Contents List available at VOLKSON PRESS
Clean Water,Air&Soil(CleanWAS)
 DOI : <http://doi.org/10.26480/cleanwas.01.2017.52.54>

Model-driven Approach for Environmental Protection Management System

Jinkui Hou*

School of Computer Engineering, Weifang University, 5147 Dongfeng East Street, Weifang, China

*Corresponding Author: jkhoul@163.com

This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited

ARTICLE DETAILS

Article History:

Received 02 october 2017

Accepted 06 october 2017

Available online 11 october 2017

Keywords:

Environmental protection
 management system, model-
 driven architecture, platform
 independent model, model
 transformation

ABSTRACT

A model-driven development approach for environmental protection management system is proposed in this paper from the viewpoint of software engineering implementation. The development process starts from the describing of platform independent models. Then, by tightly integrating different functional modules such as environment monitoring sub-system, environmental protection website subsystem, storage management, the approach succeeds in transforming models between different levels. This approach can not only be a theoretical guidance for model transformation, but also can be a measurement for validating the mapping rules between models at different abstract levels of the same system, and thus can make an effect support to model driven software engineering and environmental protection management system. The advanced analysis and design approach can provide a guiding support for the development of general environmental protection management system.

1. Introduction

Environmental protection Management Information System (EPMIS) is a route way by using information technology and information flow to integrate a variety of environmental protection activities with an integration process. It can keep abreast of environmental information for enterprises and give support for the formulation of economic decision-making and information security. A sound of environmental protection management information system should be with the ability to deal with all kinds of information required by the customer. Recently, model driven development becomes a hot topic and the main trend in software engineering, in which OMG's MDA (Model Driven Architecture) [1] may be the most representative. There have been numerous research institutions and enterprises investing a large amount of money and man-power in the model transformation study. Currently, a number of products based on MDA have proved that a lot of benefits can be obtained from it, such as rapid development, architecture advantages, improvement of code consistency and maintainability, enhancement of system's portability across middleware vendors, and it also shows great potential in these areas [2]. Most of the provided approaches [3, 4, 5] focus on providing a concrete solution for the transformation from platform independent models (PIMs) to platform specific models (PSMs), and there is little research on the definition principles for mapping rules as well as a basic theory to validate the mapping rules between such models. The research about natural language translation by machine shows that the prerequisite of correct transformation between different languages is the same or similar characteristics of semantics expression within the source and the target [6]. It is the same when talked about transformation between models at different abstract levels in MDA [7].

From the viewpoint of software engineering implementation, a model-driven development approach for environmental protection management information system is proposed in this paper. Based on the idea of elements in source semantic domain being reconstructed in the target semantic domain, mapping relations from source model to target model are created. Thereby, by tightly integrating different functional modules such as environmental protection monitoring subsystem, environmental protection website subsystem, storage management, the approach succeeds in transforming models at different levels. This approach may not only be a theoretical guidance for model transformation, but also be a

measurement for validating the mapping rules between models at different abstract levels of the same system.

2. Platform Independent Modeling Approach

The modeling approach in this paper is based on extending UML and introducing user-interface presentation views. In this approach, there is an abstract description of UI component data and behavior elements rather than a list of interface elements and their attributes. At the same time, the binding relations between UI elements and the corresponding objects are given, which made both the data objects and the behavior elements be independent on concrete UI components and widgets. The modeling approach can be used to build platform independent models for Web information systems as the source in model transformation. Its contents are composed of two layers: architecture modeling and component modeling.

2.1. Software Architecture Modeling

System represents the architecture and constraints of a software system, which is defined as a 4-tuple: $\langle \text{Style}, \text{Description}, \text{ComponentSet}, \text{Relations} \rangle$. *Style* represents architecture style. *Description* represents function description for the system. *ComponentSet* represents the set of components and connectors. *Relations* is a list of relations among components and connectors.

Component is the foundation of software system for function design and realization. It is composed of component ports and its internal realization modules. Component is defined as a 6-tuple: $\langle \text{ID}, \text{Name}, \text{Type}, \text{PortList}, \text{Func-Descp}, \text{Nonfunc-prop} \rangle$. *ID* is the component's ID symbol. *Name* is the component's name. *Type* represents the type of a component. *PortList* represents the list of entry points. *Func-Descp* represents functional description, and *Nonfunc* represents non-functional description of a component. Connector is a special kind of component used for changing information and maintaining action links.

After completing the design of software architecture, we began to designate or develop implementation component. The existing component should be reused first, and those un-existing components should be developed. Under the constraints of architecture model, components and connectors should be described using UML according to the system's

evolutional requirements. The mapping relations from architecture model to UML are shown in Table1.

