Formal Modeling for Information Appliance Using Abstract MVC Architecture

Arichika, Yuji
Department of Computer Science and Communication Engineering, Faculty of Information Science and Electrical Engineering, Kyushu University

Araki, Keijiro
Department of Computer Science and Communication Engineering, Faculty of Information Science and Electrical Engineering, Kyushu University

出版情報：The Proceedings of ISFST-2004, pp.180-185, 2004-10-21. Software Engineers Association
バージョン：
権利関係：

Formal Modeling for Information Appliance Using Abstract MVC Architecture

Yuji Arichika
Kyushu University
6-10-1 Hakozaki, Higashi-ku
Fukuoka 812-8581, JAPAN
arichika@ale.csce.kyushu-u.ac.jp

Keijiro Araki
Kyushu University
6-10-1 Hakozaki, Higashi-ku
Fukuoka 812-8581, JAPAN
araki@csce.kyushu-u.ac.jp

ABSTRACT
In information appliance development, it is important to divide core functions and display functions because information appliance have various user interface and display functions changed frequently. Using MVC architecture is one way to divide display functions and core functions. But MVC architecture is implementation architecture and there are some gaps to get abstract model. On the other hand it is known that formal methods are useful for constructing abstract model. Therefore we intend to use formal methods and MVC architecture to get abstract model which divide core functions and display functions. In this paper we propose abstract MVC architecture which abstracted traditional MVC architecture. And we introduce how to describe abstract model of information appliance using formal methods and abstract MVC architecture using cork board system as a case study. After that we discuss the benefit of our modeling approach and the guide line for modeling display functions.

Keywords
Formal methods, MVC architecture, information appliance, modeling

INTRODUCTION

In software development, most serious errors are generated in upstream software development and it need huge cost to correct these error as compared with the error in downstream development. It is known that for reducing the error in upstream software development, formal methods are useful. In the formal methods, we construct abstract system model, and check the feature of the system by using that before we implement software. Therefore, it is important to construct abstract model in order to develop software effectively. But there is no guide line for constructing abstract model in the formal methods and it is difficult to build proper model.

On the other hand, with the spread of the Internet in recent year, commercial production of an information appliance is progressing rapidly. Information appliance is an embedded system and is consumer device which can connect with other system through network. In information appliance development, because information appliance has various user interfaces, specifications of the display functions are changed frequently for requirements of a users and changes of hardware. It is important to get abstract model which specify essential feature of display functions and divide display functions and core functions.

To get software divided display functions and core functions, one of the popular architecture is MVC architecture. But traditional MVC architecture was made for dividing display functions and core functions in design and implementation stage of the software development. So when we use MVC architecture to describe the model, the model will be too detailed and complex. Therefore we intend to use both formal methods and MVC architecture to get abstract model which divide core functions and display functions. We intend to use formal method to construct abstract model and MVC architecture to divide core functions and display functions.

The purpose of our research is to get case study of formal model of information appliance and guide line for constructing abstract model of information appliance. In this paper we introduce how to describe abstract model of information appliance which specify essential features of display functions and divide display functions and core functions using formal methods and MVC architecture. We use cork board system as a case study. After that we discuss the benefit of our approach and the guide line for modeling display functions

FORMAL METHODS AND MVC ARCHITECTURE

Formal Methods

Formal methods are methods of software development. Formal methods provide formal specification languages which based on mathematics and we can describe strict model. And formal methods provide the way to describe abstract model which specify essential property of the systems. By describing abstract system model, we can analyze essential property of the systems. But formal methods themselves don't provide how to pick out the essential property of the systems and there is no framework and guideline for describing system model.

In this paper we intend to describe and analyze abstract model using formal methods and present guidelines to describe abstract model which divide display functions and core functions. There are a lot of formal specification languages, and popular languages are VDM-SL[1], Z[2]. In this paper we

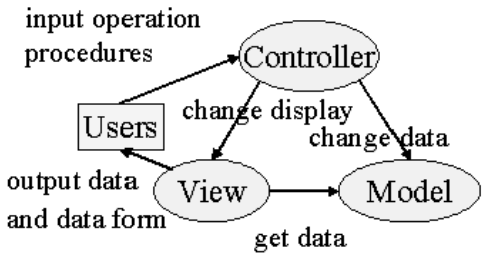

Figure1:MVC architecture

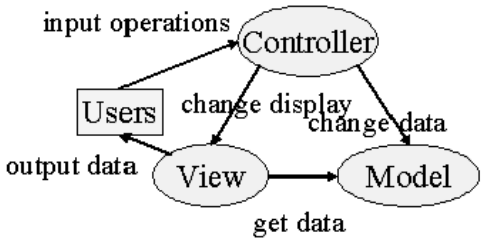

Figure2:Abstract MVC architecture

use VDM-SL as a formal specification language.

