An Executable Business Model for Generic Web Applications

Zhenxiang Chen
Shandong Provincial Key Laboratory of Network Based Intelligent Computing
University of Jinan
Jinan, China
lebolx@gmail.com

Kun Ma
School of Computer Science and Technology
Shandong University
Jinan, China
nic-makun@ujn.edu.cn

Ajith Abraham
Machine Intelligence Research Labs (MIR Labs)
Scientific Network for Innovation and Research Excellence, USA
ajith.abraham@ieee.org

Bo Yang
Shandong Provincial Key Laboratory of Network Based Intelligent Computing
University of Jinan
Jinan, China
yangbo@ujn.edu.cn

Runyuan Sun
Shandong Provincial Key Laboratory of Network Based Intelligent Computing
University of Jinan
Jinan, China
sunry@ujn.edu.cn

***Abstract*—In this paper, a novel platform-specific executable business model called *xBM* is proposed to sufficiently represent the business logic process. The metamodel of xBM have been analysed, formalised and illustrated. Traditional version control idea based on the copy-modify-merge approach is also applied in *xBM*. Moreover, our approach for model development approach based on *xBM* is presented for generic web applications. Finally a case study of simplified Provincial Excellent Academic Degree Thesis Review System is demonstrated how to build *xBM*s and transform them into the final system.**

Keywords- Model-driven Software Development; Business Process Modeling; Model Transformation; Version Control; Web-based Applications

I. INTRODUCTION

The Model-Driven Architecture (MDA), initiated by Object Management Group (OMG), starts with the well-known and long established idea of separating the specification of the operation of a system from the details of the way that system uses the capabilities of its platform [1]. A model of a system is a description or specification of that system and its environment for some certain purpose, which is divided into platform independent model (PIM) and platform specific model (PSM) [2]. Model-Driven Software Development (MDSD) techniques are increasingly used in large development environments where different team members work on distinct executable models. However, MDD languages, such as UML, do not provide explicit abstractions to capture the business process [3].

The rest of the paper is organized as follows. Section 2 discusses related work. In Section 3, the metamodel of executable business model (*xBM*) is presented and then discussed in details. Section 4 discuses the model transformation approach based on *xBM*. In Section 5, a case study of Foundation Project Management System is demonstrated using this model-driven approach. Finally conclusions are provided in Section 6.

II. RELATED WORK

UML, defined in a MOF-compliant language, has been chosen to describe platform independent model (PIM), since it is a specification language that is widely used in the context of Model-Driven Architecture [4]. Only the PIM can not meet the new requirements or changed business rules.

However, there are no unique specifications for PSM. For different application fields, there are different models. Architecture Analysis and Design Language (AADL), for its strict syntax, enables the creation and exploration of PSMs in the development of embedded real-time system. The AADL draft standard includes a UML profile of the AADL, which after being approved as part of the SAE standard [5]. Object Management Group (OMG) proposes a PSM specification with CORBA Interface Definition Language for Super Distributed Object, which is a logical representation of a hardware device or a software component that provides well-known functionality and services [6]. The Java Metadata Interface (JMI) Specification, which is based on the Meta Object Facility (MOF) specification from the OMG, defines a dynamic, platform-neutral infrastructure that enables the creation, storage, access, discovery, and exchange of metadata [7].

III. METAMODEL OF EXECUTABLE BUSINESS MODEL

This paper proposes a novel platform-specified executable business model called *xBM* for generic web applications. *xBM* defines business service, business domain object and user interface of web applications. In order to

978-1-4244-7818-7/10/$26.00 ©2010 IEEE

reduce the complexity of models, *xBM* is divided into three tiers: business service model, business domain object model and web presentation model. Models in different tiers are relatively independent with specific responsibilities and loosely coupled structure. A separation of design concerns into distinct model layers has several advantages such as ease of maintenance, the ability to select specialized tools and techniques for specific concerns. The metamodel of *xBM* is shown in Figure 1. In the *xBM*, systems are modeled as hierarchical collections of *MObject*, *MAttribute*, *MFrame*, *MElement* and so on.

