Journal of Software Engineering and Applications, **2011, 1, 18-22**
doi:10.4236/jsea.2011.41003 Published Online January 2011 (http://www.scirp.org/journal/jsea)

Scientific Research

Incorporating AOSD to Enhance Model Driven Architecture

Yashwant Singh[1], Manu Sood[2], Tarun Gupta[1], Atish Thakur[1]

[1]Department of Computer Science Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat, India; [2]Department of Computer Science, Himachal Pradesh University, Shimla, India
Email: {yashu_want, soodm_67}@yahoo.com, tarungupta_ce@yahoo.co.in, atish092208cse@gmail.com

Received December 14th, 2010; revised December 21st, 2010; accepted January 12th, 2011.

ABSTRACT

Various software development approaches in the present scenario are best suited for a specific application. The software development strategies include both merits and demerits when talked in terms of generalization. The Model Driven Architecture (*MDA*) *describes software development based on models on various levels of abstraction. The model driven software development process consists of sequence of model transformations between various models. This paper presents incorporation of the merits of Aspect Oriented Software Development* (*AOSD*) *like modularization, reusability and reduced complexity into Model Driven Architecture* (*MDA*) *software development strategy. The proposed Model Driven Software Development approach which combines the merits of MDA and AOSD meets the ever changing and challenging demand of the enterprise software development.*

Keywords*: *Model Driven Architecture, Aspect Oriented Software Development, Computational Independent Model, Platform Independent Model, Platform Specific Model

1. Introduction

Various day-to-day advancements are made in the field of Software development approaches. It results in eruption of various software development approaches to specific enterprise systems. The use of new technologies and modern trends are not being considered sufficiently due to the software development paradigm shift. One such solution of the problem could be a regular study program review and integration of new modules into the current software development processes.

The challenges faced by enterprise software development are the effective design of software system to support ever-changing business capabilities in a timely manner and bridging the vision of an application to its realization. One such solution to the problem is provided by Model Driven Architecture approach of software development under Model Driven Software Development (MDSD).

MDA approach of software development under MDSD helps producing three models [1]: 1) Computation Independent Model (CIM), which specifies the models for domain and requirements of the system, 2) Platform Independent Model (PIM), which specifies the models for abstract concepts of the system and excludes platform specific details and 3) Platform Specific Model (PSM), which specifies and models how the functionality in a PIM is brought to reality on a specific computing platform, with high abstraction.

MDA also supports the transformations [2] from CIM to PIM and from PIM to PSM. Transformation of PIM to PSM includes specifying the rules to transform a PIM into [3]. 1) An application layer model to be implemented using object oriented languages, 2) A database layer model to be implemented using RDBMS, and 3) An interaction between application layer and database layer is also implemented by object oriented languages like JAVA; all as a part of PSM.

This paper is an attempt to incorporate the merits of (AOSD) [4] into Model Driven Architecture approach of software development. The Aspect-Oriented Software Development decomposes a software system into modules in such a way that modules responsible for a common concern are tightly coupled and modules responsible for different concerns are loosely coupled. Aspect-Oriented Software Development is a post object-oriented technology that helps achieve better separation of concern by providing mechanisms to localize concerns like security, synchronization, and logging in software development process. Furthermore new MDA models CIM, PIM

and PSM with added advantages of Aspect Oriented Software development has been proposed.

The remainder of this paper is organized as follows: Section 2 gives the illustration of Model Driven Architecture with its various models. Section 3 presents Aspect Oriented Software Development (AOSD) and its importance in development of enterprise system. The proposed MDA approach of software development with added advantages of AOSD illustrated in Section 4. The comparison between traditional and proposed model has been shown in Section 5 and Section 6 presents conclusion.

2. Motivation the Model Driven Architecture

MDA is a flagship initiative of the Object Management Group (OMG) [2,5] presenting a new vision on how enterprise systems should be developed and managed. Launched in 2001, MDA is now making a great impact in the areas related to the enterprise software development. It is an approach that addresses the increasing complexity in enterprise-system design by combining various technologies for an effective software development. The MDA approach is a model-centric paradigm which automates the generation of system implementation artifacts from the model directly [6]. The approach works on the following layers of model:

- Computation Independent Model (CIM)
- Platform Independent Model (PIM)
- Platform Specific Model (PSM)

The **Figure 1** shows the various models, transformations and mappings of MDA.

