# CONCEPTUAL MODELING FOR WEB APPLICATIONS AND DEFINITIONS OF BUSINESS LOGIC FOR END-USER-INITIATIVE DEVELOPMENT

Takeshi Chusho and Jing Li
*Department of Computer Science, School of Science and Technology, Meiji University*
*Kawasaki, 214-0033, Japan*

**ABSTRACT**

The development of Web applications should be supported by business professionals themselves since Web applications must be modified frequently based on their needs. In our recent studies with the three-tier architecture of the user interface, business logic and database, the construction of the graphical user interface and the simple database system was supported by using a domain-specific application framework and visual modeling technologies. The conceptual model is based on CRUD (create, read, update, and delete) operations. As for the business logic, however, it is rather difficult to support it by the same method because there are various kinds of business logic. This paper describes the ABC model: application = business logic + CRUD. This approach is applied to Web applications in practical use with case studies. Finally it is confirmed that the development process based on the ABC model is useful for an end-user-initiative approach.

**KEYWORDS**

Conceptual model, Web application, End-user computing, Business logic, CRUD.

## 1. INTRODUCTION

The number of Web applications which end-users have access to has been increasing. Most of these applications are developed by IT professionals. Thus, attempting to achieve automation is limited to particular tasks which calculate profit over the development cost. Furthermore, it is difficult to develop applications quickly. Primarily, Web applications should be supported by business professionals themselves since Web applications must be modified frequently based on users' needs. Therefore, end-user-initiative development has become important for the automation of end-users' fulfilling their own needs [7].

There are several approaches for end-user-initiative development. The UI-driven approach makes it possible to easily develop applications for the UI-centered front-end systems. It is strengthened by using domain-specific framework technologies. The model-driven approach makes it possible to easily develop applications for workflow-centered back-end systems. It is strengthened by using a visual modeling tool. Some paper described a summary of the trends of end-user development without IT professionals' assistance [14]. End-user software engineering research for end-user programmers and domain experts also appeared [6].

There are some other related works. In the programming field, the technologies for programming by example (PBE) [9] were studied. PBE implies that some operations are automated after a user's intention is inferred from examples of operations. Non-programming styles for various users, including children, and for various domains, including games, were proposed.

Since our research target was designed for business professionals and business domains, it was different from these technologies. The user's intention is defined as requirement specification without inference as business professionals with domain expertise develop software which executes their own tasks.

Therefore, this paper focuses on a Web application in which the user interface is a Web browser because most users are familiar with how to use the Internet. Furthermore, the three-tier architecture which has become popular recently, is supposed. Generally, there are three approaches corresponding to the user interface (UI), business logic (BL) and database (DB). In our studies, domain-specific application frameworks and visual modeling tools based on components were developed for an end-user-initiative approach. They support the construction of the graphical user interface and the simple database system [2]. The conceptual model is based on CRUD (create, read, update, and delete) operations.

As for the business logic, however, it is rather difficult to support it by the same method because there are various kinds of business logic. Therefore, for end-user-initiative development, the business logic should be expressed from the view of the service providers or the support systems instead of the view of the clients. It was confirmed that the template based on the ABC model (application = business logic + CRUD) was useful for requirement specification of business logic.

This paper presents basic approaches for Web application development in Section 2, the modeling of Web applications for case studies in Section 3, the definition of business logic at the requirement specification level in Section 4 and implementation techniques in Section 5.

## 2. BASIC APPROACHES FOR WEB APPLICATION DEVELOPMENT

Our approach to Web application development is shown in Figure 1. The business model at the business level is proposed by those end-users who are business professionals and domain experts. Then, at the service level, the domain model is constructed and the required services are specified. At the software level, the domain model is implemented by using components. In this approach, the granularity gap between components and the domain model is bridged by business objects [12], patterns and application frameworks [5]. The semantic gap between the domain model and end-users is bridged by domain-specific technologies [13].

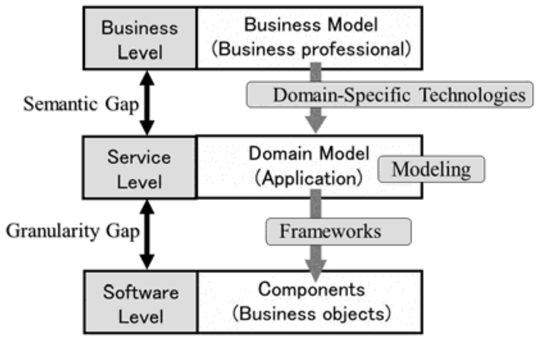

Business Level
Business Model (Business professional)
Semantic Gap
Domain-Specific Technologies
Service Level
Domain Model (Application)
Modeling
Granularity Gap
Frameworks
Software Level
Components (Business objects)

Figure 1. Technologies for end-user-initiative development

The approaches to end-user-initiative Web application development methodologies based on the three-tier architecture are classified into the three categories of UI-driven, model-driven and data-driven processes by first focusing on either the UI (user interface), the model (business logic) or DB.

