A Model-Oriented Dynamic Architecture and Its Supporting System

Yang He-biao, Zhang Jian-ming, Zhan Yong-zhao

School of Computer Science and Telecommunication Engineering, Jiangsu University, P.R.China

Summary

A dynamic architecture is proposed to facilitate model-oriented application systems' dynamical adaptation to evolving user requirements. In this approach, the architecture dominates the action of components and models with framework instance. Formalizing user requirements to executable data model, a convenient dynamic architecture mechanism for model-oriented evolution is thus naturally derived. A corresponding supporting system named Ipmis, providing a visual integrated environment to facilitate the application system development, is developed. A Hospitalization Insurance Integrated System is also implemented with Ipmis to demonstrate the effect of dynamic adaptation.

Key words:

Model; Software architecture; Dynamic evolution; Framework

1. Introduction

In the process of software development, changes of requirement are inevitable, on the contrary they are helpful in meeting the need of customers maximally [1-3]. Continual changes, however, affect the whole development efficiency, advance the difficulty level of system realization. Furthermore, they increase the cost of system implementation and lengthen the development cycle. Now that the requirement changes of customers are ineluctable, it becomes a key point in system development that how to set up an evolvable system architecture to adapt the changes of requirement.

Aiming at the problem above, this paper puts forward a kind of model-oriented dynamic development architecture. In this architecture, the system's dynamic evaluation is realized through defining the framework-unit and then generating the instances of corresponding objects. An application system can be developed in this way: firstly, take frameworks and models as the main unit of system development; secondly, abstract customer's business requirement into drivable data models; finally, bind the models to corresponding frameworks. To apply the model-oriented dynamic architecture, a corresponding supporting system is developed. It provides a visual integrated environment to integrate and run the model-oriented application system.

2. Definitions in dynamic architecture

The complexity of software evolvement results in that the research on software evolvement should be started from macroscopic level [4]. Thus the research could be avoided getting bogged down in the complicated details. The architecture in the paper is set up predominated with frameworks and data models. Following are the definitions of the related parts in a dynamic architecture.

Definition 1 Architecture S is a triple set (F, M, R), where F is framework set; M is data model set; R is the binding relation between data models and frameworks.

Definition 2 Framework F can be described with triple set (e, f, c), where e is a group of elements that constitutes a framework. The element group is classified two classes by feature: Component element and Data

object. f is the framework structure, consisting of particular characteristics and relations. Particular characteristics are used in restricting which component elements could be contained in the frame, and the relations are used in describing the topological position of component elements in the framework. c is a control set, used for coordinating the work flow of Components in the framework.

Definition 3 Component element $C=(C_c, C_s)$. C_c is the set of Component object in the framework, such as tool bar, status bar, tree component, etc; Component element is large granularity reusable units based on class library. C_s is the status set of the internal object in the component[5].

Definition 4 Data model M is a 5-tuple, M=(T,S,C,D,P).Among them, T is relational table; S is SQL syntax of the table; C is formal parameters of the SQL ,that can be null. D is data element; P is the parameter set of model .

Definition 5 Model parameter P is composed of a pair set which is {Action, Parms}. Action is the triggering behavior based on data element and Parms are parameters of the Action. Action is divided into two kinds: one is functional behavior implemented by the framework. The other is characteristic behavior, which is parsed by framework to change states of UI. Action is given with different parameter values it can complete different operational demand of the same behavior.

3. Design and realization of the supporting platform

Supporting platform's function is to set up the frameworks and describe data models, which applied the definition mentioned above. It is shown in Figure 1.

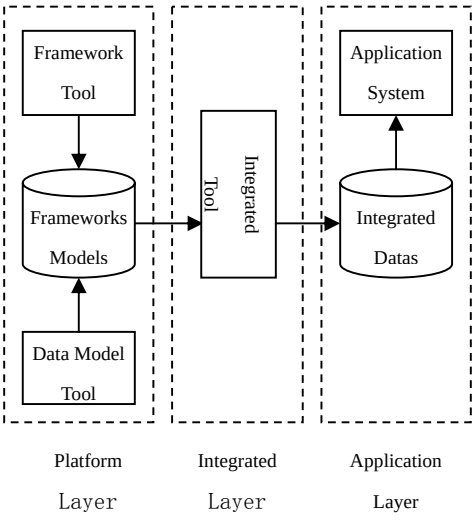

Fig 1 The architecture of supporting platform

The model-oriented dynamic architecture is divided into three layers: platform layer, integration layer and application layer. The tasks of platform layer are as follows: one is to construct the frameworks; the other is to take out domain's components-Data Models. The integration layer describes the definition of an application configuration. The application configuration defines which data models are to be included in the target application and how they are to be configured.