2.1. Computation Independent Model

A CIM is also often referred to as a business or domain model. It describes the actual requirement of the system and hides all technology specific details related to specification. It presents a system independent specification by abstracting the technology specifications and provides a bridge between the domain experts and information technologists which are responsible for the development process. The OMG MDA guide refers to the Computational Independent Model as the highest level of abstraction. In terms of the MDA [7] refers CIM as real world model.

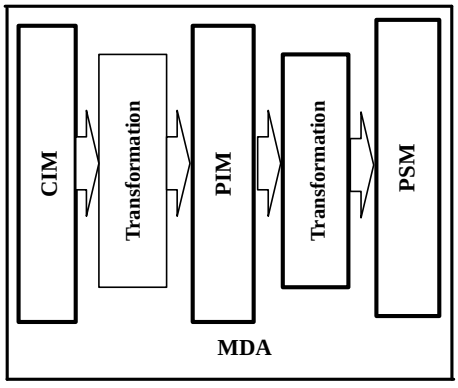

Figure 1. Model Driven Architecture.

2.2. Platform Independent Model

The PIM does not include any platform specific details; it obtains the domain's key features. The main objective of the PIM is to create the precise concept of the system using the vocabulary of the domain as far as possible. A software developer can represent this concept through models in different modeling languages. PIM demands a general representation to capture the semantics of many different domains, which should also be precise enough to support transformation into target code [8].

A PIM exhibits a sufficient degree of independence so as to enable its mapping to one or more platforms [9]. This is commonly achieved by defining a set of services in a way that abstracts out technical details. Other models then specify a realization of these services in a platform specific manner.

2.3. Platform Specific Model

A PSM combines the specifications in the PIM with the details required to stipulate how a system uses a particular type of platform. If the PSM does not include all of the details necessary to produce an implementation of that platform it is considered abstract which means that it relies on other explicit or implicit models which do contain the necessary details. The key features of PSM are as follows [8,10]:

- Specifies how the functionality in a PIM is brought to reality on a specific computing platform.
- Derived from the PIM by adding some plat form-specific details to the standard components.
- Multiple PSMs can be associated with a single PIM.
- Designed to specify the target platform system.

3. Aspect Oriented Software Development

The Software development of complex software applications is a challenging task. Computer hardware and software have evolved together over the years. In the early days, due to hardware limitations, the problems solved by computers were simple, easier and also the software applications written to solve them were not complex. Modern software development or software engineering is more than just coding; it is an iterative process made up of several stages with methodologies to guide each stage and also have tools to support each and every methodology [4].

Aspect-oriented technology is not different from others it just followed the aspect oriented languages and extended the idea to the entire software development

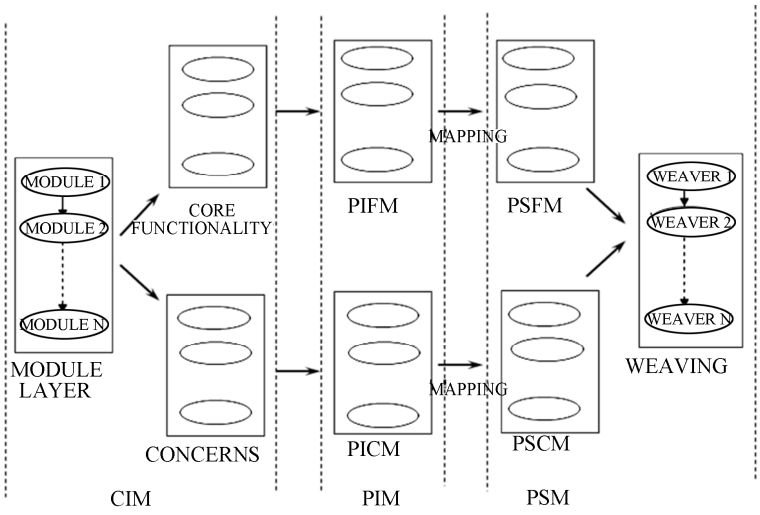

Figure 2. Proposed Architecture (Combining AOSD principles with MDA).

process and forming the field of Aspect-Oriented Software Development (AOSD) [6]. It provides unique and advanced program structuring and modularization techniques.

The implementation of software applications using AOSD techniques provides a better software implementation structure which has an impact on many important software qualities such as improved interoperability, portability, longevity, enhanced reusability and reduced complexity. It is desirable to decompose a system into different modules in such a way that modules responsible for a common concern are strongly coupled and modules responsible for different concerns are weakly coupled. AOSD helps to achieve better Separation of Concerns (SOC) by providing methods to localize cross-cutting con-cerns like security, synchronization, and logging in software artifacts throughout the software development process. Using AOSD the different concerns in real applications are modularized, developed separately and then woven together to create software applications.

