Panel Report:

Development of Distributed and Client/Server Object-Oriented Applications: Industrial Solutions

Lutz Heuser, Digital Equipment GmbH

Introduction

Computer networks have been introduced in almost all areas of business. Moreover, connecting local area networks through wide area networks has become quite popular. However, the software running on such an infrastructure still seldom use it. Most often the problem is the lack of expertise, on the part of the application programmers, in writing such applications. The IT industry as well as computer scientists have investigated and evaluated approaches that should help these programmers to make better use of the networks. This panel discussed certain solutions that help to better utilize the network. DSOM [1], OODCE [2], and COM [3] were the systems presented by the panelists.

The remainder of this report describes the outcome of the panel session. In order to get a complete picture I recommend to read the panel paper published in the OOPSLA'94 proceedings [4] first.

Summary of Presentations

The panel started with bad news, unfortunately. Steve Rabin was not able to attend the panel session. Since no replacement had been appointed, his position statement was not presented.

I started with an introduction statement where I emphasized the purpose of the panel, i.e. that application programmers should get some of their questions answered around development of distributed and client/server applications. The taxonomy of distribution was discussed to make the audience aware of the fact that the "client/server" approach is not the same as the "peer-to-peer" approach. Major characteristics of distribution were presented such as *locality* of objects and the *environment* of the application, i.e. external constraints that require different locations of objects. *Interoperability* with other applications in the network was also mentioned as well as *integration* of existing applications into new ones and the issue of *mobility*, i.e. moving objects between different locations.

Another topic of the discussion was the *level of granularity* to be supported. Should application-level objects such as documents be treated like programming-level objects such as C++-objects? Is there one unique approach that can be used or are the requirements different? Which of the existing approaches can be seen as a de-facto standard? The panelists addressed these questions during their position statement as you can see below.

The situation application programmers are faced today with is as follows: The IT-networks are complex and heterogeneous such that different operating systems and network architectures are used within a given enterprise. PCs entered the office and became the most favorite desktop environment with various server systems at department level. Recently, a number of new operating systems showed up that increased the complexity of systems and network management. Moreover, the so-called middleware, i.e. API-based services, became so complex that application programmers could not keep pace. As a result, many features resp. services are poorly used today mainly because of missing expertise. This holds in particular for distributed applications. Therefore, the following requirement was stated: Any given approach to support the development of distributed applications should lead to extendible, maintainable and (partially) reusable software. I concluded my introduction with the statement that application programmers should enforce the development of (de-facto) standards that really work.

First panelist to address the issues mentioned was Hari Madduri from IBM. He is one of the principle architects of SOM resp. the distributed version DSOM. Hari started with describing the criteria of *industrial solutions*:

- An industrial solution has to follow (de-facto) *standards*.
- It has to be *language neutral*, i.e. the distribution service should enforce the use of a particular language.
- The service has to be *open* and has to handle *evolution*.
- It has to deal with legacy applications.
- The industrial solution needs to address *development* and *deployment* needs.

Hari stated that he believes that industrial solutions for distribution should be based on class libraries and frameworks. Distributed programming is hard to do and distributed languages have not caught on. Whereas class libraries can hide nasty aspects of distribution and can extend a language "ad infinitum". Today's problems with class libraries are that they cannot be shipped in binary form, they do not support multiple languages and multiple compilers. Hari stated that SOM is braking these barriers.

SOM has a robust object model and provides an open architecture. DSOM is the distributed extension of SOM supporting location transparent object invocation, flexible class frameworks and wrapping of legacy software.

Next panelist was Shawn Woods from Digital. Shawn is working on the integration of Microsoft's OLE V2 and CORBA called COM (Common Object Model). He agreed to Hari that an industrial solution should be language neutral, open and extendible, and the libraries should be shipped as binaries. Re-compilation of applications should be avoided in case of new versions of class libraries. Technical features of COM include *scaleable typing*, i.e. strong typing with "mixed versions", seamless distribution, and a scaleable class model. Shawn also pointed out some business aspects of COM such as market size, proven technology, and the ability to add value by building library extensions.

Finally, John Dilley from HP presented his position statement. Different to the two before, he was not in favor of CORBA and criticized the outcome of the CORBA V1 specification. He presented OODCE, a new product of HP that relies on OSF-DCE, a set of services that include remote procedure call, threads, security, naming and time services.

Goals of OODCE were the support of software development using standard DCE and C++, the encapsulation of functionality, and the provision of distributed objects using the DCE object model. They extended the interface description language such that C++ class are generated rather than C functions. John stated that the advantages of OODCE include simpler development of distributed applications, interoperability with other DCE-based applications and the fact that no C code has to be written.

Questions and Answers

In this section the questions of the audience are reported as well as the answers of the panelists. In order to focus no transcript of the session is given but the essentials are highlighted.

Q: How does security play a role?

John: Secure communication is essential. DCE includes Kerberos.