So we can build functional module with the application configuration. The application layer provides the run environment for functional module. The separation of framework and data model is aiming to reduce the degree of coupling between database and application code and code influence accompanying the variation of database.

Adopting the big granularity's components to build framework makes sure the system expansibility. Replacing data model to the domain's component can enhance the systemic evolvement ability. Applying the integration information to compose the application system makes it possible to reorganize application system.

The system framework classes could be abstracted from customer's functional requirements, i.e. what and how many business process patterns should be provided

to response the request of data processing. The framework can be set up in two ways: one is to load the existing framework class library and then select one; the other is to develop a framework class personally [7]. Framework class restricts which the components and split services are to be included in framework. The framework configuration is done using the property sheet. After this, we can define the control flow of components in framework to make it controllable. Finally, the configuration data should be stored in database persistently.

As data is the object processed by the software, how to access and process it become the primary problem. As for data model it is used to represent the entity of the specific problem domain. The construction process of data model is to describe the user's business, mainly including data element (or bill), behavior of data element, parameter for behavioral activity and state set of behavioral transformation. Therefore we can abstract data models set based on user's business requirement.

The business logic invoke is performed by transmitting a group of messaging events. The messaging is defined at the time of application integration. The separation between messaging events and framework means that framework will process transaction according to the messaging events .So the same framework can be used in different business processes repeatedly. As for the platform, limited to the length of this paper, only the implementation of several key technology problems relating dynamic architecture will be discussed here.

3.1 The framework design

Framework is considered the best approach, which is a reusable, largely granular software unit in the object-oriented system development [6]. In the process of framework design, it is important that how to distinguish and organize layers to meet the needs of framework's dynamic evolvement.

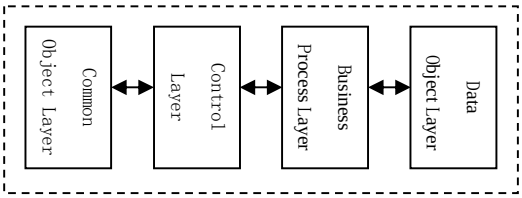

Fig 2 Framework layer structure

Figure 2 shows framework layer structure. We divide it into four layers: common object layer, control layer, business process layer and data object layer.

Common object layer is composed of visual components and non-visual service components, and constructs the fundamental functional unit of the framework. This layer focuses on application programming not on business logic. According to the application requirement of specific domain, we design common object layer. When the application requirement varies, we can modify the component of common object layer to adapt to the requirement changes.

The control layer's function is to control the executable flow of framework's functional unit. Business process layer, according to business process restriction rule, take charge of parsing and executing the behavior request of data object layer. This layer which separates from Data object layer is aiming to reduce the degree of coupling between application code and data object. Therefore, it is convenient for the frame extending.

For data object layer, its function is to parse the syntax of data model which can be set in the run-time of the system and create UI. Separating domain's businesses from the complex UI, thus it is very convenient for the system development and maintenance. In the run time of the target system, binding different data models to data object layer which can accomplish different process requirement of the same kind of business, after doing that, it is possible to make framework's dynamic evolvement. For example, a process of data collecting, the difference is only the data source and the data destination, but the process of collecting is the same.

The control layer is composed of a group of logic and instance variables. According to the defined workflow, it controls the run-time states of components and transmits the messaging among the layers with the messing mechanism.

The structure information of framework is stored in database persistently by using following relational schema.

Framework register (framework ID, framework name, framework description, window);

Framework structure (framework ID, component ID, coordinates of component);

Component register (component ID, component name, component description);

Component property (component ID, attribute)

3.2 Model Description

Data model is consisted of relation table, SQL syntax, retrieve parameter, fields and parameter sets. Data object component creates the user interface by parsing the syntax of data model as below description. At the run-time of program, providing data model for data object component, we can get input model, output model, store model etc.

We use the formalized language to define the syntax of data model, consequently it can list the attributes of element, describe relations among elements and other correlation information clearly.