4. Proposed Approach

The architecture illustrated in **Figure 2** include the incorporation of Aspect-Oriented Software Development (AOSD)[4] concepts in the Model Driven Architecture approach of software development [2,11]. The proposed architecture makes it more suited to the challenging and changing demands of the present enterprise software development. Analysis and implementation of both functional and non-functional requirement plays equally important role in the development process. The highly reusable MDA is clubbed with the separation of concerns (non-functional requirement) and analyzing them separately with the core requirement. AOSD helps meet the challenges of MDA such as comprehensive modeling standards and mapping technologies with its modularization and weaving concepts. The Architecture uses the base as the MDA with its three broad layers and combines the modularization, concern separation concepts of AOSD. The new three layers of the modified MDA are:

- Modified CIM
- Modified PIM
- Modified PSM

4.1. Modified CIM

The Computational Independent Model Layer is divided into different module layers by combining aspect oriented programming concepts as depicted in **Figure 3** and different module layers are as follows:

4.1.1. Module Layer

It includes the identification of various modules present in the enterprise software system. The application that needs to be designed is broken into sub-systems or modules so as to reduce the complexity of the system and model each sub-system uniquely. The breaking up of the main application into sub-systems facilitates the efficient design and analysis of each part.

4.1.2. Identifying Core Functionality and Concerns

The module layer is further divided into Core Functionality Layer (CF) and Concerns.

- **Core Functionality (CF)**: Each module is analyzed for the core functionality and a CF is prepared.
- **Concerns**: The concern layer consists of the common non-functional requirement in the modules.

The functional and non-functional requirements are separated so that they can be modeled more specifically.

The output of these 2 layers is passed to the next MDA layer, *i.e.*, Platform Independent Model (PIM).

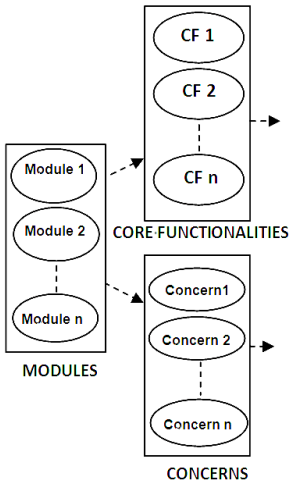

Figure 3. Modified CIM LAYER

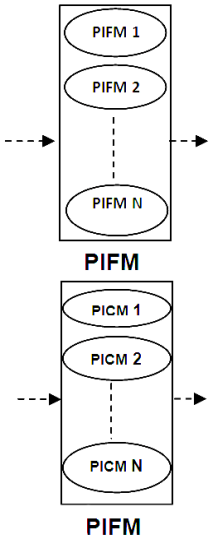

Figure 4. Modified PIM.

4.2. Modified PIM

This model include the specifications of how the things need to be done, hiding the details about the technology that will be employed to implement, the modified PIM as in **Figure 4** contains the separate specification details related to Core Functionality (CF) and the Concerns. The PIM is divided into Platform Independent Functional Module (PIFM) and The Platform Independent Concern Module (PICM).

4.2.1. PIFM

The Platform Independent Functional Module (PIFM) includes the implementation details of the core functionalities of the specific modules that are modeled in the Computational Independent Model. It helps to develop an outline that is technology independent and thus provide a high degree of reusability.

4.2.2. PICM

The Platform Independent Concern Module (PICM) includes the details of how the different concerns would be implemented. As the concerns are needed with various modules so the technology independent code provides great level of reusability.

4.3. Modified PSM

The PSM Layer is broken into two layers namely core functionality and concern layer and weaving layer as shown in **Figure 5**:

4.3.1. Implementing Core Functionality and Concerns

The Layer provides the technology specific implementation of the core functionality and the different concerns identified.

- **PSFM:** Platform Specific Functional Module (PSFM) provides the implementation of core functionality of the modules which is technology specific or platform dependent. The PIFM provided by the PIM layer is implemented using the desired technology.
- **PSCM:** Platform Specific Concern Module (PSCM) provides the implementation of various concerns of the system by a technology specific code. It implements the PICM provided by the PIM layer for each concern.

4.3.2. Weaving Layer

The last layer includes the merging of core concerns with the different concerns needed by the particular module. It contains the set of rules that defines the integration of functional and non-functional requirements of the system providing a one complete system ready to be deployed. In this layer different modules are woven together to create a software application of enterprise systems.

