Applying MDA to P2P Platform

José Geraldo de Sousa Junior and Denivaldo Lopes

1 Federal University of Maranhão - UFMA
Campus do Bacanga
São Luís - MA - Brazil
jgeraldo@dee.ufma.br and dlopes@dee.ufma.br

Abstract. In order to support the creation of P2P applications, some frameworks have been proposed such as JXTA. Model Driven Architecture (MDA) can support the management of the complexity in the software development process through transformations of Platform Independent Models (PIM) into Platform Specific Models (PSM). In this paper, we apply an MDA approach to allow the development of applications based on JXTA.

1. Introduction

One of the most subjects in investigation on MDA is the transformation from Platform Independent Models (PIMs) to Platform Specific Models (PSMs) and the transformation from PSMs to source code [Frankel and Parodi 2002, QVT-Merge Group 2005]. In this paper, we aim to present how we can develop P2P applications according to an MDA approach.

This paper is organized in the following way. Section 2 describes the use of MDA to develop JXTA applications. Section 3 presents a case study that consists in the parallel/distributed array multiplication using P2P networks. Section 4 presents the transformation steps from our PIM (that conforms to UML) to a PSM (that conforms to Java+JXTA). Finally, in section 5, we conclude this paper presenting final considerations and future research directions.

2. Applying an MDA Approach to Develop P2P Applications

Figure 1 presents an MDA approach applied to JXTA platform. In this figure, the PIM is described using the UML metamodel[OMG 2003], and the PSM is described using the Java metamodel 1 and JXTA template. We need match the UML metamodel and the Java metamodel, and we need create transformation definitions conform to this matching. Afterwards, these transformation definitions must be incremented in order to apply the JXTA template for creating the PSM conform to Java metamodel and JXTA template. In order to realize the model-to-code transformation, we define a transformation from a PSM (in Java+JXTA) to a Java+JXTA code and configuration files.

1In this research, we use the Java metamodel proposed in [Lopes 2005]
3. Case Study: Array multiplication

The parallel/distributed array multiplication using a P2P network can be described in the following way. First, we must read both $A_{i,j}$ and $B_{i,j}$ arrays. Afterwards, we divide the array $A_{i,j}$ in $n$ sub arrays (let $n = \sqrt{m}$, where $m$ is the total of computers). Then, array $A$ is divided in $i/n$, i.e. $A_{i/n,j}$ sub arrays, and array $B$ is divided in $j/n$, i.e. $B_{i,j/n}$ sub arrays. Thus, $i$ and $j$ must be integer numbers that are multiple of $n$, in order to have sub arrays with the same dimension.

Figure 2 (a) illustrates an example of array multiplication using a P2P configurator. $A_1$ and $A_2$ are sub arrays of $A$, $B_1$ and $B_2$ are sub arrays of $B$, and $R_1$, $R_2$, $R_3$ and $R_4$ are sub arrays of the result array.

The coordinator is responsible to make the division of the array $A$ and $B$ and to request four peers (named Worker peers) on P2P network for doing the sub array multiplications. Afterwards, each peer returns its result. Once those four peers calculated the sub array multiplications, the final result is constituted with the composition of the results of each worker peer.

Figure 2. Array multiplication: (a) a P2P configuration and (b) Class diagram: a PIM (fragment)

The PIM of Figure 2 (b) has a class Operations that is responsible for some special array operations such as array reading and multiplication.

Figure 3. JXTA Template for Java (fragment)
4. Transformation from UML to JXTA

In this section, we present a transformation definition from UML to JXTA platform. In this research, we have used the Java metamodel provided in [Lopes 2005].

Figure 3 shows a JXTA template for Java applications using UML notation. There we can think about Java model\(^2\) as following: each \texttt{JClass} that implements a \texttt{<< Peer >>}, in this case \texttt{MyPeerJxta}, must extend a \texttt{JClass} named “PeerJxta”. In our approach, the JXTA platform information is present in the template and it is added through a transformation definition that relates an UML metamodel to a Java metamodel including the JXTA template in order to create a PSM.

\textit{PeerJxta} class must have all \texttt{JMethods} specified in the template, as for instance, \texttt{startJxta}, \texttt{sendMessage} and \texttt{receiveMessage}. Pay attention that implementations of these methods are necessary for a class which is a peer, then we constructed a class \textit{PeerJxta} with those methods and attributes that are essential for the communication among peers.

Listing 1 presents an ATL\[^3\] transformation rule\(^3\) that transforms UML classes to Java+JXTA classes.

```
list 1. Transformation definition in ATL - UML Class to Java Class + JXTA
1  rule C2JXC { 
2    from c: UML!Class(c.stereotype->select(e|e.name = 'Peer')->notEmpty()) 
3    to jxc: JAVAM!JClass ( 
4       name <- c.name, 
5       super <- peerC, 
6       ***{other bindings} 
7    }, 
8    peerC: JAVAM!JClass { 
9       name <- 'PeerJxta', 
10      visibility <- #public, 
11      super <- JAVAM!JClass, 
12      ***{other bindings} 
13    }, 
14    startJ: JAVAM!JMethod ( 
15       name <- 'startJxta', 
16       owner <- peerC, 
17      visibility <- #public, 
18      ***{other bindings} 
19    ), 
20      ***{other bindings} 
21 }
```

In ATL rule structure, we have filters (see lines 2-3) that are applied to the variable “c”. The filter describes basically that, if the expression between parentheses is true, then the rule must be executed. The transformation rule in Listing 1 transforms a class from UML model which has the stereotype “Peer” to a Java class according to the template (Figure 3).

In ATL language, \textit{queries} have functionalities as navigating on models enabling creation of outputs in textual form. Through this functionality we can generate source code from a model. \textit{Queries} iterate through packages, classes and interfaces from PSM (Java+JXTA model), extracting information from this PSM, to write them in specific files which contain source code and documents.

\[^2\]We use UML graphical notation with Profiles to express a Java model.

\[^3\]In order to simplify the presentation of this paper, we provide only fragments of transformation definitions, PSM in XMI format and source code.
Listing 2. Operation class generated by query program

```java
package solve;

public class Operacoes {
    public int matrizB[][];
    public int matrizA[][];

    public void readMatrix() {
    }

    public Object setParts(int order) {
    }

    public MatrizObject result(MatrizObject matrizes[]) {
    }
}
```

In Listing 2, we have a fragment of generated source code from the class *Operations* that is mentioned in Figure 2.

5. Conclusion

In this paper, we presented our approach to develop P2P applications through models. Once we created the transformation definitions, we can apply them to different PIMs conform to UML metamodel and obtain the source code for JXTA platform. Thus, the time to develop software applications using an MDA approach is more profited in modeling task than in coding task. Moreover, as the final source code is generated from a model, some prone errors of coding task can be avoided. The results obtained in section 4 demonstrate that an MDA approach can be used to develop P2P applications from UML models.

Acknowledgments

The work described in this paper was financed by Fundo Setorial de Tecnologia da Informação (CT-Info), MCT, CNPq (CT-Info/MCT/CNPq).

References