MVC Architecture

MVC architecture is architecture for object-oriented system development[3]. It divides the system into three parts. *Figure1* shows MVC architecture. Three parts are model, view and controller. Model part is a part which expresses core functions of the systems and view part express output functions, and controller part express input functions. MVC architecture provides a guide line to divide display functions and core functions. But MVC architecture is made for design and implementation stage of the software development, and there are some gaps when we use MVC architecture to describe abstract model which divide core functions and display functions and constructed model will be too detailed and complicated. In this paper we intend to use MVC architecture to divide core functions and display functions and address this issue.

FORMAL MODELING OF INFORMATION APPLIANCE USING ABSTRACT MVC ARCHITECTURE

In this section we present how to describe abstract model which specify essential features of display functions and divide core functions and display functions using formal methods and MVC architecture.

First we introduce overview of Formal modeling using abstract MVC architecture. Then we use cork board system as a case study of the information appliance, so we introduce cork board system. And next we show how to describe specification of the cork board system. And last we show the extendability evaluation of the specification to check that the specification divides the core functions and display functions and make sure that our modeling approach make abstract divided model of information appliance.

Overview of Formal Modeling Using MVC Architecture

To get abstract model which specify essential features of display functions of the information appliance, we have to consider essential features of display functions. Display functions show the user required information on the display by following the user operation, so we consider that the relationships between user operations and what data the system display are essential features of the display functions. Therefore in this paper, we construct system model which specify the relationships between user operation and what data the system display as an abstract model which show the essential feature of the display functions model.

To get model which divide display functions and core functions, we use MVC architecture. MVC architecture was made for using design and implementation stage of the software development and there are a lot of case examples using MVC architecture in design and implementation stage of the software development, but there are few case examples using MVC architecture in specification definition stage of the software development and there are some gap traditional MVC architecture for describing abstract model. Therefore we make abstract MVC architecture for describing abstract model. *Figure2* shows our abstract MVC architecture. Overview of each part of our abstract MVC architecture is as follows.

model This part specifies what data the system treats and how to treat these data

view This part specifies what data the system displays

controller This part specifies what operation the system provides for users

About model part, traditional MVC architecture specify core functions which means what data the system treat and how to treat these data. As mention above, we construct model which specify the relationships between user operation and what data the system display, so we specify almost same things of the tradition MVC architecture in the abstract MVC architecture.

About view part, traditional MVC architecture specify what data system display and display forms of these data. But display forms of data are not necessary for specifying relationships of between user operation and what data the system display. Therefore we specify only what data the system display in view part of abstract MVC architecture.

About controller part, traditional MVC architecture specifies how to input the operation. But we want to specify what operation the system provide and don't need how to input these operation for specifying relationships of between user operation and what data the system display. So we specify only

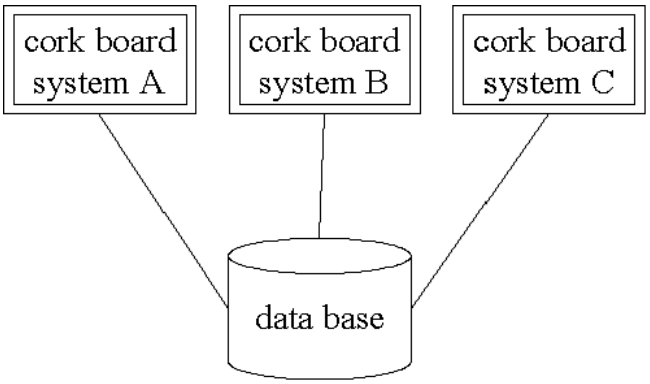

Figure3:Relationship between cork board systems and database

Figure4:Three type of cork board systems

what operation the system provides for users in controller part of abstract MVC architecture.

Next we show how to describe abstract model of information appliance using our abstract MVC architecture by using cork board system as a case study.

Cork Board System

In order to construct abstract model of information appliance, we use cork board system as a case study. Cork board system is an information sharing tools. Cork board system has display and users can write and read the various information on the display. Examples of information on the corkboard are members' location data, message and picture. The Cork board system connects other cork board system through network and share information. *Figure3* shows relationship between cork board systems and database. For this features, we can regard cork board system as an information appliance.