5. Traditional Model vs. Proposed Model

The comparison to the traditional model with the pro-

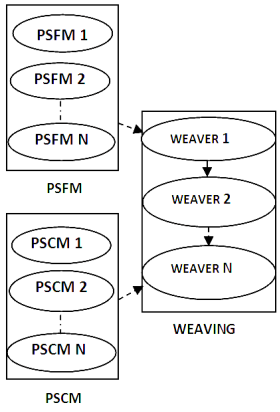

Figure 5. Modified PSM.

posed can be viewed as follows:

5.1. Traditional & Modified CIM

The Traditional CIM provides a system independent specification by abstracting the technology specification. The specifications are monitored as a combined view of functional and non-functional requirement with no specific module discrimination.

The proposed CIM first identifies the major sub modules of the system so that they can be analyzed and designed more specifically and individually. After the module separation the functional and non-functional requirements are separated and modeled. The separation benefits the issue of reusability and ease of modeling.

The consideration of core-functionality and concerns separately is the main implementation focus of the AOSD model, which is incorporated in the MDA approach.

5.2. Traditional & Modified PIM

The traditional PIM provide the platform independent details with presenting domain's key feature. The modified PIM does not include any significant transformation except the modeling of core functionality (PIFM) and concerns (PICM) separately without specifying the platform specific details. The separated modeling benefits in considering g the design of non-functional requirements as well as functional requirement with more concerned focus that leads to efficient design of the system and increases reusability in the model.

5.3. Traditional & Modified PSM

Traditional PSM include the transformation of PIM to PSM including the platform and technology specific details. The proposed PSM include the transformation as same with the traditional but require two transformations i.e., from PIFM to PSFM and PICM to PSCM.

The weaving model presented as an advantage over the traditional model as the layer specifies the rules to group the core functionality with the concerns. The grouping process is dynamic which increases the flexibility and scalability of the module.

6. Conclusions

Enterprise Information System is the main artifact of the enterprise information construction. The study of how to improve the quality, longevity, cost of production and success rate of enterprise Information System is absolutely necessary and significant. The MDA software development approach in the recent years has been able to improve all these above listed factors by separating the concern through abstractions at various levels. This paper has shown that the notion of various highly reusable models in MDA by combining the merits such as better separation of concerns, modularization and weaving concepts of Aspect-Oriented Software development (AOSD). Modified MDA has been proposed which is best suited to the ever changing demand of the enterprise system development. The architecture helps in modeling each and every requirement of the system efficiently.

REFERENCES

[1] D. S. Frankel, "The Model Driven Architecture: Applying MDA to Enterprise Computing," OMG Press, Massachusetts, 2003.

[2] "Object Management Group (OMG), Model Driven Architecture (MDA)," 16 September 2008. http://www.omg.Com/mda

[3] Y. Singh and M. Sood, "Model Driven Architecture: A Perspective," *IEEE International Advance Computing Conference* (*IACC'*08), Thapar University Patiala, India, March 2008, pp, 1644-1652.

[4] Tzilla Elrad *et al*., "Special Issue on Aspect-Oriented Programming," *Communications of the ACM*, Vol. 44, No. 10, October 2001. doi:10.1145/383845.383853

[5] S. S. Alhir, "Understanding the Model Driven Architecture (MDA)," 2003. http://www.methodsandtools.com/archive/archive.php?id=5

[6] Aspect-Oriented Software Development Steering Committee, "Aspect-Oriented Software Development," Available http://aosd.net/

[7] G. Genova, M. C. Valiente and J. Numbiola, "A Semiotic Approach to UML Models," *Proceeding of the* 1*st Workshop on Philosophical Foundation of Information Systems Engineering* (*PHISE'*05), 2005.

[8] G. A. Lewis and L. Wrage, "Model Problems in Technologies for Interoperability: Model Driven Architecture," Software Engineering Institute, Carnegie Mellon University Pittsburgh, Technical Note CMU/SEI-2005-TN-022, May 2005.

[9] R. Heckle and M. Lohman, "Towards Model Driven Testing," *Electronic Notes in Theoretical Computer Science*, Vol. 82, No. 6, September 2003, pp. 33-43. doi:10.1016/S1571-0661(04)81023-5

[10] A. Demir, "Comparison of UML 2.0 and DSLs in the context of MDD," Diploma Thesis, Technische University, Muchen, 2005.

[11] T. O. Meservy and K. D. Fenstermacher, "Transforming Software Development: An MDA Road Map," *IEEE Computer*, Vol. 38, No. 9, 2005, pp. 52-58.

