Model-Based Approach for Reporting System Development

Jinkui Hou

Abstract From the viewpoint of software engineering implementation, a model-based development approach for reporting systems is proposed systematically based on the concern of separating application descriptions and UI designs. The development process consists of four steps: data modeling, report modeling, model transformation, and code generation. The experiment shows that this approach enhances the efficiency and quality of reporting system development, which can be well combined together with other application development frameworks, and thus can support model-driven software engineering effectively.

Keywords Software engineering · Model-driven development · Reporting system · Modeling approach

1 Introduction

Reporting system is a very important subsystem in the business application system, which is used frequently. It is a heavy task to develop embedded reporting system, and the product is with vulnerability of short life cycle. Therefore, research of automatic code generation for reporting system can reduce the workload of system development, and enable the system to meet the complex application environments. The current Web-based applications are gradually replacing the traditional C/S mode software, which becomes the mainstream of application software. It is also an urgent need for reporting system to adapt to this situation. Most of the existing reporting tools do not have the learning function, in which the versatility is not enough [1]. It generally cannot generate reports of different styles in the

J. Hou (✉)
School of Computer Engineering, Weifang University, 261061 Weifang, China
e-mail: jkhou@163.com

F. Sun et al. (eds.), *Knowledge Engineering and Management*,
Advances in Intelligent Systems and Computing 214,
DOI: 10.1007/978-3-642-37832-4_19,

same run-time, and cannot meet the needs of generating reports of real-time based on user requirements. When the report format change greatly, it is difficult to add new reports to meet user requirements dynamically.

Model-driven development has become a hot topic and main trends of software engineering, which enhances abstraction level to deal with the complexity of software development through the application of models and modeling techniques. OMG's model-driven architecture (MDA) [2] provides theoretical support for automatic transformation between models. Through in-depth understanding of traditional approaches of report generation and basic requirements of reporting system, this paper provides a MDA-supported development model for Web reporting system. It can be well combined together with other application development frameworks. Thereby, it is a good application development mode with many merits, such as simple, intuitive, automatic generation of target code, and so on.

2 MDA-Supported Development Model for Web Reporting System

On the basis of the model-driven development model named ASLP [3] proposed in our previous study, a development model for Web reporting system is proposed in this paper based on the theory of MDA [4, 5]. This model support model-driven software development, which is shown in Fig. 1.

The model mainly comprises four parts: data model, report model, model transformation engine, and code generator. Data model is the basis for the whole framework of the model, which separate the report module from the actual data source. It reduces the coupling degree of the report module and the actual data source, and ensures the continuity of the knowledge of specific report definitions. It also makes the information of report definition to be no longer depended on the actual database table. Report model describes the realization of the content and format of the report which is customized by the user, which provide all the parameter information for target code generation. Model transformation engine is used to achieve model transformation from abstract model to specific platform model, while code generator generates final source codes of the report system.

2.1 Data Model

Data model comprises two parts: functional view and data object view. Functional view is used to determine the requirements of report pages on the inner model and relationships between interfaces through the analysis of user's needs. Data object view is used for modeling domain concepts, which describe the object class used

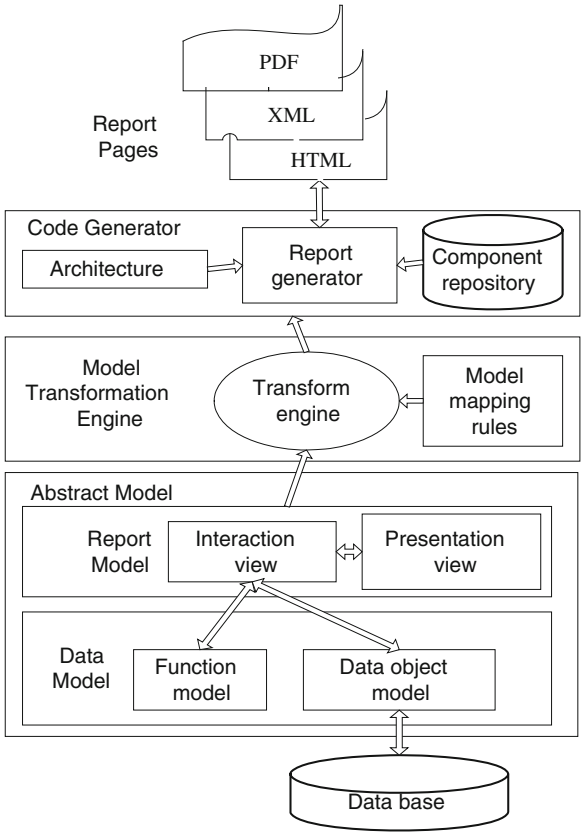

Fig. 1 Model-driven development model for Web reporting system

in application systems. It describes the data objects and their relationships required by the reports from static aspects.