Characteristic of cork board system is that cork board system has various user interfaces, such as fixed type, web type and mobile type(show *Figure4*). One example of differences of each user interface type is that fixed type of cork board system has large display, so this type can show a lot of data in the same time, on the other hand, mobile type has small display because this type of cork board system is required to be compact size, and this type of cork board system can show only a few data in the same time. If we get abstract model which divide core functions and display functions of cork board system then we can reuse core functions part to develop various display functions of cork board system. Therefore it is important to get essential feature of the display functions of each three cork board system and to divide core functions part and the display functions in order to develop cork board systems effectively.

Formal Specification of Cork Board System

In this subsection, we show how to describe abstract system model using abstract MVC architecture by using cork board system as a case study. Cork board system has various user interfaces as mention above, and we describe model of fixed type cork board system.

Model Part In model part, we describe only what data the system treats and how to treat these data. For example, the fixed type cork board system has functions to treat member' location data. About this function, users can register them name, them location and them group, and users appoint group name then they can get all members' name and location data who are member of the appointed group. We specify these functions in model part in VDM-SL as follows.

```
types
Model::corkboard:CorkBoard
       database:DataBase;

CorkBoard::members: map MemberID to Member
           messages: map MessageID to Message
           picture: map Coordinate to Color;

Member:: name: Name
         location: Location
         groups: set of Group;
Name = token;
Location = token;
Group = token;

DataBase::members : map MemberID to Member
          messages: map MessageID to Message;
```

First we specify what data the system treats. Type `Model` means model part of fixed type cork board system and type `CorkBoard` means fixed type cork board system and type `DataBase` means data base which connect with fixed type cork board system through network. Field `members` means registered members' data. Type `Member` means member's data registered in the system. This specification show that all member has unique identifier (`MemberID`) and each member has a name (`name`)and a location data (`location`) and belong some groups(`groups`).

Next we specify how to treat these data. As an example to specify how to treat the data, we show the specification of the function to show appointed group members' name and location.

```
functions
```

```
MGetLocation:Model * Group ->
             map MemberID to (Name*Location)
MGetLocation(m,g)==
  {mid |->
    mk_(m.corkboard.members(mid).name,
        m.corkboard.members(mid).location)
   |mid in set dom m.corkboard.members &
    g in set m.corkboard.members(mid).groups}
pre g in set dunion
    {m.corkboard.members(mid).groups
      | mid in set dom m.corkboard.members};
```

By specifying model part, we can analyze what data the system treat and how to treat these data.

View Part In view part, we describe what data the system displays. We describe view part of fixed type cork board system as follows.

```
types
View::display_board:Board
      nameplate:VNameplateBoard
      message:VMessageBoard
      white:VWhiteBoard;

Board = <Nameplate>|<Message>|<White>;

VMessageBoard::display_page:nat1
               pages:seq1 of MessagePage
inv mk_VMessageBoard(d,p) == d in set inds p;

MessagePage = map MessageID to Message
inv mp == card dom mp <= MessageSize;
```

Fixed type cork board system has three views which are members' location board view(`nameplate`) and message board view (`message`)and white board view(`white`). And fixed type cork board system can display only one view in the same time, so this feature specified by field `display_board` of type `View`.

Display of fixed type cork board system has fixed scale and a number of the data which fixed type cork board system can display are constant and if the number of the data which fixed type cork board system have to display is over the limit, then the system divide data on the several pages. For example, fixed cork board system has functions to display registered message and the limit number that the system can display message in the same time, so the number of the registered message is over the limit, then message board view divides messages into several pages. *Figure5* shows this function. In VDM-SL, type `VMessageBoard` means the view of the message board and it consist of several message page view and what page the system display. We decide field `display_page` means what page the system displays in the case that the system display message board view. For example, in the case that the value of field `display_page` of

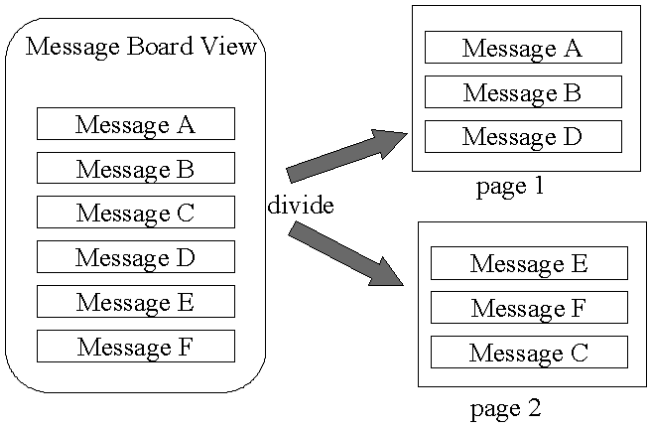

Figure5:Functions to divide messages into several pages

type `VMessageBoard` is 1 and the value of field `display_board` of type `View` is `<Message>` then the system display first page of message page view. But if the field `display_board` of type `View` is not `<Message>` then the value of field `VMessageBoard` means nothing. We describe that the number of the message on each message page view should not over the limit as an invariant of `MessagePage`. And we describe that message board view should display existing page by using type invariant of `VMessageBoard`.