Recently, a UI-driven approach has emerged as Web applications are increasing. In our UI-driven approach [4], the forms were defined first and the framework was used. The business logic depending on the application was defined by the form definitions. The other business logic was embedded into the framework. However, this framework did not support the back-end system with the workflow and DB.

One solution for workflow-centered back-end systems is the model-driven approach. Around the 90's, object-oriented analysis and design (OOAD) technologies came out and have become the major methodologies. The unified modeling language (UML) is used for definitions of the system model. In addition, UML2.0 requires more rigorous definitions of models for automatic generation of program codes based on the model-driven architecture (MDA) [10, 11]. The initial model is described as a platform-independent model (PIM) and transformed into some platform-specific model (PSM).

Our approach is different from the MDA-based approach. For end-user-initiative development, our model-driven approach is based on component based software engineering (CBSE) with a formula of “a domain model = a computation model.” This formula implies that one task in a domain model of cooperative work corresponds to one object in object-oriented model. Therefore, it is not necessary for end-users to convert a domain model into a computation model with application architecture. The domain model is considered as the requirement specifications.

End-users can get application software by visual modeling which defines forms and form-to-form transformation. A Web application model which is defined by end-users is finally transformed into Java codes of the Web application. One of main problems for end-user-initiative development is how to describe business logic. In our past studies, some scripting languages and rules were attempted. However, in these methods, end-users were required to learn some programming concepts. Therefore, tile programming was adopted for the visual modeling tool [3]. The system prepared some tile templates for instruction statements. End-users construct the business logic by combining these templates. However, it may be difficult to prepare a sufficient set of tile templates. Further studies are needed for reinforcements of the modeling for easy description of business logic.

## 3. MODELING OF WEB APPLICATIONS FOR CASE STUDIES

### 3.1 Domains of Web applications

In our studies for end-user-initiative development, the following application domains were selected:

* lending books
* lending equipment
* reservation of meeting rooms
* reuse of second-hand articles

All of them required at least two DB tables for services. One is for member registration and the other is for object registration. The member implies a system user, including a system administrator. The object is a book, a piece of equipment, a room or an article. The basic operations are CRUD (create, read, update and delete). Although columns of a record are dependent on an object, these differences are unified by the concept of “matching” between an object provider and an object requester. Figure 2 shows the entity-relationship diagram for basic object management. In the model, the object provider and the object requester are discriminated by the attributes of the member. Generally, there will be some variations. Furthermore, Figure 3 shows the basic behavior in the use cases of UML. The update and deletion operations among CRUD operations are omitted in the ERD since these operations are included in the registration relationship.

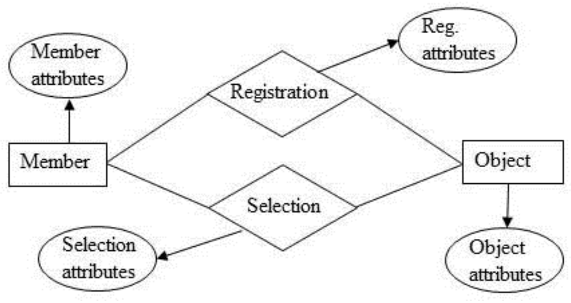


Figure 2. The data model in ERD

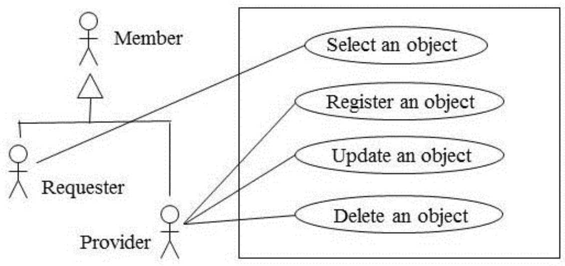


Figure 3. The model in the use case diagram

The practical naming of the selection may be different in each application. The lending of books or equipment, reservation of meeting rooms and request for disused articles may be employed. Furthermore, business logic must be very different in each application. Especially, within the first three applications, a matching concept is a little complicated because a use period is added into factors for matching in addition to a member and an object. For example, someone borrows a book for a week, borrows some equipment for one day or reserves a meeting room on the morning of the following Monday. On the other hand, in the fourth application, a matching concept is rather simple because the matching is limited between a member and an object with no consideration about the use period. Someone requests some article for reuse. Therefore, we studied the ABC model with the case of a reuse promotion service in the remainder of this paper.