Hari: Kerberos is not the only way to go. Security can be also supported by class libraries.

Shawn: Security should go into the object request broker specification.

Q: How are highly scaleable distributed applications supported?

Hari: Good architectures and class libraries are needed.

Shawn: Middleware support is not enough. Class libraries have a role in this.

John: DCE already has some support for scalability. We do not need to start from scratch.

Lutz: Object-oriented design leads to peer-to-peer rather than client/server solutions. That helps to scale application.

Q: What about fine grained distribution?

Hari: DSOM can handle fine grained distribution.

Shawn: Any solution need to support fine grained objects. It is needed to move small objects around respectively to decide on a per object base.

John: We need to tune for lots of small objects. DCE has some problems here. The semantics of object passing is not understood well. Passing object references is not the right solution for small objects but object migration has too much overhead.

Q: Are there any plans for adding transaction monitors?

The panelists support the idea but none of them knew of any plans about this.

Q: How does object interoperability across languages work?

John: This is done by the transport layer and the IDL description.

Hari: Through the binary standard of DSOM.

Shawn: You cannot map one language to another. Coming up with an object model and maps it to languages means you might not be able to use all of the features of a given language.

Q: How do you handle network separation or disconnection of nodes?

Shawn: COM handles this for you.

Hari: DSOM does it by proxies.

John: A hard problem is to handle reference counters, i.e. for garbage collection. DCE contexts can deal with some of the aspects.

Shawn: Reference counting and garbage collection need substantial assistance from the system. Cyclic references are a problem to address.

Conclusion and Outlook

Distribution has become an issue to be addressed by vendors and application developers due to the increasing network infrastructure. Communication inside local area networks has become as popular as using wide area networks such as Internet. Therefore, users demand that their applications make easily use of new services such as WorldWideWeb. In order to do so, application programmers need adequate industrial solutions to support the characteristics of distribution mentioned earlier.

The panel addressed the requirements based on two different approaches: CORBA and DCE. While the CORBA-based approaches easily incorporate the object into their models of distribution they suffer from the fact that "CORBA-compliance" is no guarantee that applications on heterogeneous networks will work. In fact, the OMG and its members are currently working on CORBA V2 to overcome such shortcomings like missing interoperability of two and more ORBs based on the same CORBA specification. OODCE overcomes the interoperability constraint by relying on OSF-DCE which supports interoperability even with none object-oriented applications. But, DCE has been specified without an object-oriented approach in mind, it is procedural oriented. Therefore, many features such as inheritance, encapsulation, and object integrity are not addressed and need to be added by upper layers.

There was a common understanding by all panelists that any solution has to be language independent. Hari stated that distributed object-oriented languages were good to learn from but would be inadequate as industrial solutions. Based on my experiences with DODL [5] and DOWL [6], a design language and an extension of Trellis [7], I agree with Maduri but language specific library extensions and tools as in our recent system, DC++ [4] (distributed C++), are beneficial for those application developers that could concentrate on one language such as C++. The same should hold for OODCE.

In summary, industrial solutions for distributed object-oriented applications are available but they have to become more mature and more widely used. CORBA V2, OLE V2, DSOM and OODCE seem to have the best chances to overcome the shortcomings mentioned. The solutions are there. What is needed is a common effort that combines the best features. I know that sounds like an unrealistic dream but -as stated earlier- application developers should require merging the efforts of the vendors for the sake of simplicity. If the vendors will fail in this, developers of distributed applications might step back from the object-oriented approach.

Acknowledgment

I like to thank Michael Kilian for taking notes during the panel session. I also like to thank Pat O'Brien and Ira Forman for their valuable comments and help.

References

[1] I.R. Forman, S. Danforth, H. Madduri, Composition of Before/ After Metaclasses in SOM, OOPSLA'94 proceedings, 427ff, October 1994

[2] J. Dilley, Object-Oriented Distributed Computing With C++ and OSF DCE, DCE - The OSF Distributed Computing Environment, A. Schill (ed.), Lecture Notes in Computer Science, Springer, 256ff, October 1993

[3] K. Brockschmidt, Inside OLE2, Microsoft Press, 1994

[4] L. Heuser, J. Dilley, H. Madduri, S. Rabin, S. Woods, Development of Distributed and Client/Server Object-Oriented Applications: Industry Solutions, OOPSLA'94 proceedings, 317ff, October 1994

[5] M. Mühlhäuser, W. Gerteis, L. Heuser, DOCASE: A Methodic Approach to Distributed Programming, Communications of the ACM, Vol. 36, Nbr. 9, 127ff, September 1993

[6] B. Achauer, The DOWL Distributed Object-Oriented Language, Communications of the ACM, Vol. 36, Nbr. 9, 48ff, September 1993

[7] T. Cooper, B. Bullis, M. Kilian, C. Wilpolt, C. Schaeffert, An Introduction to Trellis/Owl, OOPSLA'86, 9ff, 1986