Application MDA in a Conception Design Environment

Wuzheng Tan[1], Lizhuang Ma[1], Jun Li[1], and Zuxiu Xiao[2]
[1]*Department of Engineering and Computer Science, Shanghai Jiao Tong University, 200240, China*
[2]*Education Technique Center, Yu Lin Normal University, 537000, China*
tanwuzheng@sjtu.edu.cn; ma-lz@cs.sjtu.edu.cn; jleesr80@163.com; xiaozuxiu@yahoo.com.cn

Abstract

This paper proposes a conception Design environment that intends to support service collaboration. In this platform, technology independence is a very important goal to be achieved. Model Driven Architecture and metamodels are some of the resources to provide such independence. This article offers a modified software development process that would leverage MDA, and studies a web application case.

1. Introduction

Model-Driven Architecture (MDA) provides a framework for software development that uses models to describe the system to be built. The system descriptions that these models provide can be expressed at various level of abstraction, with each level emphasizing certain aspects or view points of the system.

Building enterprise-scale software solutions has never been easy. The difficulties of understanding highly complex business domains are typically compounded with all the challenges of managing a development effort involving large teams of engineers over multiple phase of a project spanning many months. In addition to the scale and complexity of many of these efforts, there is also great complexity to the software platforms for which enterprise-scale software are targeted.

For a successful conception design initiative, an open and evolutionary platform must be considered in order to provide means to enable the old world of the legacy systems accessible to the new facilities brought by Internet. The

Web Service (WS) architecture delivers standards for such collaborative environment. Although a good reference, the WS specifications, and the technologies to implement them, are still in evolution. To preserve the development efforts, at least minimal technology independence is desirable at the legacy integration and at the services design and compositions.

To meet these needs, we propose the construction of conception design platform, which main objective is to provide an effective and consistent approach to manage metadata and a service oriented architecture for the cooperation among disparate administrative units in a collaborative environment.

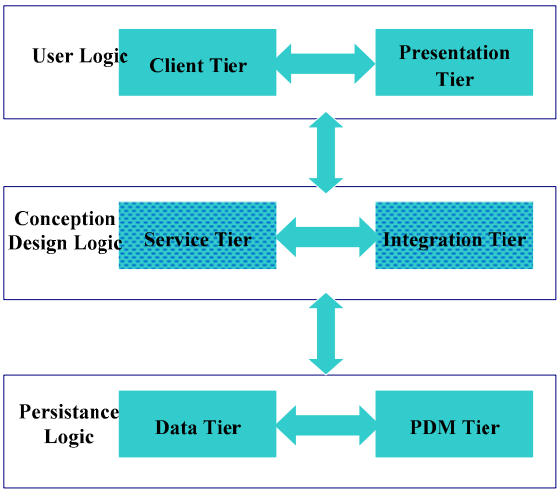

Figure 1. Platform n-tier architecture

Figure 1 shows n N-tier architecture for the platform. In the conception design logic, the service tier is responsible for the service management and the integration tier provides an integrated approach to the existing legacy systems.

The remainder of this paper is organized as follows. In section 2, we present key concepts related to MDA and Web Services. In section 3, we present an outline of MDA-Compatible conception design development process. Section 4 and section 5 deploy a case study of the conception design environment and MDA of web application.

2. Background information

2.1 Models and Metamodels

The OMG′ Meta Object Facility (MOF) defines a model as an instance of a meta-model. A metamodel can describe properties of a particular platform. In this case, the models that are instances of such a metamodel are

Proceedings of the First International Multi-Symposiums on Computer and Computational Sciences (IMSCCS'06)
0-7695-2581-4/06 $20.00 © 2006 IEEE

said to be platform-specific, while models that describe a system at a level of abstraction, one that is sufficient to allow use of their entire contents for implementing the system on different platforms, are referred to as platform-independent. Figure.2 [1] illustrates the relationships.