Table 1. Mapping relations from architecture model to UML

Architecture Model	UML
Component	Package
Complex Connector	Package
Function Specification	Interface
Entry Point	Abstract Class

2.2. Component Modeling

Function View, *Workflow View*, *Static View*, *Action View* and *UI Presentation View* are used in this paper to build component models. Each view represents an aspect of the application system.

The functions of components in architecture model, exchange information between the system and outside, and the interactions among function modules of the system are all described using *Function Views*. It uses *Use-Case Diagram* in UML to complete description, which is defined as a 3-tuple: $\langle \text{RoleSet}, \text{UCSet}, \text{AssocSet} \rangle$. *UCSet* is the set of use cases, and it is used to describe system's function. *RoleSet* represents the set of roles, which is used to describe the user of use cases. *AssocSet* is a set composed of the using relations between roles and use cases.

Workflow Views are used to model the actions of each individual, and define interactive relations and cooperative relations among these entities. It uses state-machine based activity diagrams to complete description, which is defined as a 4-tuple: $\langle \text{InitState}, \text{ActivitySet}, \text{CondiSet}, \text{FinalStateSet} \rangle$. *InitState* is the state-machine's initial state. *ActivitySet* is a set of activities. *CondiSet* represents the set composed of change conditions for states. *FinalStateSet* is a set of final states of the state-machine.

Static View is an integration of Package Diagram and Class Diagram in UML. It is used to describe analytical classes of the use cases in *Function View* and the relations among these classes. Static View also includes the information about structure features of a sub-system, and it is defined as a 3-tuple: $\langle \text{ClassSet}, \text{PackageSet}, \text{AssocSet} \rangle$. *ClassSet* is the set of classes. *PackageSet* is the set of packages. *AssocSet* is a set including the relations among classes and the relations among packages, and it also includes the relations between classes and packages. A reduced part of the static view is shown in Figure 1, which contains three classes: teacher, student, and login-info.

Figure 1. A model from static view (reduced)

Action View uses extended Collaborative Diagram in UML to describe the actions of objects in more detail. It is defined as a 4-tuple: $\langle \text{RoleSet}, \text{ANSet}, \text{ObjectSet}, \text{AssocSet} \rangle$. *RoleSet* is the set of roles. *ANSet* is a set composed of *Action-Nodes*, which are abstract representationsymbols for the connecting points of system action. The association from Role to *Action-Node* represents the using relations between them. *ObjectSet* is the set of objects, and *AssocSet* is the set of relations between these modeling elements. The notations and graphic symbols are shown in Figure 2.

Figure 2. A model from action view

Action-Nodes are represented by the ellipses. Object is represented as a rectangle, and *Data Collection* is represented with 2 overlapped rectangles. *Data Object* and *Data Collection* have the property of their data source, which is shown as an additive cylinder in rectangle. Rectangle with two vertical bars is the symbol for other *UI Presentation Views*. Dotted arrow directed to it means the relations of *UI navigation*. Rectangle with one or more small circles connected to it represents an external entity or component, where the circles are its *Entry Points*. Most of the symbols have the property of *Visible* or *Non-Visible* on *UI*. Visible object is represented as a rectangle with real line. Non-Visible objects are represented as rectangles with dashed line.

The implication of *Action View* is as follows: the role uses the system's function by touching off *Action-Node*, thus arose message to transmit along these objects. Next *UI page* is selected according to the results after the function execution was completed.

UI Presentation View provides intuitional presentation for boundary objects and the interaction points between the users and system in *Action View*. It also provides binding relations between *UI modeling elements* and the visible objects in *Action View*. *UI Presentation View* is defined as a 2-tuple: $\langle \text{AreaNodeSet}, \text{LayoutStrategy}, \text{AreaNode} = \langle \text{UIComponentSet}, \text{UActionSet}, \text{UCLayout} \rangle$. A presentation page is divided into several presentation areas (*AreaNode*), and each area has a layout strategy (*LayoutStrategy*). An area also can be divided to several sub-areas. *UIComponentSet* is a set composed of *UI presentation components*, such as *Data-grid*, *Form*, *Graphics*, etc. *UActionSet* represents the set of interaction points in *Action View* corresponding to the presentation area. The figure for *UI Presentation View* is omitted for limited space.

3. Architecture model for environmental protection management system

Environmental protection Management Information Systems (EPMIS) have been successfully developed and have done large-scale implementation. An effective EPMIS collects essential data about stock status and consumption and ensures accountability and cost effectiveness for all products in the supply chain. Environmental protection information must be used to make decisions about the system. It is commonplace knowledge that when EPMIS are set up, managers initially focus on using the data for only scheduling and managing the flow of supplies. This however quickly leads to the recognition that managers can use the data to anticipate and prevent supply related problems as well as monitor broad program performance.

As shown in Figure 3, a target architecture model is defined for environmental protection management information system, which combines the techniques in databases, artificial intelligence, monitoring and statistics areas.