Relation table can be shown in BNF pattern as follows:

<Table> :: = Des{
<Table.Field>
<Type>
<Format>
{<Attribute>}}
END <Table>

Field is data model's assistant information, which is shown in BNF pattern as follows :

<Node>:: = Des{
<Name>
<Text|Object|Parameter>
<Values>
{<Attribute>}}
END <Node>

SQL syntax is SQL retriving sentence composed by the table field.

Retrieve parameter is the condition parameter of SQL, which is shown in BNF pattern as follows :

<Retrieve> :: = Des{
<Name>
<Type>
END <Retrieve >

Model parameter set is composed of a group of fields. The value of field's attribute defines the properties of parameter,which is shown in BNF pattern as follows :

<Parameter_Property> :: = Des{
<Field> trigger field
<Event> trigger event
<Type> type of action（logic、character）
<Action> action name
<Value> value of parameter
END <Parameter_Property >

Triggering event is execution time of action, and the action name is a string. Business process layer invokes the corresponding process logic according to action name. The parameter is described with xml schema that is convenient for parsing and extending.

After defining data model, we can get a text which is consisted of four sections :relation table, syntax SQL, retrieve parameter and fields.It is stored in the database persistently with blob text.

3.3 Realization of dynamic framework

The architecture of dynamic framework is shown as figure 3. Its basic principle is to use driven framework as the executing instance, and performs business process by driving the bound data model and logic.

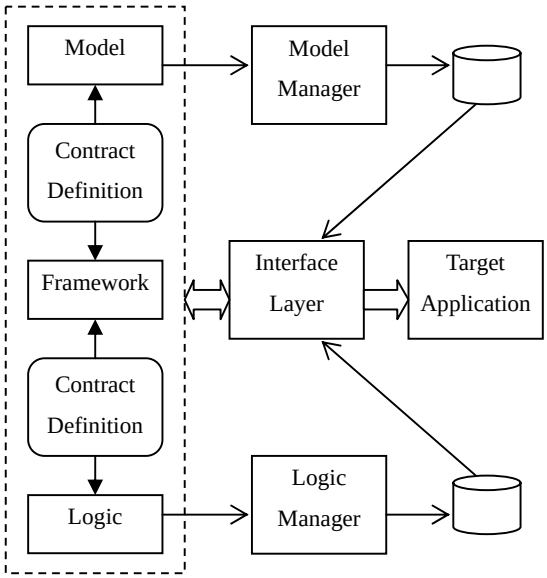

Fig 3 The architecture of dynamic framework

3.3.1 Specification definition

The specification definition among frameworks, model and logic can be described by a group of relation schemas. The main relation schema is as follows:

- Register table of module (module ID, module name, framework ID), describe correspond relation between application module and framework;
- Parameter information table of framework (framework ID, parameter ID, parameter type), it is the parameter template of framework;
- Run-time register table of framework (module ID, parameter ID, parameter name, parameter value), it describes the run-time parameter of framework

From the relation schema above, the type of parameter is classified two kinds: module and logic. Run-time register table initializes the framework and sets up the binding relation between module and logic to frame.

3.3.2 Action operation

The action of data model is considered as specific event and we offer a mechanism of event triggering to invoke action corresponding logic.

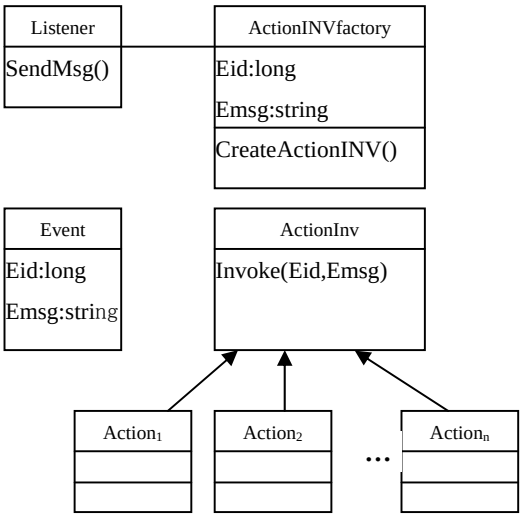

Fig 4 The factory pattern of listener and transfer

As shown in Figure 4 we adopt the factory design pattern to process the messaging transmitting and event monitoring.

- Listener is a monitor class and monitors the event of data field.
- ActionINVfactory creates interface instance of Eid.
- ActionInv is a invoking class and switches which action will be executed at run time.