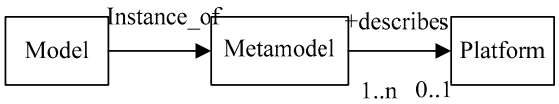

Figure 2. Model, Metamodel and Platforms

In our system, we have adopted Model-Driven Architecture (MDA) [OMG2004] as the modeling framework. The three primary goals of MDA are portability, interoperability and reusability [2] through a consistent separation of some basic concerns in the descriptions of software. Regardless of what kind of system is involved, MDA recommends a separation among(ⅰ) platform independent descriptions of components and their interactions(Platform Independent Model-PIM); (ⅱ) technical artifacts, such as specific design models(Platform Specific Model-PSM) and source codes; and(ⅲ) conceptual design models that are being implemented by the platform independent descriptions of how components and services interact.

2.2 Mapping between Models

A metamodel mapping can provide rule and /or algorithms for transformation of a PIM to PSM, of a particular platform, or a PSM to PIM.
In general, Every Model may have semantic relationship with other models. When you describe a particular system at different levels abstraction, you will find there exists mapping between different but related models performed automatically.

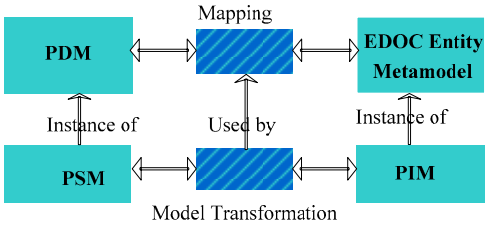

Figure 3. MDA mapping and transformation

These relevant concept in our work are illustrated in Figure 3

2.3 Legacy integration

In this section, we describle the way PIM, PSM, mappings and transformations are used in the collaborative environment, and it can provide legacy integration [2] and service collaboration support. As illustrated in Figure 4 and [2].

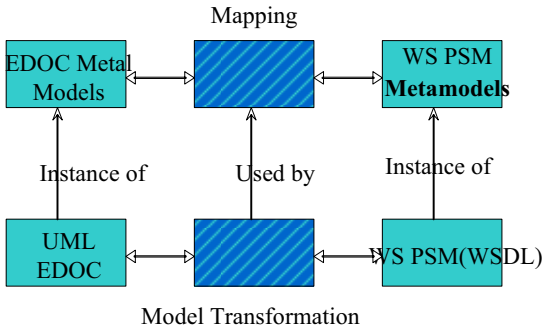

Figure 4. Legacy integration

3. Outline of MDA-Compatible conception design development process

The following ten modified process steps [1], taken together, offer a simple a simple robust way to incorporate MDA into a software development project.

1. Establish the domains of interest, and the requirements within those domains.
2. Identify a set of target platforms.
3. Identify the metamodels that we want to use for describing models for conception design platform, and also the modeling language/profile in which we will express our models.
4. To Find or select the proper abstracting metamodels.
5. Define a model as an instance of a metamodel.
6. Define the mapping techniques that we will use with our metamodels so that there are full paths form the most abstract metamodels to the metamodels of all of our target platforms.

6.1 Define mapping Language Requirements.
6.2 Define the functional requirements
6.3 Define the usability requirements.

7. Define the annotation models that these mapping techniques require.
8. Implement the mapping techniques either by using tool support or by describing the steps necessary to manually carry out the technique.
9. Modelling: use ArgoUML [6] to build an executable specification, or Platform-Independent Model(PIM) for each of the new domains/steps to be developed.
10. Conduct iterations of the project. Each iteration will add detail to one or more models describing the system at one or more levels of abstraction. We will map the additional details all the way down to the target platforms

The transformation language in these steps must be able to accord with the following rules.

1. Match sets of source model elements.
2. Use types to match elements, types include instances of sub-types and precise type such as exclude instances of sub-types.
3. Filter the set of matched tuples based on associations, attribute values, and other criteria.
4. Establish associations between source and target model elements.
5. Others criteria [10].

In general, readability and expressiveness is desirable, it concerns that:

1. Use a single rule to define multiple target elements,
2. Rules group naturally,
3. Can make transformation immediately,
4. Can deal with optional attributes.

4.Case study-The conception design system

In Fig.5. We show a simplified example independent model (PIM) and its transformation into three different platform-specific models (PSM), in this paper, we select the three platform specific model(PSM).