The extended use case diagrams in UML are used in functional view to describe user requirements. Through analyzing user requirements, functional view can be used to determine the functions and the framework of user interfaces, the relationship between user interfaces as well as the requirement on inner models of interface presentation.

Static view of ASLP describes the composition and behavior of objects from the view of abstract calculation relationships, which cannot meet the requirements of establishing interface structure. In order to meet the requirement of the customization of report interface, object view is built by expanding static view of ASLP. In addition to considering the general composition and behavior of objects, object properties used for interface presentation and code generation is introduced and the constraints on the realization of specific interactions is minimized to the maximum extent. Expansion of the data object view includes the following aspects. Object properties are expanded by adding UI type, default values, labels, and units of measurement, and the description of data types is also extended in

order to provide information of interactive units of user interface. Deduction relationship and linkage relationship are proposed to respectively represent correlation and association between UI objects. The description of property group is provided in order to meet the psychological and visual requirements of users. Object relationships are used to express the navigation between use interfaces.

In the following formal description, the expressions enclosed by {} indicate optional parts, and those ones enclosed by <> indicate keywords.

The keyword followed by an asterisk (*) indicates that the object may have zero or many. The symbol indicates + more than one instances. The symbol = means the definition. Thus, the structure of classes and objects are defined as follows:

<Class>:=<ClassName>(Group+|<Attribute>+)+<Method>*<Attriblink>*;

<Object>:=<ObjectName>:<ClassName>

In the above expressions, <ClassName> is the name of class, and <ObjectName> represents the name of object. <Group> means the set or group of objects, and <Method> is represents method of class, and <AttribLink> is the linkage relationship between attributes.

The expansion description of properties is defined as the following expression:

<Attribute>:=<AccessType><DataType>(<AttribName>{=<DefaultValue>}) {<ValueRange>}{<Unit>}{<DataSource>}{<UIType>}{<Label>}

In the above expression, <Attribute> represents property, and <AccessType> represents the visibility at runtime. <DataType> is the type of data, and <AttribName> is the name of a property. <DefaultValue> represents the default value of a property, and <ValueRange> represents the range of value. <Unit> is used as measurement units. <DataSource> represents for the source of values. <UIType> represents the type of visual objects, and <Label> is the label attached to visual objects.

The concept of attribute groups is also added in the model to support the generation of user interface. Attributes of the same group can be combined together by a frame and used as a whole. In complicated situation, the group is divided into sub-groups, and forms a nested relationship.

Attribute group is a gathering of attributes of sub-groups or properties, which is depicted with the following manner.

<Group>:=[<GroupName><Attribute>+]|[<GroupName>{<Attribute>+ |<Group>+}+].

In the above expression, <GroupName> represents of the group name, and <Attribute> represents its properties.

2.2 Report Model

Report model consists of interaction view and presentation view. Interaction view is used to describe the system from dynamic aspect, which is also an abstract description of behavior of user interface and provides the internal association between UI behaviors and system functions. The function of UI presentation view

is to show the layout and forms of the interface according to the inner models (data object view and interaction view) and user requirements for data presentation. It provides a full description for the intuitive presentation of user interface, and provides the binding relationship between the interface elements and the visible elements of interaction view.

Interaction view plays a key role of connecting link between the preceding and the following in the whole framework, which is shown in Fig. 2. On the one hand, interaction view, function view, and data object view are organically combined together to form abstract outline of the interface. On the other hand, the classification of interaction views is the basis of the retrieve of interface template in presentation model. That is to say, the type of interface templates is determined by the interaction relationship between objects of interaction views. It provides constraints for the users to select the appropriate interface and ensures the correctness of UI generation. In addition, the degree of abstraction of interactive view is lower than that of the functional model and object model. Compared with them, interaction model is more close to the interface.

Data objects of interactive view is the instance of the data objects defined in object view. Data object of object view is the concept of class, and data object of interactive view is the instance of class applied in specific user interface. Data objects of object view are instantiated in interaction view, which can be further subdivided to determine the appropriate presentation form according to its role in the description of different interfaces at the same time. Interactive view is defined as a triple $<V(G), E(G), \varphi_G>$, where $V(G)$ is a collection of interacting objects, and $E(G)$ is the set of interactive relations, and φ_G is a function from the set of interactions to the set of ordered pair of interactive objects.