Figure 4 shows the basic work principle of action operation. Listener monitors the event of data field, and transmits the messaging Eid to ActionINVfactory when some events are triggered, after this an interface instance of Eid is created by ActionINVfactory, finally ActionInv invokes proxy components to execute corresponding action of interface instance of Eid.

3.3.3 logic Invoke

Logic invoke is realized by using message mechanism that the logic itself is not defined in the component. The component carries out the logic invoke only by sending request messaging to framework, and framework search the logic register table to find corresponding logic register information, finally the application program passes it through proxy component to perform the logic invoke.

The process of logic invoke described above consists

of the following group of relational schemas.

- Messaging register (messaging ID, messaging name)
- Logic register (logic ID, logic name, description of formal parameter)
- Run-time messaging register (module ID, component ID, messaging ID)
- Run-time logic register (module ID, message ID, logic ID, invoking parameter)

Formal parameter and invoking parameter can be described with XML grammar. The corresponding relation between logic and component is built at the phase of system integration.

3.3.4 dynamic extending

The dynamic extending process of system is come true by decoupling framework and model. Different data models are bind to the same framework to perform processing the same request of different businesses. Such as activity of data collection, data sources may be various, but the process of collecting is the same. In addition, the data model is attached to different frameworks to perform different processing request of the same business. For example some work flows of business come through filling bill, auditing, keeping accounts, making voucher and so on. The data source of the business activity is the same, but the process of data transfer is different.

3.3.5 application integration

Application integration is composed of a group of interfaces and managements. We can construct the configuration information between frameworks and data models to generate the system function modules. In addition we grant the authority of executing modules to users to build relation between users and modules.

4. Conclusions and future works

To adapt application system to variation of user requirement, we present an approach to dynamic oriented-model architecture; a design and implementation of supporting platform is used to demonstrate how to build an oriented-model integration system, which implements the dynamic evolved principle mentioned above.

We claim that the distinctive feature of our approach is its simplicity and reliance on mainstream technologies. In order to substantiate our claim, the platform is applied to the development of the Integrating System of Medicare Insurance in JuRong city. As a result, the development efficiency was improved practically, and the development cycle was shortened essentially. However, to become a development technology of system architecture really, there is a lot of work to do persistently, such as how to drive characteristic action dynamically, how to control components action, etc. Our future work will follow two broad directions. On the one hand, we intend to further verify the validity of our approach by applying it to ERP system. On the other hand, we will consider to use compute model to standardize the control of component action in framework and use work-flow engine to drive data model's characteristic behavior dynamically.

Reference

[1] Matthias Weber , Joachim Weisbrod. Requirements Engineering in Automotive Development: Experiences and Challenges[J] . IEEE Software , 2003 ,120(1) :162-241

[2] Saffena Ramzan, Naveed Ikram. Making Decision in Requirement Change Management Information and Communication Technologies, 2005.ICIT 2005.First international Conference on 27-28 Aug. 2005:309-312.

[3] Lian Wen, R.Geoff. Dromey.From Requirements Change to Design Change: A Formal Path .Software Engineering and Formal Methods, SEFM 2004.Proceeding of the Second international Conference. Sept.2004:104-113.

[4] Wang Yinghui,Liu Yu,et al. SA Dynamic Evolution

Model Based on Static Point Transition. Chinese Journal of Computers, 2004,27(11) : 1451-1456.

[5] Chen Wenzhi, Xie Cheng ,Shi Jiaoying Component-based and module driven embedded operation system[J]. Journal of Zhejiang university (Engineering Science), 2005, 39(9) : 1348-1352.

[6] Sun Changai, Jin Maozhong, Liu Chao. Overviews on Software Architecture Research[J]. Journal of Software, 2003,13(7): 1228-1237.

[7] Ma Xiaoxing,YuPing,TaoXianping,Lu Jian.A Service Orienter Dynamic Coordination Architecture and Its Supporting System[J]. Chinese Journal of Computers, 2005, 28(4) : 467-477.

[8] Yang Hebiao,Ding Yong,Du Jiang.Model Design of domain oriented class library based on triple commune architecture[J].Journal of Jiangsu University(Natural Science Edition), 2005, 26(6):521-524.

[9] Chen Wenzhi,Xie Cheng,Shi Jiaoying Component-based and module driven embedded operation system. Journal of Zhejiang university (Engineering Science), 2005,39(9): 1348~1352