Middleware Unaware Software Development and Interoperability using MDA

Nelly Bencomo, Gordon Blair

Computing Department, Lancaster University,
Bailrigg, Lancaster, LA1 4YR, UK
nelly@acm.org, gordon@comp.lancs.ac.uk

Abstract. The main interest of this position paper is how to separate the development of distributed applications from specific middleware technologies. Unfortunately, the large number of middleware technologies conspires against this purpose – the development and maintenance of distributed systems have become coupled to the constant evolution and changes of middleware technologies. Authors back the idea that the Model Driven Architecture (MDA) gives the basis to tackle this problem. Our proposed research focuses on interoperability among applications relying on different platforms and the necessity that transformations and mapping of concepts between different PSMs should be address according the context given by the Domain Problem.

1. Introduction

The goal of middleware is to provide an integration means for diverse computing platforms. Middleware makes the development of distributed applications much easier, providing the abstractions to cope with distribution and its coordination. Currently, we have different middleware technologies such as CORBA, Java/RMI, EJB, Jini, Web Services (XML/SOAP) and .Net. All share the purpose of given the infrastructure for distributed application development by providing abstraction over the complexity and heterogeneity of the underlying distributed environment with its multitude of network technologies, machine architectures, operating systems and programming languages [6]. Unfortunately, the proliferation of middleware technologies has brought a new difficulty to distributed software development – the constant change and evolution of middleware technologies. Software Engineers are then challenged both in the area of development of new and scalable middleware systems, where open and adaptable platforms should offer richer functionality and services, and in the area of application development, where developers have to worry about constructing distributed applications that are able to evolve with the underlying middleware technologies. Our interest focuses on the latter challenge, how to manage the differences among Middleware Technologies when developing software applications, where the main idea is to study how software development can be carried out unaware of middleware concerns. The Model Driven Architecture (MDA) has been proposed as a good solution for this research problem. MDA applies the basic principle of separation of concerns by separating the specification of the system functionality from its specification on a specific platform. The former is defined as a Platform Independent Architecture (PIM), the latter as Platform Specific Model (PSM). The mapping from PIM to PSMs is performed using transformation rules. Interoperability among applications relying on different platforms can be realized by tools that not only generate PSMs, but the bridges between them. We support the idea that bridges need to focus on the context of specific Domain Problems. Our proposed research focuses on the identification of pertinent Domain Problems and the consequent definition of mappings and transformation between the abstractions of different Middleware Technology Models.

2. Middleware technologies à la carte

CORBA, Java/RMI, EJB, Jini, Web Services (XML/SOAP) and .Net. address, in general, the same problems but with different approaches. For example, the Common Object Request Broker Architecture (CORBA) is the Object Management Group's specification for achieving interoperability between distributed computing nodes. Their objective was to define an architecture that would allow heterogeneous environments to communicate at the object level regardless of who designed the two endpoints of the distributive application. A cornerstone of CORBA is its support for multiple programming languages like C, C++, Java, COBOL, Smalltalk, and Python. The CORBA standard includes mappings from IDL for each

supported programming language [5]. Currently Web Services present another alternative distributed computing infrastructure; an alternative that is being strongly promoted (commercially and from the point of view of research) as preferable to the use of distributed object middleware such as Java RMI or CORBA. This new distributed computing solution exploits the openness of specific Internet technologies to address many of the interoperability issues of CORBA and other former solutions.

For years it was assumed that a clear winner would emerge and stabilize this state of flux, but the time has come to admit openly: The string of emerging contenders will never end! And, despite the advantages (sometimes real, sometimes imagined) of the latest middleware platform, migration is almost always expensive and disruptive [9]. On the other hand, companies have to preserve their software investments as the middleware landscape changes underlying it.

3. MDA: a solution

The first step in MDA is to construct a model with a high level of abstraction that is independent of any middleware technology, obtaining the Platform Independent Model (PIM). Within a PIM, the system is modeled from the viewpoint of how it best supports the business [2]. Whether the system is going to be implemented using CORBA, Java/RMI or Web Services technologies plays no role in a PIM. In the next step, the PIM is transformed into one or more Platform Specific Models (PSMs). In our specific case, the PIM is mapped (transformed) to one or more Middleware Technologies Models via OMG Standard Mappings. This transformation might be made by a MDA tool that applies a standard mapping to generate a PSM from the PIM. Depending on the tool, code production will be partially automatic, partially hand-written. Finally, each PSM is mapped (transformed) to code. Because a PSM fits its technology rather closely, this transformation is relatively straightforward [2].

4. Interoperability: mapping between the concepts

One important aspect to take into account is interoperability of applications that use different middleware technologies. This is achieved in MDA using bridges between PSMs. It is necessary to transform concepts from one platform into concepts used in another platform. The results of these transformations will be used to construct the bridges between the PSMs. If we are able to transform one PIM into two PSMs, all the information we need to bridge the gap between the two PSMs is available [2]. For each element of one PSM we know from which element in the PIM it has been transformed. From the PIM element we know what the corresponding element is in every PSM. We can therefore deduce how elements from one PSM relate to element in other PSMs. So, we have all the information we need to generate a bridge between every pair of PSMs.

Interoperability among applications relying on different platforms can be realized by tools that not only generate PSMs, but the bridges between them. The idea of the OMG is that transformation definitions should be in the public domain, perhaps even standardized and tunable to the individual needs of its users [2].

5. Proposed Research

We support the idea that bridges need to focus on the context of specific Domain Problems instead of using proposed generic PSM-to-PSM transformations [3]. We think that a standard bridge between CORBA and Web Service applications, for example, is difficult, if not impossible to develop. Bridges should address different Domain Problems, for example, Banking, E-commerce, Telecommunications, etc. PSMs must model the target platforms with sufficient precision. The use and definition of general-purpose concepts might lead to unsuccessful results due to their latent complexity. We are investigating a formal definition of Domain Problems to consequently start defining mappings and transformation between the abstractions of different Middleware Technologies.

References

1. Gray N.A.B.: Comparison of Web Services, Java-RMI, and CORBA service implementations, Fifth Australasian Workshop on Software and System Architectures, Melbourne, Australia, April, 2004
2. Kleppe A., Warmer J., Bast W.: MDA Explained The Model Driven Architecture: Practice and Promise, Addison-Wesley, 2003
3. Kovse J.:Generic Model-to-Model Transformations in MDA: Why and How?, Workshop Generative Techniques in the Context od Model Driven Architecture, OOPSLA 2002
4. Mellor S., Scott K., Uhl A., Weise D.: MDA Distilled Principles of Model-Driven Architectures, Addison-Wesley, 2004
5. Vinoski S.: It's just a mapping problem, IEE Internet Computing, pp. 88-90, May/June 2003
6. http://dsonline.computer.org/middleware/
7. http://www-106.ibm.com/developerworks/webservices/library/ws-arc3/
8. http://www.xs4all.nl/~irmen/comp/CORBA_vs_SOAP.html
9. http://www-106.ibm.com/developerworks/rational/library/403.html