

Component Migration with Enterprise JavaBeans

Michael Richmond

Information and Communication Sciences

Macquarie University

Sydney, Australia

+61 2 9850 6346

mar@ics.mq.edu.au

1. INTRODUCTION

Currently available component infrastructure frameworks such as CORBA[3], DCOM[6] and Enterprise JavaBeans[7] support distributed applications by providing a transparent communication mechanism between components across hosts. Existing distribution support implicitly prevents a component's re-distribution after binding: these frameworks prevent a component from being moved once the application has started running.

Distributed applications often have a relatively long execution time so they need to adapt to changing application loads and usage patterns. Such applications require the ability to move components at any time during the application's execution.

2. DISTRIBUTION SUPPORT

Without dynamic support for these changes it is necessary to restart the application when structural changes are made. We have developed the following taxonomy[4] to consider the possible alternatives in specifying the location of components in an application:

Static configuration

In a statically configured system, each component's location is specified before the system begins running. For example, the location of DCOM components are defined by entries in the Windows system registry.

Dynamic configuration

Dynamically configured distributed applications rely on the underlying framework to determine each component's location. In DCOM and CORBA, this type of distribution is achieved through the use of *trader services*. A trader accepts requests for component instances, servicing the requests with references to existing components, or by creating new components in the system. Once bound, the component cannot be moved without halting the system.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

OOPSLA 2000 Companion Minneapolis, Minnesota

© Copyright ACM 2000 1-58113-307-3/00/10...\$5.00

Dynamic relocation

Unlike the previous alternatives, dynamic relocation allows components to be moved at any time without halting the system. This allows the system to be restructured to take advantage of changing system loads and adapt to varying user requirements.

The initial configuration of a system using dynamic relocation is specified by either of the preceding methods, or a combination of both.

Each of these options requires an increasing level of support from the underlying component framework. Statically configured applications require support to instantiate components on remote hosts according to a specified application topology. For dynamically configured applications the framework must support some way of algorithmically determining the best location for each component. Typically, this is based on the load of individual hosts in the system, known behaviour of the application from previous executions and the existence of other instances of the required component already in the system.

Dynamic relocation, the third category in our classification, requires the component infrastructure to support some form of component migration.

3. COMPONENT MIGRATION

We define component migration as the movement of a component instance from one host to another in a distributed system. Component migration can be used to improve the performance and flexibility of an application in a number of areas. For example:

Application load balancing

In a distributed application the transaction load on given components and hosts often vary significantly. In such cases a component can be migrated from a host with a relatively high load to one under a lower load to improve the application's performance. In this way, we can improve the overall system performance.

Application restructuring

A distributed application's usage pattern and processing load are likely to change over the lifetime of an execution. For many distributed applications, it may be preferable to restructure the application to reflect these changes without halting its' execution.

Possible changes to a distributed application's structure, which will benefit from component migration, include those resulting from consolidation of processing nodes in the system, re-inte-

gration of stand-alone distributed components, and introducing component replicas to the system.

Component survivability

At times a host's impending shutdown or removal from a distributed application's pool is known in advance (such as for periodic maintenance, or host's allocation to other tasks). In these cases, a component can be migrated to another host to ensure that the component survives the shutdown or removal of its current host from the application.

Reducing network costs

In some situations it is preferable to relocate a relatively small component to the site of 1) a resource which cannot be moved (such as a particular device), or 2) a resource whose movement would cause more traffic than migrating the component (such as a large database).

When the component no longer requires the resource it can be migrated back to the original host. This is similar to data collection agents[2] in effect, with the exception that control of the component's movement is maintained outside of the mobile entity.

Decisions such as when to migrate a particular component and to which host are system policy decisions that are outside the scope of our work. We need to establish an understanding of how component migration is used in practice before these policy issues can be addressed. Through the study and use of component migration in real applications we plan to determine which of these policy decisions are best automated and which are better left under user control.

Several concepts used in component frameworks, such as transparent remote method calls, global name spaces, and group-wise communication, originate from distributed operating system research. This community also investigated the idea of moving an executing processing entity from one host to another. Their goal, with *process migration*, was to migrate a process from one host to another.

Process migration[1][5] has been criticised as being a solution looking for a problem. The requirement for process level location transparency is one of the main hindrances to wider acceptance of process migration in general purpose systems. Component frameworks, however, have been designed to support location transparency from very early in the design process. The lower cost/benefit ratio associated with introducing migration to component based systems may mean that component relocation is the problem that process migration techniques have been looking for.

4. CONCLUSION

Component technology promises to simplify the design and imple-

mentation of distributed applications by encapsulating dependencies within components and furnishing programmers with distribution support. In current component frameworks, this support takes the form of making component communication and distribution mechanisms transparently available across multiple hosts.

We have presented a distribution support classification which introduces component migration to support the restructuring of a system without halting its execution. In this way, high system availability can be maintained while providing the flexibility adapt to changing requirements.

Our ongoing work involves extending the Enterprise JavaBeans component framework to support component migration. We expect our work to address the following open issues:

- What changes to the EJB container and API does migration require?
- What levels of programmer control and use of migration are desirable?
- Does EJB need versioning at the class or Bean level?
- Is deep component migration necessary?
- How can the Java RMI API be extended to support relocating objects?

5. REFERENCES

- [1] Douglass, F. & Osterhout, J., "Transparent Process Migration Design Alternatives and the Sprite Implementation", *Software - Practice and Experience*, 21(8), pp. 757-786, August, 1991.
- [2] Harrison, C., Chess, D. & Kershenbaum A., "Mobile Agents: Are they a good idea?", Technical Report, IBM T.J. Watson, 1995.
- [3] Object Management Group. "The Common Object Request Broker: Architecture and Specification. Revision 2.2.", Massachusetts, February 1998.
- [4] Richmond, M. "Support for Dynamic Distribution in Component Systems", *Workshop on Component-Oriented Software Engineering '98*, Adelaide, Nov. 1998.
- [5] Richmond, M. & Hitchens, M., "A New Process Migration Algorithm", *Operating Systems Review*, vol. 31, no. 1, January 1997, pp. 31-42.
- [6] Roy, M. & Ewald, A., "Inside DCOM", DBMS, April 1997, pp. 27-34.
- [7] Sun Microsystems, "Enterprise JavaBeans Specification v1.1", Sun Microsystems, Palo Alto, California, March, 2000.