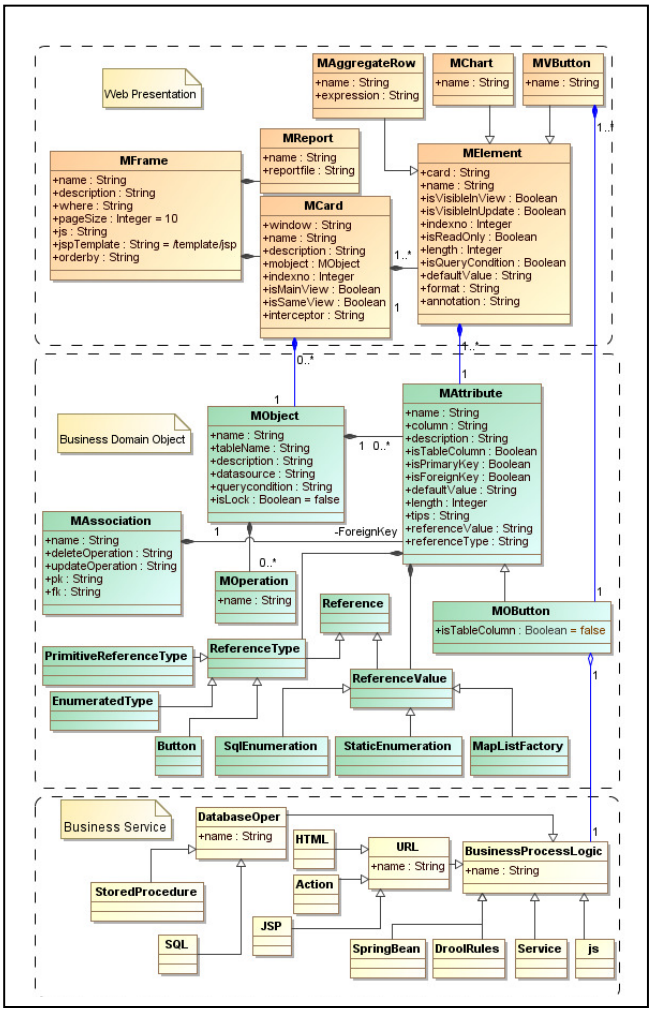

Figure 1. Metamodel of A Novel Executable Business Model.

A. *Architecture of Executable Business Model*

Business service model describes the basic business process of system including CRUD (create, read, update and delete) business and the following special business. Generally, the specific business process is triggered by clicking user interface button of web applications. The representation of business logic includes database-related operations, URL, service interface, action, JavaScript, Drools rules [8] and so on.

(1) Database-related operation is direct operation of database, such as SQL statements and stored procedure;

(2) Service interface is a service class implementing user-defined business logic;

(3) JavaScript defines some scripts based on browser;

(4) URL means navigation of a JSP page or HTML page;

(5) Action is to execute a Struts2 action;

(6) Drools rules [8] is a representation of business rules, and a production rule is a two-part structure using first order logic for reasoning over knowledge representation.

Business domain object model is a kind of platform-specified business data entity, which includes *MObject*, *MAttribute*, and *Reference*. *MObject* defines business object name, description, table mappings (corresponding to relational database table transformed from data entity of PIM), and query condition (values range of the instance of *MObject*). *MAttribute* describes the mapping from table columns of relational database or user-defined button to the property of business object.

The most important property of *MAttribute* is the *reference* and *foreign association* with other business objects.

(1) *Reference* is made up of reference type and reference value. Reference type can be further broken into *primitive data type*s and *special reference type*s, containing information passed from business object tier to presentation tier. *Primitive data type* is the data type identified by the system, such as *string*, *integer*, *uuid* and *stringdate*. While *special reference type* includes user-defined button and enumerated data type. These references need reference value that is additional information for reference type. For the enumeration data type, the reference value is a series of concrete enumerated value or the list of data from the SQL and JAVA codes. And the value of button is the business logic processing JAVA class name.

(2) Foreign association means that the *MAttribute* of business object is the foreign key of another business object, which generally represents master-slave relationship between business objects.

WEB presentation model is composed of *MFrame*, *MReport*, *MCard*, *MElement*, *MChart*, *MButton* and MAggregateRow.

