Application MDA in a Collaborative Modeling Environment

Wuzheng Tan[1], Lizhuang Ma[1], Zhiliang Xu[1], Junfa Mao[2]
[1]Department of Engineering and Computer Science, Shanghai Jiao Tong University, 200240, China
[2]Department of Electronic Engineering, Shanghai Jiao Tong University, 200240, China
tanwuzheng@yahoo.com.cn, ma-lz@cs.sjtu.edu.cn, xuzhiliang@sjtu.edu.cn, jfmao@sjtu.edu.cn

Abstract. This paper proposes a modeling 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. Based on [1] and [2], this article offers a modified software development process that would leverage MDA, and studies a web application case of conceptual innovation modeling system

Keywords: MDA, Collaborative Modeling, Collaborative Modeling Platform, Web Application

1 Introduction

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.

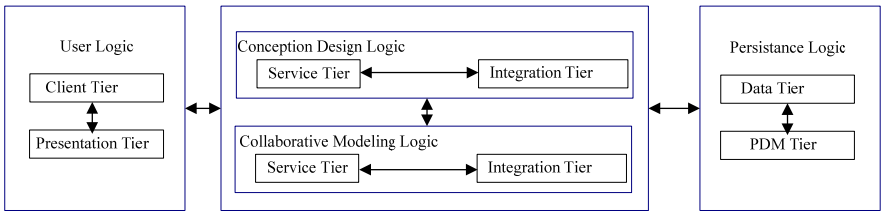

Fig.1. Platform n-tier architecture

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.

In [2], it introduced some related works about models and meta models, mapping between models and legacy integration.

In [14], it propose a sugarcane harvester modeling framework based on the commonly available communication platform and illustrate it with the implemented software that can be used as a core part for different real life applications. It is possible to perform their collaborative interactive modifications with concurrent synchronous visualization at each client computer with any required level of detail. The platform provides a product modeling and assembling environment and changes paradigms between I_DEAS, Smarteam(PDM software) and Conceptual Innovation Design System (CIDS). The platform is illustrated for sugarcane harvester modeling and assembling analysis as an example, and provides a collaborative intelligent environment for the model of products, aiming at integrating people, process and data in the model development.

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.

3 Outline of MDA-Compatible Collaborative Modeling Development Process

The following twelve modified process steps based on [1][2], taken together, offer a simple robust way to incorporate MDA into a software development project for CIDS.

1. Establish the domains of interest, and the requirements within those domains.
2. Establish the connection between requirements and target platform.
3. Identify a set of target platforms.
4. 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.
5. To Find or select the proper abstracting metamodels.
6. Establish the connection between abstracting metamodels and their instances.

7. Define a model as an instance of a metamodel.
8. 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.
   Define mapping Language Requirements.
   Define the functional requirements
   Define the usability requirements.
   Define the transfering requrements.
   Define the collaborating modeling requirements.
9. Define the annotation models that these mapping techniques require.
10. Implement the mapping techniques either by using tool support or by describing the steps necessary to manually carry out the technique.
11. Modelling: use ArgoUML [6] to build an executable specification, or Platform-Independent Model(PIM) for each of the new domains/steps to be developed.
12. 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 rules[2].

4 Case Study-The Collaborative Modeling System

We study 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).
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).

The project had two main objectives.

It provides a stable application interface between PDM framework layer, a configuration model layer, a functional model layer, a fuzzy reasoning layer, an integration layer for I-DEAS and CIDS, Service-Oriented Conception Design (SOCD) Model layer, design evaluation layer, a computer methods layer and a personal Web Graphical User Interface (GUI).

It provides organizational use of access control and collaborative services, and allows users to access to the CIDS.

4.1 Requirements and Domains

The requirements for access control and collaboration could be traced into the UML models. We build and test UML models. It was configured using the requirements schema in Fig.2 below, and represented as a UML class diagram.

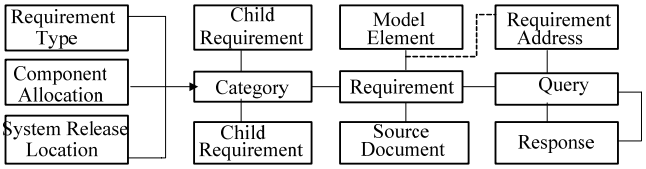

Fig.2. Requirement Schema

These requirements can be categorized, arranged into parent-child hierarchies, traced back to their source, or forward into the models, as illustrated in Fig.2.

Each requirement also went through a set of life cycle phase, as shown in the state chart diagram in Fig.3 below, that is represented using the Moore formalism as described in [12] and [13].

