INTEGRATION OF ARCHITECTURAL DESIGN AND IMPLEMENTATION DECISIONS INTO THE MDA FRAMEWROK

Alti Adel[1] and Smeda Adel[2]
[1]*Department of Computer Science, Engineering Faculty, University of Setif, P.O. Box 19000, Setif, Algeria*

[2]*Department of Data-Analysis, Faculty of Accounting, University of Al-Jabel Al-Gharbi*
P.O. Box 64200, Gharian, Libya

Keywords: Software Architecture, ADL, MDA, decision-centric design, decision-centric implementation, DCMDD.

Abstract: Model Driven Development (MDD) is typically based on models which heavily lead the quality production of application's architecture. This is because architectural decisions are often implicitly embedded in software engineering, therefore lacks first-class consideration. Architecture has been established as a key to develop software systems that meet quality expectations of their stakeholders. The explicit definition of architectural decisions, aims to well control the quality on the software development process. In this paper, we propose to extend the MDA framework by integrating decision aspects. We propose also an approach to use architectural decisions as a meta-model for the MDD process. Integration of architectural decisions allows architectural design to be defined explicitly and guides architects in creating systems with desirable qualities; and for MDA it extends the approach by integrating true decisional concerns into MDD process.

1 INTRODUCTION

MDA (Model Driven Architecture) is a recent initiative of the OMG that supports the definition of models as first-class elements for the design and implementation of systems. According to the MDA approach, the most important activity now is modelling the different aspects of the system and then defining the transformations from one model to another in a way that allows them to be automated (Fernández and Vallecillo-Moreno, 2004).

MDA is increasingly becoming popular as a widely accepted standard for software development process. These have been elaborated in order to improve the comprehensibility of complex systems, favourite their portability, their productivity as well as their maintainability. MDA doesn't really take into account the architecture design and implementation decisions. Architectural decisions are defined implicitly in software development process and lack a first class consideration and therefore the quality control in software development process is limited. The inclusion of the architecture design and implementation decisions into the MDA framework yield two main advantages. On the one hand, it allows architectural decisions to benefit from the advantages of the MDA approach and on the other hand, it introduces a change in the development process, turning the MDD approach into a decision-centric MDD (*DCMDD*). This aims for controlling quality on software development process.

In this article, we try to integrate architectural decisions into a MDA framework. We also discuss the usefulness and the importance for architectural design and implementation decisions to use an appropriate Architecture Description Language (ADL) that meet quality expectations and an appropriate MDA platform that meet software architecture integration. The development process is supported by the architecture decision. For this reason this approach is called *Decision-Centric Model-Driven Development* (DCMDD). The proposal approach presented here is the result of our previous work on using profile transformations for integrating software architecture concepts into MDA platform (Alti, Khammaci, Smeda and Bennouar, 2007), we identified the need to specifically consider architectural decisions concerns. Different ways to integrate architectural decisions into our model-driven software architecture have consequently been tested; in this paper we present the outcome of this work. We provide also in this paper a description of

Adel A. and Adel S. (2008).
INTEGRATION OF ARCHITECTURAL DESIGN AND IMPLEMENTATION DECISIONS INTO THE MDA FRAMEWROK.
In *Proceedings of the Third International Conference on Software and Data Technologies - PL/DPS/KE*, pages 366-371
DOI: 10.5220/0001881403660371

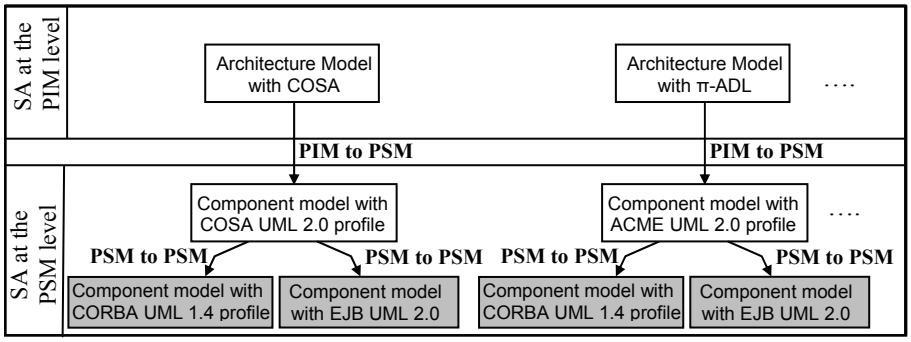

Figure 1: Model-driven software architecture construction and integration (structural view).

a suitable case study, which applies the *DCMDD* approach to the model-driven software architecture design and integration and illustrates its use with the help of a client/server system.