Interactive objects is entity objects in UI which can interact with other objects, such as data object, collection, user, use case, page reference, and so on. Data objects of interactive view come from data object view and serve for the corresponding user interface. After being presented in UI, the corresponding interface elements can be in various forms of navigation links and a variety of forms, such as text box, password boxes, radio buttons, check boxes, and so on. Interactive relationship is interactions between interactive objects, which emphasize interactive behavior between objects and use cases as well as the influence to the relationship between UI objects, which including calls, participation, information access, and navigation. Invoking relationship refers that one object calls the method of another object and returns the results of the function.

In interactive view, the relationship between user to use case is use case calling, which reflects on UI is the click on buttons. The calling relations between use cases and data objects are trigged by causing the behavior of data object or data collection. Participation relationship is used to express data providing, where the general pointer is from the object or collection to use cases, which indicates that the users provide data or parameters. Message connection is used to express message transfer among objects and visible data objects or data collection. It can also be message transfer from an event trigger to component, or message transfer

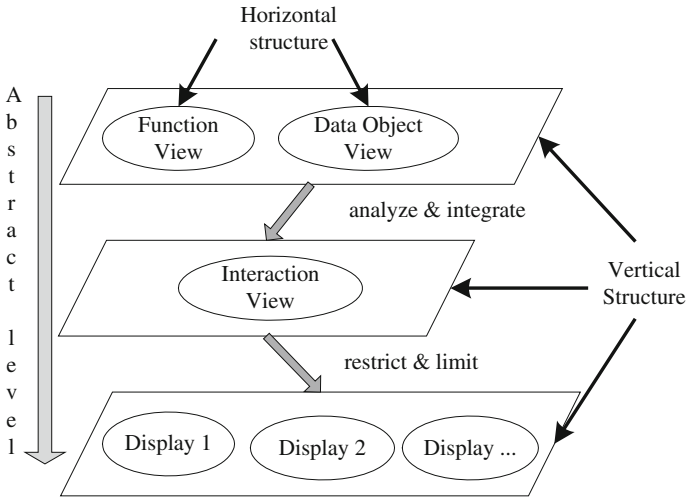

Fig. 2 The role of interaction view

from visible objects or data collection to component. Navigation relation refers page forwarding operation inspired by use case.

UI presentation view provides a constraint environment to show the layout of user interface. Interactive view is the inner basis of UI presentation view, while UI presentation view is the external depiction of interactive view. UI presentation view is used to describe the appearance of UI on the basis of interaction view, which mainly deals with macro layout of UI and display-control, and provide service for automatic code generation of graphical user interface. It embodies abstract interface based on the inner model of UI, thus solve the problem of the overall layout of user interface. UI presentation mainly includes template object, area object, dividing line object, and template interactive objects. Its comprehensive introduction can be seen in [3], and here we are no longer on it.

3 Model Transformation and Code Generation Based on ASP.NET

ASP.NET is a framework widely used for Web application development [6]. C# is used as the target code in the experiment introduced in this paper.

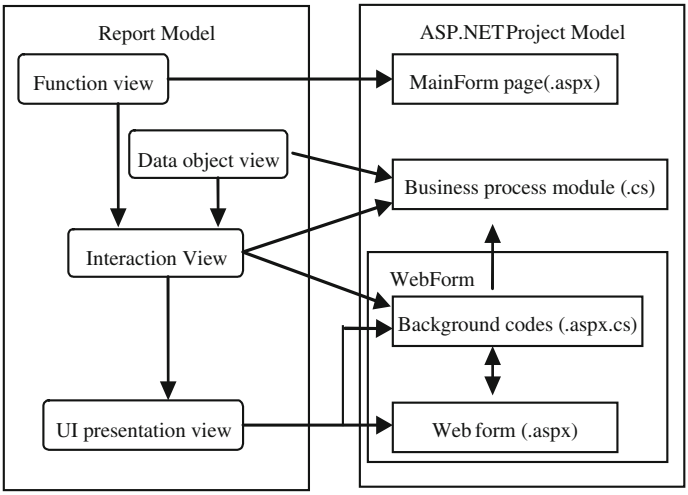

Fig. 3 Mapping relations from report model to target model

3.1 Model Mapping Relations

The mapping relations from report model to ASP.NET project model are shown in Fig. 3, in which the corresponding generated codes mainly include project information, business processing module, Web forms, and background codes.

Object view is mapped to the business processing module in the generated ASP.NET project, which provides corresponding support for interactive view and UI presentation view. The compound use case of functional view is mapped to functional selection items of the target project. Interaction view combined with module processing information of UI presentation view are mapped to the background codes of the generated Web pages (**.aspx.cs* files). Use cases of interaction view are mapped to operations of corresponding UI control objects, such as menu, button or hyperlink, and so on. Method invoking relations of the source model are mapped to operation call of the corresponding object. The navigation relations are mapped to display operations of the target pages. The template object of UI presentation view are mapped to Web pages, in which object information are mapped to attribute information of presentation elements of Web forms, such as the type, location, size, color, and other information (**. aspx* files).