(1) *MFrame* is the entrance of operation of business object, which presents CRUD of business objects and other business. MFrame is composed of many *MCard*s and *MReport*s, including only a main *MCard*, some other detail *MCard*s and some *MReport*s;

(2) *MCard* is the element of *MFrame* for the sake of the maintenance a business object. A *MCard* is composed of some *MElement*s;

(3) *MReport* is also an element of *MFrame* for the web report;

(4) *MElement* is the smallest unit of web presentation, which may be the attribute of business object, user-defined button or web charts. The position of *MElement* can be adjusted to display in a form or a grid in Page Layout Manager. User-defined button is a kind of *MElement* adding specific business logic to the system. *MChart* is a kind of

web statistics graph, displaying professional quality charts in web applications;

(5) *MAggregateRow* is total value of some *MElement* using MDA expressions.

B. Extension Mechanism of Executable Business Model

xBM Interceptors include logic that is triggered by specific events. In the basic case, interceptors are inserted between a caller and a callee for method execution, which is defined in Spring Bean. For instance, adding interceptor to *MCard* to do some extra work before or after maintenance of business data.

In order to provide a flexible extension mechanism, a variety of MDA variables and expressions are provided to help developers customize the pages or models. The return type of expression is String, which is replaced when running. If the return type is object, the value is return value of method *toString*. Common types of expressions are session variables, attribute value of business object, JAVA variables and request variables. $S{parameterName} is used to represent session variable, the value is null if the parameter dose not exist. For instance, the username of current user is represented as $S{user.userName}. It is replaced with code "*session.getAttribute("user").getUserName()*" after model synchronization. $O{[TableName.]columnName} is used to represent the attribute value of business object. $I{ClassName.methodName} means invoking the static method of class. Finally, $R{parameterName} returns the value of specific parameter of HttpServletRequest.

C. Version Control of Executable Business Model

The *xBM* is to describe the system business to the utmost. When the current *xBM* can not represent more complicated business, the only thing to do is extending the metamodel of *xBM* to satisfy the requirements.

Moreover, *xBM*s are saved as XML format file. Therefore, conventional version control systems can also man-age the models based on the *copy-modify-merge* approach [9] shown in Figure 2. In this solution, modelers contact the repository and create a personal working copy (a snapshot of the repository's files and directories). Modelers work simultaneously and independently, modifying their private copies. Finally, he private copies are merged together into a final version. The models of maximum version number formulate *xBM*s. *xBM* can be restored to a previous version according to the revision number, which is useful when you have made some fatal mistake in modeling the system.

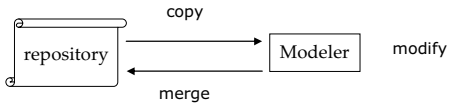

Figure 2. Modeling Process of xBM.

IV. Model-Driven Development Based on Executable Business Model

This section proposes a a model-driven development approach based on *xBM*.

A. Modeling of Executable Business Model

Some data entities in UML class diagram are extracted from the business requirement. In order to describe model transformation rule from data entity to business domain object model, the representation method of model element in our previous work [10] [11] is adopted. The transformation rule *r* is further defined in predicate logic of mathematical relation. *MObject* is transformed from *Class* in class diagram, and its attribute *name* is the same as the name of *Class*; *MAttribute* is transformed from *Property* in class diagram, its attribute *name* and *column* is the same as the name of *Property*, its attribute *isTableColumn* is true, its attribute *referenceTye* is the same as the type of *Property*, its attribute *length* is the length of the type of *Property*; *Property* with tag *PK* is mapped into primary key of *MAttribute*; *Property* with tag *FK* is mapped into foreign key of *MAttribute*.

$$
\begin{aligned}
&r=\{<s,t>|\ s\square PIM\square t\square xBM\wedge \\
&(\forall(s\in Class) \\
&\rightarrow \exists(t\in MObject\wedge t.name=s.name\wedge t.tableName=s.name))\ ①\wedge \\
&(\forall(s\in Property) \\
&\rightarrow \exists(t\in MAttribute\wedge t.name=s.name\wedge t.column=s.name\wedge t.isTableCo \\
&lumn=true\wedge t.referenceType=s.type\wedge t.length=len(s.type)))\wedge ② \\
&(\forall(s\in Property\wedge s.type\in<<PK>>) \\
&\rightarrow \exists(\ t\in MAttribute\wedge t.isPrimaryKey=true))\ \wedge ③
\end{aligned}
\tag{1}
$$

In order to obtain more detailed *xBM*s, the behavior of components must be specified. Some *xBM*s such as *MFrame*, *MCard, MReport*, *MElement*, *MChart* and *MButton* can be obtained by modeling the J2EE business accord with the metamodel in Fig. 1.