The remainder of this article is organized as follows. Section 2 provides a brief overview of the model-driven software architecture construction and integration. Section 3 presents the related work. Section 4 proposes a model-driven for software architecture construction and integration that includes the architecture design and implementation decisions; this new model architecture becomes the Model Driven Development (MDD) approach in a Decision-centric Model Driven Development (ACMDD). As a case study, we use the previous DCMDD approach to in the development of the client/server system. Finally, section 5 concludes this article and presents some future works.

2 INTEGRATION OF SOFTWARE ARCHITECTURE CONCEPTS INTO MDA: AN OVERVIEW

In our previous work (Alti, Khammaci, Smeda and Bennouar, 2007), we integrated software architecture concepts (e.g. connectors, configurations...etc) into MDA platform, concepts of ADLs such as COSA (Oussalah, Smeda and Khammaci, 2004), ACME (Garlan, 1997) and π-ADL (Ouquendo, 2004) are considered as PIM and explored in MDA platform as PSM. We proposed a strategy for direct transformation using a UML profile. It represents both software architecture model (PIM) and MDA platform model (PSM) in UML meta-model then elaborates transformation rules between resulted UML meta-models. The goal is to automate the process of deriving implementation platform from software concepts (see Figure 1).

The main problem found when applying this strategy was that the architecture decision was not being considered. The software architecture was designed with an arbitrary ADL and also was integrated into a given MDA platform.

At PIM level the description of software systems using an arbitrary ADL; that do not meet quality design such as resuabilty and extensibility. This is because various ADLs are based on different architectural concepts and treated them differently; each ADL evaluate systems behaviors with its appropriate method (B method for COSA (Alti, Khammaci and Smeda, 2007), CSP method for ACME (Garlan, 1997) and π calculus for π-ADL (Oquendo, 2006). We need to find a software architecture model to describe software systems and we need to find a well MDA platform to integrate architecture concepts.

When do we use a given ADL? What happens with the architectural design and implementation decisions? How does it affect the MDA platform? Is it easy or difficult to describe a given software system with a given ADL? Early design decisions can greatly affect the amount of architecture changes are required. To be able to answer these questions, we included architectural decisions in the model-driven software architecture as configuration inputs.

3 RELATED WORK

Alesseandro and al (Alessendro, Thais, Awais and Claudio, 2006) handled architectural decisions as separate architectural aspects, not just simply symmetric view. They used Aspect-Oriented Software Development (AOSD) techniques for capturing widely-scoped architectural decisions and for compositing them. Their approach allows a well

separation between different crosscutting decisions, but disagrees in more representation of architectural decisions at the ADLs level and in the driving of the MDD process. Jansen et al (Jansen, Jan ven der, Avgeriou and Hammer, 2007) treated software architecture as a composition of a set of architectural decisions. They supported the definition of design decisions as first-class elements that guide the architecture construction process. They proposed a meta-model for maintaining relationships between design decisions and software architectures with support tool. They studied the solutions and their consequences of Athena system related to some design decisions into Archium meta-model. They ended up with one solution for each design decision. They concluded that even though architectural decisions are associated with the description of software architecture, they still have considerable drawbacks. Our approach shares the elements of the architectural design decisions model, and differs since software architecture language is our potential solutions. In (Capila, Nav and Dueňas, 2007) Capila et al. turned traditional approach of software architecture as a set of components and connectors as the result of a set of design decisions. They proposed a meta-model and a web-based tool able of recording, maintaining and managing the decisions tacking during the architecture construction process. They define an integrated view for evolving design decisions. Then describe decisions (architectural products are defined using PDF documents); this makes it difficult to promote decisions reuse. The Architecture-Centric Concern Analysis (ACCA) method (Wang, Sherdil and Madhavju, 2007) employs a meta-model for capturing architectural design decisions which are linked to software requirements and architectural concerns. The authors identify the causes of architectural concerns and they assess the wrong or incorrect decisions that are taken. Zimmermann et al (Zimmerman, Koehler, Leymann, 2006), positioned architectural decision models as the control center of model-driven SOA construction and followed a transformation approach on the top of decision models. This approach is decision-centric because it defines the model-driven devolvement process on top of a decision meta-model which provide powerful tool support.