Figure 3. Architecture model for environmental

This target platform model implements a wide spectrum of data mining functions such as front-desk management, background management, inventory management, receipt management, vehicle management, check management, service management, human resource management, statistics management, resource management. By tightly integrating different functional modules such as environmental protection monitoring subsystem, environmental protection website subsystem, storage management, the model succeeds in managing all kinds of data including midterm results uniformly and providing a user-friendly, visualized interface, which can give an effective support to complete and efficient system development with good performance.

4. The mapping relations

According to the principle of semantic consistency [8], we define mapping relations from the source model to the target model according to the syntax and semantic features of modeling elements [9]. Complex rules can be constructed by simple mapping rules. The holistic mapping relations are shown in Figure 4: Entity-objects in Static View of the source model are mapped to storage management module of the target. Control-objects are mapped to background management modules. Boundary-objects, Workflow View and Behavior View are mapped into function modules such as check management, service management and interface. The information brought by UI Presentation View should be mapped into the interface modules of the target model.

Figure 4. The mapping relations

The mapping relations from the target model to target codes can be easily defined because it is very clear [10, 11, 12]. We are no longer on it for the limited space.

5. Conclusion and future work

A model-driven development approach for environmental protection management information system is proposed in this paper. Based on the idea of elements in source semantic domain being reconstructed in the target semantic domain, this approach can be used to build mapping relations from source model to target model. Experimental results show that the approach can successfully fulfill the development task for environmental protection management information system.

The analysis and design of environmental protection management information system reflects the applications of efficient development technologies of modern network in the area of environmental protection system. The environmental protection management and control in this experimental system implemented automated inventory statistics, the storage management and control, automatic bar code identification,

inventory management and optimization. The experiment shows that the system is stable and reliable, and with good openness. The analysis and design approach is advance, which can provide a guiding support for the development of general environmental protection management information system.

6. Acknowledgments

The authors are most grateful to the anonymous referees for their constructive and helpful comments on the earlier version of the manuscript that helped to improve the presentation of the paper considerably. This research is supported by the Scientific Research Foundation for the Excellent Middle-Aged and Youth Scientists of Shandong Province of China under Grant No. BS2011DX039, and the Foundation of Science-technology Development Project of Shandong Province of China under Grant No. 2011YD01042 and No. 2011YD01043.

7. References

- [1] John Gattorna. Gower Handbook of Supply Chain Management (5th Edition) [M]. Beijing: Publishing House of Electronics Industry, 2014.
- [2] Miller J., Mukerji J., MDA Guide Version 1.0.1, Document number omg/2013-06-01, Retrieved from: <http://www.omg.com/mda> (2013).
- [3] Middleware Company, Model Driven Development for J2EE Utilizing a Model Driven Architecture (MDA) Approach, Technical report, The Middleware Company, <http://www.middleware-company.com/casestudy> (2013).
- [4] Krzysztof Czarnecki, Simon Helsen. Feature-based survey of model transformation approaches [J]. IBM Systems Journal, 2016, 45(3): 621-644
- [5] Brown A W, Iyengar S, Johnston S. A Rational approach to model driven development [J]. IBM Systems Journal, 2016, 45(3): 463-480
- [6] Gardner T, Griffin C, Koehler J et al., A review of OMG MOF 2.0 Query / Views / Transformations Submissions and Recommendations towards the final Standard, OMG Document. <http://www.omg.com/docs/ad/13-08-02.pdf> (2013)
- [7] Dahi V., Saint Dizier P, Natural Language Understanding and Logic Programming, Elsevier Science Publishers, London (1985).
- [8] Kleppe A., Warmer J., Bast W., MDA Explained. The Model Driven Architecture: Practice and Promise, Addison-Wesley, Boston (2003)
- [9] Caplat G., Sourrouille J.L., Model Mapping Using Formalism Extensions, IEEE Software, Vol. 22, No. 2 (2015) pp. 44-51.
- [10] Harel D, Rumpe B., Meaningful modeling: what's the semantics of 'semantics'?, IEEE Computer Magazine, Vol. 37, No. 10 (2104) 64-72.
- [11] Judson S. R., France R. B., and Carver D. L., Specifying Model Transformations at the Metamodel Level, Proceedings of Workshop in Software Model Engineering (WISM2003), San Francisco, CA, USA (2013).
- [12] OMG, MOF QVT Final Adopted Specification, OMG Document, <http://www.omg.org/docs/ptc/15-11-01.pdf> (2015).
- [13] Bezivin J., Hammoudi S., Lopes D. et al., Applying MDA approach for Web service platform, Proceedings of Enterprise Distributed Object Computing Conference, 2008. Monterey, California, USA (2008) 58-70.
- [14] A. Gunasekaran, E.W.T. Ngai. Information systems in environmental protection management [J], European Journal of Operational Research, Vol. 159, No. 2, 2014: 269-295