B. Transformation from xBMs to Codes

The transformation is based on textual template engine, is made up of some plugin modules called plugins. Each plugin module is a series of template files syntax-based *FreeMarker* and some Java helper classes, generating code based on some open source libraries (Such as SQL, Spring, Hibernate, Struts, JSF, WebService, etc.). All the data that the template can use is from *xBM*s. Developers can easily write personalized plugin to expand current application and generate code independent of language.

Take the template file of data query JSP for example. The template is shown in Fig. 3.

V. Case Study

The model-driven development is illustrated in details by a case study of the review module of school of simplified Provincial Excellent Academic Degree Thesis Review System to validate the above model transformation approach.

System users include students, school administrators and provincial degree committee. The principal business logic of this application is as follows: degree thesis is uploaded to the system by students, and can be printed after the adoption of school; provincial degree committee selects experts to review these papers; experts log on the system to review some degree thesis to submit advices.

Due to lack of space, take the review process of school administrators for example. School administrators log on the system to review relevant degree thesis. Some advices are provided if the thesis is rejected. The adopt thesis is then submitted to the provincial degree committee. The model-driven development of this application is as follows.

```
<jmesa:tableModel id="tag" items="${r"$"}{data}" var="map">
<jmesa:htmlTable>
<jmesa:htmlRow
uniqueProperty="${mcardList[0].melementList[0].mattribute.column}">
<jmesa:htmlColumn property="select" >
<input type="checkbox" name="primaryKeys"
value="${r"$"}{pageScope.map.${mcardList[0].melementList[0].mattrib
ute.column}}"/></jmesa:htmlColumn>
<#list mcardList[0].melementList as melement>
   <#if melement.queryvisible?string=="true">
      <#if melement.mattribute.referencetype=="string">
<jmesa:htmlColumn property="${melement.mattribute.column}"
title="${melement.description}"
width="${melement.length*PIXELPERCHAR}"  /></#if>
...
<#if melement.mattribute.referencetype=="enum">
   <jmesa:htmlColumn property="${melement.mattribute.column}"
title="${melement.description}">
   ${r"$"}{mda:getEnumOption('${melement.mattribute.referencevalue}'
,map.${melement.mattribute.column}).value}
</jmesa:htmlColumn></#if>
…
</#if></#list>
</jmesa:htmlRow></jmesa:htmlTable></jmesa:tableModel>
```

Figure 3. JSP Template Fragment of Data Query of CRUD.

A. *Modeling of Executable Business Model*

Data entities represented by UML class diagram are modeled according to the business requirement. School administrators log on the system, and review the degree thesis first. As is shown in Figure 4, *ThesisInfo* is degree thesis, and *ThesisFruit* is the relevant fruit of degree thesis.

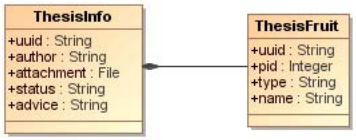

Figure 4. PIM Fragment of Simplified Provincial Excellent Academic Degree Thesis Review System.

Business objects *O_ThesisInfo* and *O_ThesisFruit* shown in Figure 6 are generated from the data entities shown in Figure 4 according the transformation rule 1 shown in Equation (1), and the *MAttribute* such as *A_Thesis_advice*, *A_Thesis_attachment*, *A_Thesis_status*, *A_Thesis_uuid*, are transformed from the property of data entity shown in Figure 4 according the transformation rules 2-4 shown in Equation (1).

Additional *xBM*s accord with the metamodel shown in Fig. 1 are created based on the generated above models. Enumeration of static review state is defined with the value approval (state of adoption), submit (state of submission) and reject (state of rejection). The buttons *A_Thesis_adoptbutton*, A_Thesis_rejectbutton, and A_Thesis_exportbutton are defined to review and print the thesis.

Web presentation models are created based on the business object. The web frame *ThesisReview* includes two cards (main card *C_Thesis* and detail card *C_Fruit*). The card *C_Thesis* contains fields of author, thesis name and the attachment of paper. Some buttons such as reviewing and printing is defined with *E_Thesis_adoptButton* and *E_Thesis_rejectButton*. The final xBM is show in Figure 6.