Our approach is similar but different. In our proposal, decision is on top of the systems which drives the rest of the development process, and play the central role that defines the structure of our concrete model architecture, and decides which architecture models are considered and which are not. Since decision is in charge of controlling the development process. That is the reason why our approach is a DCMDD process. In our approach, decisions can be seen as a meta-model for the *Decision-Centric Model Driven Developement* (*DCMDD*) itself. In fact, it shows which of software architecture languages instantiated for the representing software systems, which technologies are instantiated during the decision-centric MDD process, and in which of them are not. This quality is missing in the other models.

4 INTEGRATING OF ARCHITECTURAL DECISIONS INTO THE MDA FRAMEWORK

We think that the best way to build an architecture is to quantify the metrics of the decision binding software architecture design with a given ADL. Based on the ADL quality characteristics (extensibility, portability, reusability, etc.), design decisions give us a best architecture model that better represent our software architecture needs. In order to build a system that fulfil the expectations, it is essential to systematically take decisions and their rationales into account in architecture design step and not as an afterthought during transformation. From this point of view, we can not describe software systems with an arbitrary ADL but we need to justify our choice. Of course, architecture is an abstraction of the structure of a system, provided in terms of ADL, but what benefits gained by the software systems? Such ADL answer the architecture design needs or not? Therefore, we suggest placing architectural decisions as a control-centre for designing software architecture systems.

On the other hand, we think that the best way to integrate software architecture into the MDA platform is to quantify the decision binding software architecture integration with a given MDA platform. Based on the MDA platform quality metrics (facile integration of software architecture concepts and supportability), implementation decisions gives us a best implementation model that better represents our software implementation needs. From this point of view, we can not integrate software concepts with an arbitrary MDA platform but we must justify our choice.

4.1 Decision Models as a New Dimension in the Model-driven for Software Architecture

After different attempts to include architectural decisions in the model-driven software architecture

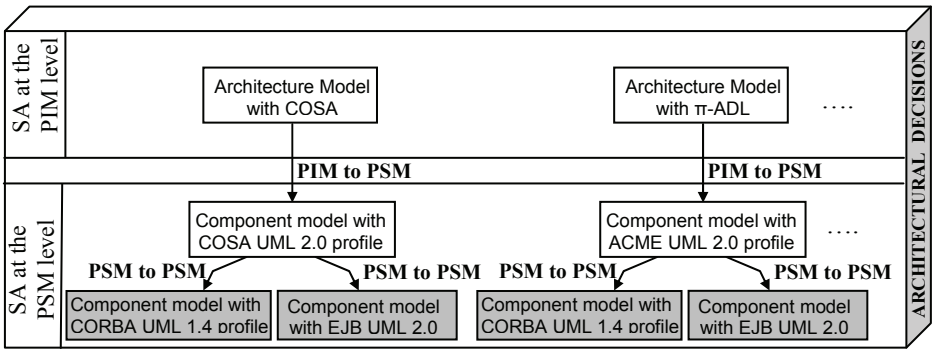

Figure 2: Model-driven software architecture, including the architecture decision model (structural view).

integration, we decided finally that the architecture decision has to be a different view, but in a new dimension. This new dimension will be orthogonal to the level of abstraction and to the aspects (see Fig.2). This is because architecture design decisions are related to the other system aspects (structure and behavior). Moreover, architectural design decisions will determine which software architecture models of each aspect we'll need for better describing software architecture from a conceptual point of view, allowing an architectural design decisions to be free from the technological constraints. Next, after some design decisions, all the required components of software systems can be implemented by specific technologies depending on specific needs, available technologies, etc. at PSM level.

We propose to consider software quality attributes (Jansen et al, 2007) of the link binding software system with a given ADL as a primary input before the design phase. Next, we propose to consider software quality attributes of the link binding software architecture integration with a given MDA platform before the implementing phase. In fact, it is recommendable in the MDA framework, to select an adequate software architecture model and to drive the transformation between software architecture models and the implementation models.

4.2 Improving the Development Process using DCMDD

To illustrate how our approach can be used, we apply it to the well known client/server system. Section 4.2.1 introduces the design decisions at the PIM level and Section 4.2.2 introduces the implementation decisions at the PSM level.

4.2.1 Design Decisions at PIM Level

4.2.1.1 Problem

It is indispensable to reason about and to explore, in architectural level (PIM), the completeness and quality attributes of the final product resulting from the architecture. In order to determine whether the client/server system fulfills non-functional properties as security, a powerful ADL is needed.