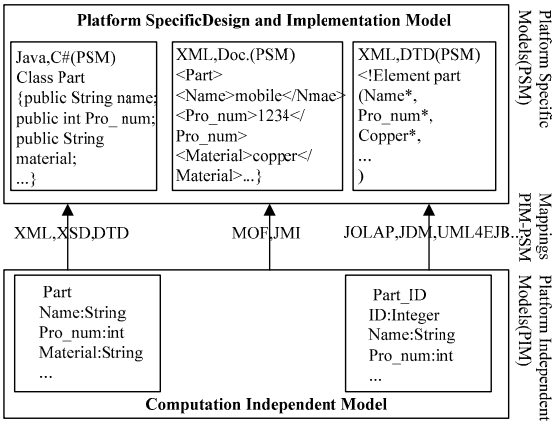

Figure 5. PIM, PSM and its transformation

Figure.5 shows a simple PIM representing a Part and Part_ID. At this level of abstraction, the model describles important characteristics of the domain in terms of classes and their attributes, but without making any platform-specific choices about which technology will be used to represent them.

In our work, we adopt the tool of TUPI [3] (Transformation from PIM to IDL)- that does an automatic transformation from a PIM to the corresponding specification in CORBA IDL [4]. TUPI receives as input a XMI (XML Metadata Interchange Format) [5] file that contains the meta-model description of the PIM model. ArgoUML [6] is used to produce the PIM Model. The PIM Model follows the syntax proposed by the UML profile for EDOC [7]. The PIM-PSM conversion rules are described in XSLT (eXtensible StyleSHeet Language Transformations) [8] and they produce a specific model to the CORBA platform represented in IDL (Interface Definition Language).

5. Model driven architecture of web application

The proposed platform supports the design of collaborative services using the MDA concepts and the Web Services composition techniques. The services are first developed platform independent, by means of the ArgoUML [6] profile and thereafter transformed to platform specific model like Web Service(WS-PSM)

In order to define a Web application System, [9] proposes a Web application view model that is made up 8 views, grouped into three: requirements, functional and architectural viewpoints. This viewpoint includes a logical architectural view and a physical architecture view. The logical architectural view gathers the set of logical components (subsystems, modules and/or software components) and relations among them. While the physical architecture view describles the physical components that integrate the lower level specification of the application (clients, servers, networks, etc.). [9] defines a process that can shift one view to the other.

6. References

[1] S. J. Mellor, K. Scott, A. Uhl, and D. Weise, “Model-driven architecture,” *OOIS 2002 Workshops*, LNCS 2426, pp. 290-297, 2002.

[2] A. Maria, C. M. Figueiredo, M. deJesus Mendes, A. Kamada, “Applying MDA concepts in an e-Government platform,” *EGOV 2004*, LNCS 3183, pp. 260-265, 2004.

[3] T. Nasciminto, T. Batista, and N. Cacho, “TUPI: transformation from to IDL,” *CoopIS/DOA/ODBASE 2003*, LNCS 2888, pp. 1439-1453, 2003.

[4] Z. Tari and O. Bukhres, *Fundamentals of distributed object systems- The CORBA perspective*, John Wiley & Sons.(2001)

[5] OMG: XML Model Interchange (XMI) OMG Document ad/98-10-01, (1998).

[6] A. Ramirez, P. Vanpeperstraete, A. Rueckert, K. Odutola, J. Bennett, and L. Tolke, “ArgoUML, - a tutorial and reference description,” (2000). Available at argouml.tigris.org/

[7] OMG: UML profile for enterprise distributed object computing specification(EDOC), OMG Document ad/01-08-18, (2001)

[8] W3C: XSL Translations Specification W3C, (1999). Available at www.w3.org /TR/xslt.

[9] S. M. Beigbeder and C. C. Castro, “An MDA approach for the development of Web applications,” *ICWE 2004*, LNCS 3140, pp, 300-305, 2004.

[10] A. Gerber, M. Lawley, K. Raymond, J. Steel, and A. Wood, “Transformation: The Missing Link of MDA,” *ICGT 2002,* LNCS 2505, pp. 90-105, 2002.