans and distributed object technologies, such as the Common Object Request Broker Architecture (CORBA), Java Remote Method Invocation (RMI) and the Component Object Model (COM). We then address how these technologies matured into distributed component models, in partiuclar Enterprise Java Beans (EJB) and the CORBA Component Model (CCM). We give an assessment of the maturity of each of these technologies and sketch how they are used to build distributed architectures.

1. INTRODUCTION

Object-oriented technologies have had a large impact on industrial software development process. Features of OO methodologies such as encapsulation and abstraction lead to a more flexible and extensible software product. Object-oriented programming, however has failed to deliver its expectations with respect to productivity gains and in particular the creation of reuse. Component technologies build on this foundation to provide third-party components that isolate and encapsulate specific functionalities and offer them as services in such a way that they can be adapted and reused without having to change them programatically. Components usually consist of multiple objects and thus have a larger granularity. This characteristic enables them to combine functionalities of the objects and offer them as a single service.

2. TUTORIAL STRUCTURE

This tutorial provides a classification of the existing key technologies in component-based and distributed systems. We explore the relationships between these technologies and the ways in which some have been combined together. Starting from object-oriented programming we outline the different development strands of component technologies, which are:

1. Local component technology
2. Distributed Object technology
3. Distributed component technology

During the tutorial, we discuss examples technologies in each of these classes and demonstrate how they can be used to build applications.

2.1 Local Component Technology

Component models provide a mechanism by which software engineers can develop applications by composing components through their well defined interfaces rather than developing new or changing existing components. By acquiring components from third-party providers the process of development becomes less time consuming and raises the level of implementation abstraction. Moreover using components that have previously been deployed successfully should reduce bugs in the software.

Local component technologies can only operate in an intra-process fashion. Therefore all components must reside on the same host at execution time.

Two of the main component technologies include JavaBeans of Sun Microsystems [6] and ActiveX [1] of Microsoft, based on their Component Object Model (COM). The tutorial will focus mainly on JavaBeans. The concept behind JavaBeans is to enable the composition of components, known as beans, through a composition tool (Bean Box). In this way developers can simply customise imported beans via a palette of properties. Bean components communicate by sending events to one another using a publish/subscribe model. In order for the Bean Box to extract the properties, methods and events supported by a component, JavaBeans uses an introspection mechanism. Introspection, a weak type of reflection, is achieved in two ways. Firstly there is a strict guideline for naming the properties and methods of a bean during implementation. Secondly a bean provider can explicitly provide information about the

feature of a bean using a standard API and then Java's reflection package can be used.

2.2 Distributed Object Technology

Object middleware enables communication between multiple objects that reside on different networked hosts (distributed objects). The communication primitive between a client object and a server object is a method invocation. Middleware is responsible for providing transparency layers that deal with distributed systems complexities such as location of objects, heterogeneous hardware/software platforms and numerous object implementation programming languages. Shielding the application developer from such complexities results in simpler design and implementation processes.

Besides the core task of transparently relaying object invocations, some middleware technologies offer additional services to the application developer. Examples of these services include security, persistence, naming and trading.

The tutorial will look at three mainstream object middleware technologies used in industry. These are the Common Object Request Broker Architecture (CORBA) [2, 5], Java Remote Method Invocation (RMI) [3] and Microsoft COM. Java RMI is most suitable when the system purely consists of Java objects that need to communicate on a distributed system. The close integration of Microsoft's COM technology with its other products makes COM the best choice for a Microsoft based environment. The CORBA specification is the result of input from a wide range of OMG members, making its implementations the most generally applicable option.

We compare and contrast the object models of these object middleware systems and discuss scenarios in which different types of middleware may need to be integrated. In order to achieve interworking between different types of middleware system, object model mappings and interaction protocols need to be provided.

2.3 Distributed Component Models

The motivation for distributed component models partly originate from deficiencies present in available OO middleware technologies. Designs and implementations of applications using middleware invariably have a large focus on middleware which can cause a distraction from the main business problem at hand. Moreover experience has shown that integrating existing middleware technologies is cumbersome and often a source of additional complexity.

Distributed component technologies combine the characteristics of components with the functionality of middleware systems to provide inter-process communication between components. That is to say components that can communicate across machine boundaries.

The tutorial will discuss two distributed component technologies namely, Enterprise Java Beans (EJB) [4] of Sun Microsystems and the CORBA Component model. These are both server-side component models used for developing distributed business objects. These are used on the middle-tier application servers that manage the components at run-time and make them available to remote clients.

Both EJB and CORBA components operate by providing components with a container in which they can execute. Each component provides an interface used for life-cycle operations such as creation, migration and destruction, as well as the remote interface it supports. Compliant containers all support the same set of interfaces which means that components can freely migrate between different containers at runtime without the need of reconfiguration or recompilation. Containers themselves run on application servers, which offer services offered by the underlying middleware systems such as transactions, security, persistence and notification.

The CORBA component model was developed to provide a distributed component model for use with programming languages other than Java and at the same time achieve interoperability with EJB components. However its also motivated by providing an open standard that is not controlled by individual vendors. Currently there are more than 30 implementations of EJB servers and that number is increasing. CORBA component implementations are underway but have not yet been released.

3. WHO SHOULD ATTEND

This full-day tutorial is aimed at both industrial and academic participants, who wish to get an overview of the local and distributed component technologies that are currently available. We assume that participants are familiar with object-oriented programming concepts.

4. ABOUT THE INSTRUCTORS

Dr. Wolfgang Emmerich is a co-founder, Director and Chairman of Zuhlke Engineering (UK) Ltd. He has authored a Wiley text on "Engineering Distributed Objects". Wolfgang Emmerich has delivered numerous tutorials at international software engineering conferences. He is also a Senior Lecturer in Computer Science at University College London. Nima Kaveh investigates model checking techniques for distributed objects and components during his PhD studies at UCL.

5. REFERENCES

[1] D. Box. *Essential COM*. Addison-Wesley Publishing Company, 1998.

[2] W. Emmerich. *Engineering Distributed Objects*. John Wiley & Sons, Apr. 2000.

[3] JavaSoft. *Java Remote Method Invocation Specification*, revision 1.50, jdk 1.2 edition, Oct. 1998.

[4] R. Monson-Haefel. *Enterprise Java Beans - 2nd Edition*. O'Reily, 2000.

[5] Object Management Group. *The Common Object Request Broker: Architecture and Specification Revision 2.3*. 492 Old Connecticut Path, Framingham, MA 01701, USA, December 1998.

[6] *JavaBeans Component Architecture Specification Revision 1.0.1*, 1998.