Motivations. Modeling and formally specifying architectural components of the client/server system using appropriate ADL and a specific technique is considered as an important factor that helps in the construction phase especially in terms of flexibility. Other properties like security and reusability are needed.

Cause. Divers ADLs (Medvidovic and Taylor, 2000) are available for describing software architecture systems, which makes it difficult to select an appropriate ADL for a given system.

Context. The client/server system.

4.2.1.2 Potential Architectural Descriptions

- **π-ADL**

π-ADL (Oquendo, 2006) is an ADL based on π-calculus for modeling behavioral aspects. π-ADL supports formal architecture-centric model-driven development and semantics checking. It has been applied in several realistic case studies.

Design Rules. For each connection of the client/server system provided by a port of a component is attached to a connection provided by a port of a connector by *unification* or *value passing*.

Design Constraints. We must ensure that the given model complies with the semantic constraints defined by the π-ADL.

Consequences. The client/server system becomes dependent on the π-ADL.

Pros. (+) flexibility (+) Achieving a comfortable system (+) better manageability and administration (+) multi-layered architecture specifications.

Cons. (-) doses not ensure security and extensibility.

- **COSA**

COSA (Oussalah, Smeda and Khammaci, 2004) is hybrid model, based on both object and component modeling to describe software systems. The basic principal of this model is to define architectural concepts extended with object-oriented concepts and mechanisms to specify software architectures. For behavioral view, a B formal method given in (Alti, Khammaci and Smeda, 2007) need to be used.

Design Rules. For each connection of the client/server system provided by a port of a component is attached to a connection provided by a port of a connector by *unification* or *value passing*.

Design Constraints. We must ensure that the given model complies to the semantic constraints defined by the COSA approach

Consequences. The client/server system becomes dependent on the COSA language.

Pros. (+) more simplicity and genericity (+) facile construction (+) satisfy the more specific system security by using component redefinition (+) assist the configuration management (+) more reusability and extensibility (+) B formal method applied in several industrial cases.

Cons. (-) has not been used yet for industrial cases.

4.2.1.3 Decision

The decision is made to use COSA software architecture for modeling the client/server system. It allows more simplicity, extensibility, and genericity. It satisfies reusability with object-oriented mechanisms and security with B formal method.

4.2.2 Implementation Decisions at PSM Level

4.2.2.1 Problem

Actually, there are several middleware platforms (CORBA, J2EE, .Net, etc.) that focus on developing component-based systems. Which of them is appropriate platform to provide increased performance and can easily map COSA concepts?

Motivations. It is indispensable to explore, in implementation level (PSM), the integration of COSA software architecture concepts (components, connectors, configurations, etc.) in order to explicit the communications and the coordination of distributed components by mapping COSA connectors.

Cause. Various MDA platforms available for system implementation.

Context. Client/server system with COSA-UML.

4.2.2.2 Potential MDA Platforms

- **CORBA Platform**

CORBA is an international standard middleware platform that provides simplicity for development by report to other platforms.

Mapping Rules. We must define how the transformation rules map COSA concepts into CORBA.

Implementation Constraints. The CORBA model must be evaluated by the CORBA UML profile.

Consequences. The client/server system becomes dependent on the CORBA.

Pros. (+) solving the problem of interaction in CORBA components by mapping COSA connectors (+) simplicity of development (+) integrating COSA concepts can be achieved easily.

Cons. (-) has not been used yet for industrial cases.

- **J2EE Platform**

EJB is the core component model of the J2EE platform developed by sun Microsystems for full support of distributed operations/services.

Mapping Rules. We must define how the transformation rules map COSA concepts into EJB.

Implementation Constraints. The EJB model must be evaluated by the EJB UML 2.0 profile.

Consequences. The client/server system becomes dependent on the EJB.

Pros. (+) achieve a higher level of abstraction (+) solving the problem of interactions among EJB components by mapping COSA connectors into EJB

Cons. (-) has not been used yet for industrial cases.

4.2.2.3 Decision

The decision is made to use CORBA, which hides the complexity of distributed processing systems and

therefore help in integrating COSA concepts. The COSA-CORBA transformation (Alti, Khammaci and Smeda, Bennouar, 2007) is applied to the COSA model for elaborating its correspondent CORBA model for the Client-Server system.

4.3 Implementing the DCMDD Approach

We have developed a plug-in in Eclipse to implement the DCMDD approach. The DCMDD tool provides a more pragmatic approach to the usage of architectural decisions: it links the architecting process with the system implementation through transformation.