The functions related to the view are function to change displayed view and function to update displayed data by using data of model part. The example of specifications in VDM-SL is as follows. This function is to change displayed view to assigned view. When the system changes the displayed view or update displayed data, the case will be happen that the displayed view is not existing view. So we have to guarantee that displayed view is existing view after using these functions. This condition to guarantee that is described by type invariant of the each view as mention above. Therefore we can guarantee that this condition is kept by checking each functions of the view keep type invariant of the view using pre condition description.

```
functions
VChangeDisplay:View * Board * [Group] * [nat]
               -> View
VChangeDisplay(v,b,g,n)==
  cases b:
    <Nameplate> ->
      VChangeNBPage(VChangeGroup
                    (VChangeBoard(v,b),g),g,n),
    <Message> ->
      VChangeMBPage(VChangeBoard(v,b),n),
    <White> ->
      VChangeBoard(v,b)
  end
pre (b = <Nameplate> =>
       g in set dom v.nameplate.groups and
```

```
      n in set inds
           v.nameplate.groups(g).pages) and
    (b = <Message> =>
      n in set inds v.message.pages);
```

Controller Part In controller part, we specify what operation the system provides for users. For example of fixed type cork board system, we describe there is change command for user in this system as follows.

```
types
Controller::model:Model
           view:View;

functions
CChangeMemberLocation:
   Controller * MemberID * Location
                    -> Controller
CChangeMemberLocation(c,mid,l) ==
  let new_m =
      MChangeMemberLocation(c.model,mid,l) in
    mk_Controller(new_m,
      VUpdateDisplay(c.view, new_m))
pre pre_MChangeMemberLocation(c.model,mid,l);
```

Evaluation of Extendability

Above subsections, we describe how to describe abstract model of the system using MVC architecture. This subsection we evaluate the extendability of this model in order to evaluate that this model divide core functions and display functions. The evaluation procedures are as follows.

1. to pick out features of three type of cork board system.
2. to change the model by using pick out features.
3. to analyze whether we have to change each MVC parts of specification or not.

We use feature modeling which is a technique of the product line software engineering[4] and *Table1*, *Table2* shows the result of the evaluation. We classify the feature of three types of cork board systems by the name of feature. We classify that those feature that have the word which concern with display in the name are display features and other features are core features. These table shows that when we construct new model from fixed type cork board system model which we introduced previous subsection, whether we have to change each MVC parts of specification or not. The result shows that we can change model almost display features by changing view part of the system without changing model part. About core features we can change all core features by changing only model part without changing view part. By this evaluation result, we conclude that the model which is generated by using our abstract MVC architecture divides core functions and display functions and has extendability.

Changing display features	Model	View	Controller
Adding location depend display	Need	Need	Not
Adding caution displaying	Need	Need	Need
Deleting message displaying	Not	Need	Not
Deleting white board displaying	Not	Need	Not
Changing large display functions	Not	Need	Not
Changing page relation functions	Not	Need	Need
Change display size	Not	Need	Not

*Table*1 : Evaluation of extendability about display features

Changing core features	Model	View	Controller
Adding message registering	Need	Not	Need
Adding message deleting	Need	Not	Need
Adding members Managing	Need	Not	Need
Adding groups Managing	Need	Not	Need
Adding location managing	Need	Not	Need
Deleting white board writing	Not	Not	Need
Deleting white board erasing	Not	Not	Need
Changing network functions	Need	Not	Need

*Table*2 : Evaluation of extendability about core features

DISCUSSION

Benefit of Our Modeling Approach

In previous section, we introduce how to describe abstract model which specify essential features of the display functions and divide core functions and display functions of information appliance using MVC architecture and formal methods. So we discuss and enumerate the benefit to describe the abstract model using MVC architecture and Formal methods.

First, we can specify relationships of the user operations and what data the system display strictly for formal methods. In the case of cork board system, cork board system should keep data consistency between data base, and the display screen should not display the view not existing. About data consistency problem, we can specify and check in the formal model in previous work[5]. About display screen problem, we can specify conditions in the formal model as type invariant and pre condition of functions and check that these conditions are kept in the model as mention in section 3.