B. *Transformation from xBMs to Codes*

Directly from the *xBM*s shown in Fig. 6, the textual template engine generated the JSP CRUD codes. In this application, it will produce the review user interface of degree thesis shown in Figure 5. School administrators can adopt or reject the degree thesis, and fill in an opinion of the thesis. The JSP codes, *xBM*s and model execution engine formulate the Foundation Management System. Developers can improve the system by modifying the *xBM* and adding more complex business logic which is separated from the above JSP codes.

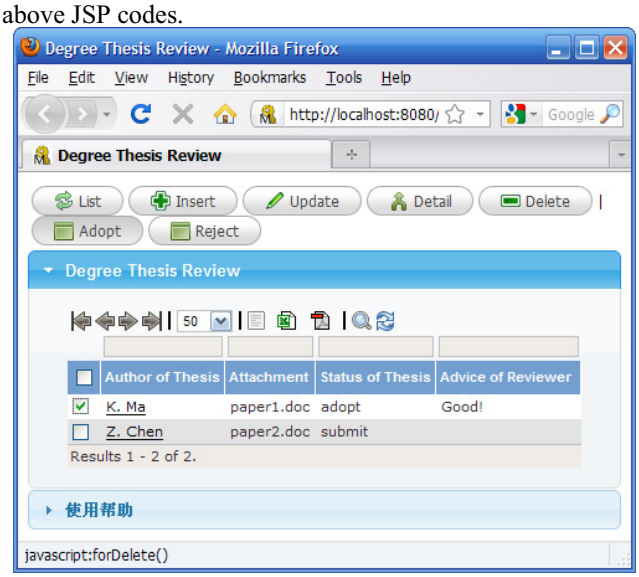

Figure 5. User Interface of Review Process of Degree Thesis by School Administrators.

VI. CONCLUSION

In this paper, a novel platform-specific executable business model called *xBM* is proposed to describe both the structural and behavioral properties of generic web applications. Its architecture of metamodel and extension mechanism is discussed in details. In addition, traditional version control idea based on *copy-modify-merge* approach is also applied in *xBM*s. Moreover, our approach for model development approach based on *xBM* is presented for generic web applications. Finally, the model-driven development process of the review module of school of simplified Provincial Excellent Academic Degree Thesis Review System is illustrated to validate this approach.

ACKNOWLEDGMENT

This work was supported by National Natural Science Foundation of China under Grant No. 60873089 and No. 60903176, Doctor Foundation of Shandong Province under Grant No. BS2009DX037, and Provincial Natural Science Foundation for Outstanding Young Scholars of Shandong under Grant No. JQ200820.

REFERENCES

[1] K. Czarnecki, S. Helsen, "Feature-based survey of model transformation approaches". *IBM Systems Journal*, vol.45, pp. 621-645, 2006.

[2] D. Frankel, *Model Driven Architecture:Applying MDA to Enterprise Computing*. Beijing: Beijing Posts and Telecom Press, 2003.

[3] X. Gao, Z. Li, " Business process modelling and analysis using UML and polychromatic sets". *Production Planning & Control*, vol.17, pp. 780-791, 2006.

[4] Object Management Group, "OMG Systems Modeling Language 1.2," http://www.omg.org/spec/SysML/1.2/, 2010.

[5] P. Feiler, B. Lewis, and S. Vestal, "The SAE Architecture Analysis & Design Language (AADL) Standard: A Basis for Model-Based Architecture-Driven Embedded Systems Engineering," *Proc. of the RTAS 2003 Workshop on Model-Driven Embedded Systems*, 2003.

[6] Object Management Group, "Platform Independent Model and Platform Specific Model for Super Distributed Object 1.1," http://www.omg.org/spec/SDO/1.1/, 2008.

[7] Java Community Process, "Java Metadata Interface (JMI) Specification 1.0," http://java.sun.com/products/jmi/, 2002.

[8] M. Proctor, "Relational declarative programming with JBoss Drools," *Proc. of the 9th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, 2008.

[9] B. Collins-Sussman, B. W. Fitzpatrick, C. M. Pilato, *Version Control with Subversion.* Sebastopol, Calif.: O'Reilly Media, Inc., pp. 4-6, 2008.

[10] K. Ma, B. Yang, "A Hybrid Model Transformation Approach Based on J2EE Platform," *Proc. of the 2nd International Workshop on Education Technology and Computer*, pp: 161-164, 2010.