3.2 Code Generation

The algorithm of target code generation is mainly responsible for the generation of reporting framework, interface elements, the target program for print, and preview of reports. The algorithm flow is shown in Fig. 4.

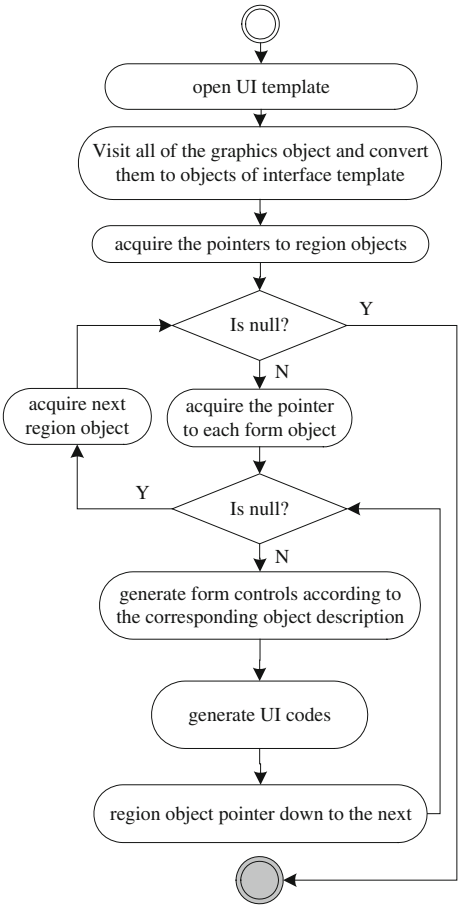

Fig. 4 The algorithm flow for generating target codes

The algorithm for generation of reporting framework is the entrance to the whole automatic code generation, which is most upper algorithm of the UI generation for the reporting system. The user interface, interface elements, and the layout of interface elements are generated according to the information provided by interface templates. The appropriate background codes are generated on the basis of interaction view corresponding to UI template. In the algorithm flow, each region of object UI template is visited and the corresponding code generation algorithm is invoked according to the data types and presentation form of each display unit.

The generation algorithm of interface elements is the algorithm for generation of the basic elements of UI, which is mainly used to generate specific interface controls. It is a recursive algorithm because some of the controls may also contain sub-controls, such as *Frame*, *Tabstrip*, and so on.

The generation algorithm for report printing and preview is mainly used to generate the print and preview program codes. The print and preview module is a

core module of the reporting system, which is output part of the whole system. The basic process to achieve report printing and preview on .NET platform is described as follows: we should first define the layout of the printed page, and then define each print area which includes the size of the print area, and the text, graphics, and images needing to be printed. Finally we call the system methods (e.g., *DrawString*() and *DrawRectangle*()) to output the corresponding text, graphics, and images.

4 Conclusion and Future Work

A model-based development approach for reporting systems is proposed systematically in this paper, which can be well combined together with other application development frameworks, and thus can provides reporting capabilities for application system and has a certain capacity of software reuse. It can dynamically add new reports to meet the requirements of users. This approach follows the essence, process, and requirements of model-driven software development, which can make an effective support for model-driven software engineering.

Future works are as follows: (1) to further improve the description of interaction view, and enhance its ability of semantic interpretation; (2) to fully abstract and describe the UI presentation view, and enhance visual attractiveness of the generated pages; (3) to diversify target platform in order to verify the practicability of this approach.

Acknowledgments The author is 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 was supported by the foundation of science-technology development project of Shandong Province of China under Grant No. 2011YD01042 and No. 2011YD01043.

References

1. Hailpern B, Tarr P (2010) Model-driven development: the good, the bad, and the ugly. IBM Syst J 45(3):451–461
2. Miller J, Mukerji J (2011) MDA guide version 1.0.1 (document number omg/20011-06-01). http://www.omg.com/mda
3. Hou J, Wan J, Yang X (2006) MDA-based modeling and transformation approach for WEB applications. In: Proceeding of the sixth international conference on Intelligent System Design and Applications (ISDA), pp 867–812. IEEE Computer Society, New York
4. Kleppe A, Warmer J, Bast W (2009) MDA explained, the model driven architecture: practice and promise. Addison-Wesley, Boston
5. Thomas D (2009) MDA: revenge of the modelers or UML utopia? IEEE Softw 21(3):15–17
6. Jeffrey R, Francesco B (2009) Applied Microsoft.NET framework programming. Microsoft Press, Washington