Second, we can reuse abstract model as an intermediate product of software development. Because the model is abstract and consists of three parts. About abstraction, the model specify only relationships of the user operations and what data the system display and does not specify the procedure for users operating the system and the display form of data. So we can reuse the model to develop various systems which are different from the procedure for users operating the system or the display form of data. About construction, the model consists of three parts, model part, view part and con-

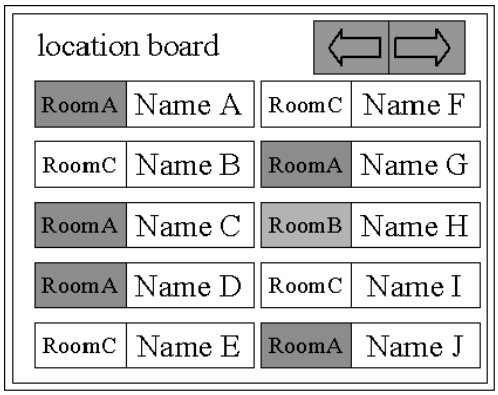

Figure6:Members' location data view

troller part. We can get new model of other system easily, because we can get it by changing only particular parts of the model.

Guide Line for Modeling Display Functions

In this paper we describe formal model for analyzing essential feature of display functions. In formal methods, there is no guide line for describing abstract model and it is difficult to describe abstract model. Therefore we discuss some guide line for describing abstract model of display functions.

About display functions of the systems, there is a lot of information. On the display, there are a lot of objects and each object has a lot of information. For example *Figure6* show one cork board system view which show the members' location data. This view has not only member's location data object, but also user command button object. And member's location data object has not only information of member's location data but also size and color information. When we describe all of this information, the model will be too complex and detailed. So we pick out the essential information in order to describe abstract model. We consider what the essential feature of the display functions is. As an answer of this, we think that what required data the system display on the screen is an essential feature of the display functions of the system.

There are a lot of object on the screen of the system such as data, menu list, button. We classify this object into two groups. One is an object which has data required by users such as data and the other is an object which provides input means such as menu list and button. And we consider we should describe only the former object in the abstract model. In the case of the cork board system, message view of the cork board system has two objects. *Figure6* shows the message view. Message view has messages and view change button. We consider the essential functions of the message view are to show the messages and not show the view change button. So we describe only messages in the model and don't describe view change button.

And objects on the screen of the system have a lot of information such as meaning, size and color. We consider that to describe abstract model we should only meaning of the object and do not describe other information such as size and color of image. In the case of the cork board system we describe only what message the system display and don't describe the size of the message image and color of the message image.

CONCLUSION

In this paper, we introduced abstract MVC architecture and how to describe abstract model which specify essential features of display functions and divide core functions and display functions of the information appliance using abstract MVC architecture and Formal methods. As a case study of information appliance, we used cork board system. We introduced how to describe abstract model of cork board system and evaluate the extendibility of the model. By using our modeling guide line, we expect that we can analyze core functions and display functions of the system easily and we can develop information appliance effectively.

In this paper, although we address to the variety display feature of information appliance, there are other essential features in information appliance. For example they provide various services by collaborating with other system through network and they have feature to be ubiquitous for the network. In the Future, we have to study how to model these feature of information appliance using formal methods.

ACKNOWLEDGEMENTS

We thank associate professor Kusakabe for providing useful comments for this paper and professor Fukuda and associate professor Nakanishi for providing cork board system and advising about system modeling.

REFERENCES

1. The VDM-SL Tool Group, "The IFAD VDM-SL Language",Technical Report IFAD-VDM-1, The Institute of Applied Computer Science, 1996.
2. J.M.Spivey,"The Z Notation: A Reference Manual Second Edition" Prentice Hall,1992
3. A. Hussey and D. Carrington, Comparing the MVC and PAC architectures: a formal perspective.,IEE Proceedings of Software Engineering, 144(4):pp224-236, 1997
4. Kyo C. Kang, Sajoong Kim, Jaejoo Lee, Kijoo Kim, Euiseob Shin and Moonhang Huh, FORM:A feature-oriented reuse method with domain-specific reference architectures, Annals of Software Engineering 5 pp143-168,1998
5. Yuji Arhichia and Keijiro Araki, Formal Specification and modeling for a ubiquitous information sharing tool, Proceedings of International Symposium on Information Science and Electrical Engineering, pp625-628,2003