[11] K. Ma, B. Yang, H. Wang, "A Formalizing Hybrid Model Transformation Approach for Collaborative System," *Proc. of 2010 14th International Conference on Computer Supported Cooperative Work in Design*, pp: 71-76, 2010.

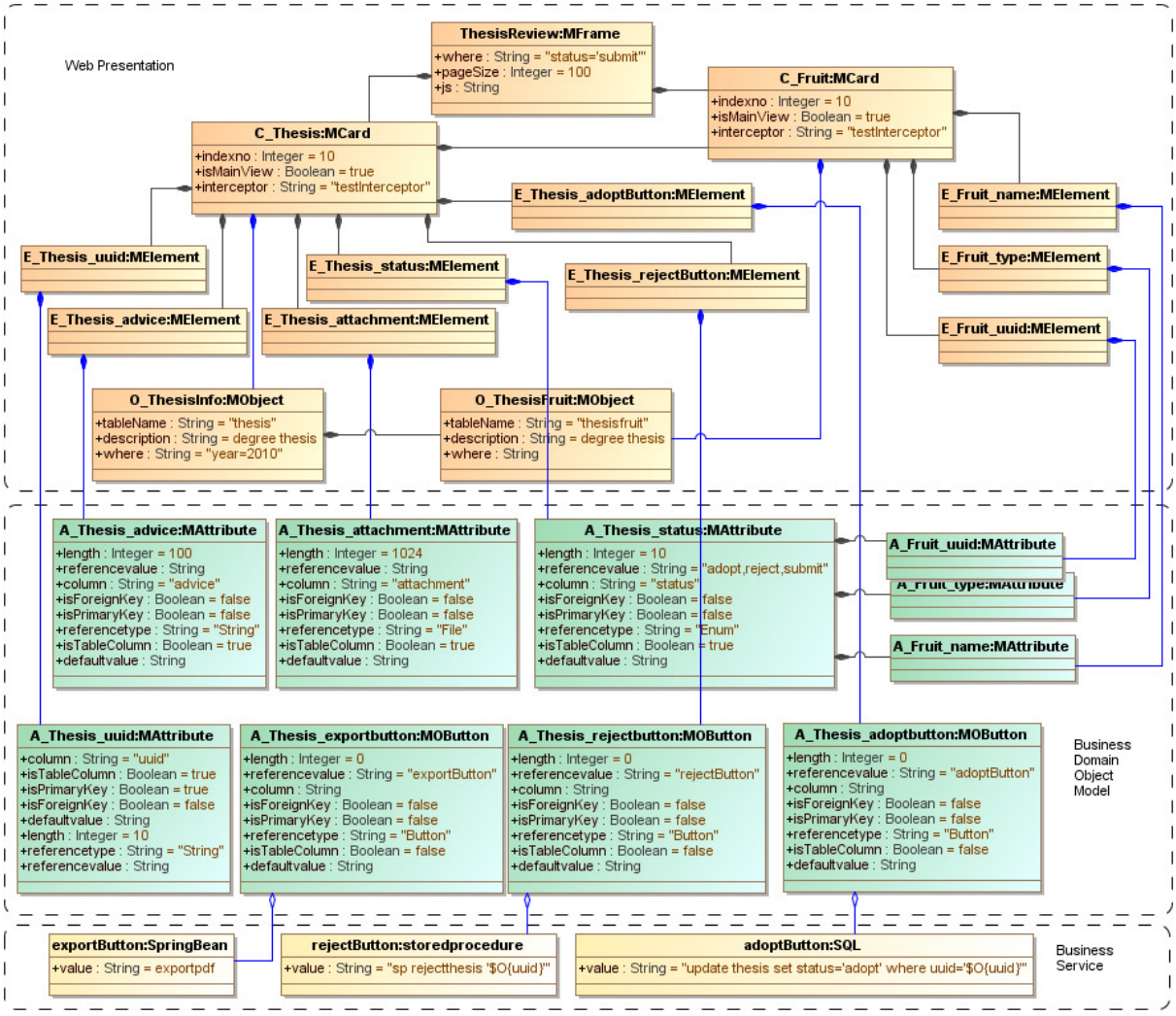

Figure 6. Executable Business Model of Review Module of School of Simplified Provincial Excellent Academic Degree Thesis Review System.