The Plug-In is developed in four steps: 1) the metamodels of Pi-ADL (COSA, CORBA and EJB) with all tagged values and OCL constraints is defined by the UML 2.0 profile 2) the architecture decisions and its consequences are expressed using Java 3) The architectural description language (MDA platform) is selected and the model is elaborated through its transformation 4) the elaborated model is evaluated by its profile.

Figure 3 shows the applied CORBA model of Client-Server system after deciding to apply COSA to CORBA transformation.

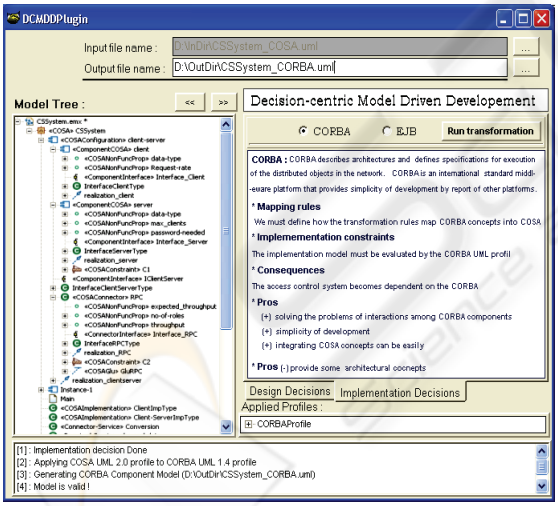

Figure 3: The DCMDD Eclipse Plugin.

5 CONCLUSIONS

In this paper we have showed how to include the architecture design and implementation decisions as a new aspect in the MDA architecture. We have also illustrated the usefulness and importance of architectural decisions in the context of MDA platform. Architectural decisions will play an important role as they promise to explicit and guidance to an architect in creating systems with desirable qualities. However, our approach does not include the description of architectural styles. In our future works we will apply DCMDD in the other SA-based and on top of a Grid architectured.

REFERENCES

Alti, A., Khammaci, T., and Smeda, A., 2007. *Using B Forma method to Define Software Architecture Behavioral Concepts,* IRECOS Review, Vol. 2, No 1, pp. 510-519, ISSN: 1828-6003.

Alti, A., Khammaci, T., Smeda, A., Bennouar, D., 2007. Integrating Software Architecture Concepts into the MDA platform, In *ICSOFT'2007, 2nd Int. Conf. on Software and Technologies*.

Alessandro G., Thais, A.B., Awais, R.S., 2006. Driving and managing architectural decisions with aspects, *ACM SIGSOFT Software Engineering Notes*, Vol.31 No 5, pp. 30-37.

Capila, R., Nava, F., and Dueñas, J.C., 2007. Modeling and Documenting the Evolution of Architectural Decisions, *6th IEEE/IFIP Working Conference on Software Architecture (WICSA'07)*.

Fuentes-Fernández, L., Vallecillo-Moreno, A., 2004. An Introduction to UML Profiles. *The European Journal for the Informatics Professional*, 7(2), pp. 6-13.

Garlan, D., 1997. ACME: An Architecture Description Interchange Language, In *CASCON'97*.

Jansen, A., Jan ven der, J., Avgeriou, P., and Hammer, D.K, 2007. Tool Support for Architectural Decisions", *Proceedings of the 2nd Workshop on Sharing and Reusing Architectural Knowledge*, ICSE Workshop, IEEE DL.

Medvidovic, N., Taylor, R. N., 2000. A classification and comparison framework for software architecture description languages. *IEEE Transactions on Software Engineering*, 26(1), pp. 2–57.

Oquendo, F., 2004. Formally Describing Software Architecture with π-ADL, *World Scientific and Engineering Transactions on Systems*, Vol. 3. N°.8, pp. 673-679.

Oussalah, M., Smeda, A., Khammaci, T., 2004. An explicit definition of connectors for component based software architecture. In *ECBS'2004, the 11th IEEE Conference Engineering of Computer Based Systems*, Czech Republic.

Zimmerman, O., Koehler, J., and Leymann, F., 2006. The Role of Architectural Decisions in Model-Driven SOA Construction, *In Best Practice and Methodologies in SOA, OOPSLA'06*.

Wang, A., Sherdil, K., Madhavju, N.H., 2005. An Architecture-centric concern analysis method, In *WICSA'05, the 5th IEEE/IFIP Working Conference on Software Architecture*.