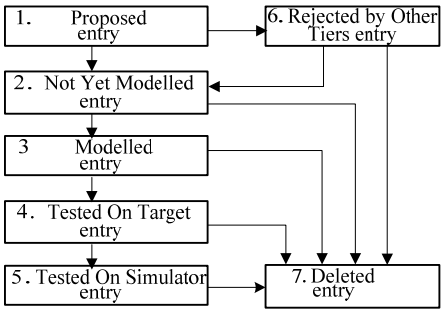

Fig.3. Requirement State Machine

4.2 The Components (Domains)

To formalize the conceptual content of the requirements and the behavior they define; models were designed to lend themselves to ongoing modification. We can combine generic modeling and isolate volatile issues in separate domains, which nonetheless offer a stable interface to the rest of the system layers.

We define several inter-related domains to capture and control the complexity of the overall system.

This domain model diagram in Fig.4 shows the various components (domains) comprising the proposed solution.

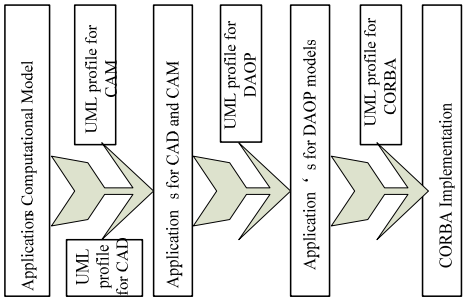

Fig.4. The stack with the different models and the MDA transformations between them

In order to address these models, we got the conclusion that , the first step was to identify the different conceptual tiers involved in the development of an application using DAOP platform [10]. The following list of models was produced.

The Computational Model focuses on the functional decomposition of the system into objects which interact at interface, exchanging messages and signals that invoke operations and deliver service responses-but without detailing the system precise architecture, or any of its implementation details. This model basically corresponds to an ODP computational viewpoint model of the system, or to Zachman's Framework for Enterprise Architecture Row3[11]. Entities of this model are objects (implementing interfaces) and operations. To which we have added some constraints for expressing extra-functional requirements (such as security or persistence, for instance).

The component and aspect model (CAM) and the component and aspect design model (CAD) define the basic entities and the structure of the system form an architectural point of view. In our case, components and aspects are the basic building blocks, following our goal of integrating CBSD and AOCD approaches.

The DAOP platform implements the concepts of the CAM model in a particular way. This level is still technology-independent, since it just deals about the way components and aspects are weaved, and how the abstract entities of the CAM model can be represented form the computational and engineering viewpoints-but still independently form the middleware platform (CORBA, EJB,.NET) or programming language used implement them.

The middleware platform provides a description of the system form a technology viewpoint. In this model we decide whether we want to implement the DAOP platform using Java/RMI, CORBA, EJB, or .NET, using their corresponding services and mechanisms.

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.

Acknowledgments. This work is supported by National Science Fund for Creative Research Groups （60521002）and national natural science foundation of China (Grand No. 60573147).

References

1. Stephen J. Mellor, Kendall Scott, Axel Uhl and Dirk Weise. Model-Driven Architecture. OOIS 2002 Workshops. LNCS 2426. pp. 290-297. 2002
2. Wuzheng Tan, Lizhuang Ma, Jun Li, and Zuxiu Xiao. Application MDA in a Conception Design Environment. International Multi-Symposiums on Computer and Computational Sciences(IMSCCS|06).
3. Teresa Nasciminto, Thais Batista, and Nelio Cacho. TUPI: Transformation from to IDL. CoopIS/DOA/ODBASE 2003, LNCS 2888, pp. 1439-1453, 2003.
4. Tari, Z., Bukhres, O.: 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. Ramirez, A.,Vanpeperstraete, P., Rueckert, A., Odutola, K., Bennett, J., Tolke,L., 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. Santiago Melia Beigbeder and Cristina Cachero Castro. An MDA Approach for the Development of Web Applications. ICWE 2004, LNCS 3140, pp, 300-305, 2004.
10. Lidia Fuentes, Monica Pinto, and Antonio Vallecillo. How Can Help Designing Component-and Aspect-based Applications.EDOC03,2003.
11. J. A. Zachman. The Zachman Framework: A primer for enterprise Engineering and Manufacturing. Zachman International, 1997. http://www.zifa.com.
12. S. Shlaer, S. J. Mellor. Object Lifecycles: Modelling the World in States. Yourdon Press Computing Series. (April 1991).
13. C. Raistrick et al. Model Driven Architecture with Executable UML. Cambridge University Press. (March 2004).
14. Wuzheng Tan, Lizhuang Ma, Jun Li, Junfa Mao. A Platform for Collaborative Modeling. Journal of Computer Science and Technology. Submitted.