## 3.2 Reuse promotion services

The reuse support system is the preferable application for our study. This is because it is expected that information technology (IT) contributes to saving resources and environmental preservation for a sustainable society. For this purpose, application software is required, and funds are needed for its development by IT professionals. However, the preparation of funds is difficult unless a profit is calculated over the development cost. The end-user-initiative development of application software is indispensable for the solution of this dilemma. For example, let’s consider a charity shop or a thrift store which sells limited goods to limited customers in a local area. The number of goods and the number of customers will increase if business professionals develop the application for the web site in which customers can easily register goods to be reused or search the list of registered goods for their own use.

Actually, when I visited Stockholm in 2012, I found the charity shop beside the Nobel Museum in which a lot of second-hand articles were sold. The staff told me that homeless people were supported with the profits. The Web site must be desirable for such social welfare promotion. Therefore, the reuse support system for a charity shop or a thrift store is considered as a typical Web application for end-user-initiative development.

## 3.3 Survey on reuse promotion services

First, actual support systems for the promotion of reusing second-hand items were surveyed by searching Web sites. As a result, the following facts were confirmed:

- Many local governments in Japan support reuse promotion activities for ecological movements. Most of them use Web sites for announcements of the activities, but do not use it for practical operations. Instead practical operations are executed at the actual counters.
- There are a lot of regulations for reuse promotion services and the regulations are strongly dependent on each local government’s policy.

We introduce some cases of big cities in Japan. In Kawasaki city where our faculty exists, two kinds of ecological supports were found. One is a reuse promotion event in which a citizen can get free and available goods which the city office has collected from places of disposal. The other is an open-air market in which a citizen can sell unwanted goods to other citizens. Both involve face-to-face dealings, although the announcement is performed via the Internet. Furthermore, there are some rules and regulations. In the reuse promotion event, a receiver must be more than seventeen years old and can gain less than four items at one event. In the open-air market, only citizens can participate in the event and goods to be sold are limited to unwanted household items. In some wards of the Metropolis of Tokyo and Osaka city, similar events are supported.

This survey on the present status of reuse promotion services confirmed some facts. First, most reuse promotion services employ the Internet for announcements but do not use it for the actual dealing of second-hand items. If the Internet is used, the number of donors and donees can increase and the effectiveness of the reuse promotion service can be drastically improved. Secondly, almost all the reuse promotion services are provided under detailed rules and regulations. Since these rules and regulations are dependent on the policy of each office, it is necessary that the general application system for reuse promotion services can be customized.

## 3.4 Matching model based on the ABC model

The reuse promotion service is a kind of a matching service between a service requester and a service provider. The similar sites are found in the Internet. There are many auction sites such as eBay, Yahoo, Rakuten etc. for buyers and sellers of used items. Some free market sites for smart phones have come out. Some shopping sites support young females selling and buying second-hand clothing. These sites require users to install the smart phone applications for easy operations. For example, a user takes some pictures of a used jacket and uploads them with some features and the price. The recent crowd sourcing for job matching may also be a kind of matching site.

In this paper, we focus our attention on the reuse promotion service only for free goods because the business logic for the reuse of free goods is limited by simple business rules in comparison with applications for auction sites or free market sites. In this domain, the case study is performed for evaluation of the ABC model: application = business logic + CRUD.

In the reuse promotion service, registered goods are considered as objects and donors and donees are considered as members, and then the data model is almost the same as Figure 2 except for replacing the word “selection” with “request.” The basic functions are CRUD-based and the use case diagram is almost the same as Figure 3 except for replacing the words “Select,” “Provider” and “Requester” with “Request,” “Donor” and “Donee” respectively. This model may be considered as the common matching model among various Web applications. Sometimes, a manager for a reuse promotion service is necessary for checking business rules.

## 4. BUSINESS LOGIC DEFINITIONS AS REQUIREMENTS

### 4.1 The UtoU template for business logic definition

The analysis of business logic is indispensable to support that end-users describe the various kinds of business logic. Therefore, first, a lot of business logic was gathered by a survey on the Internet and classified into several categories in our previous work [1]. The following five types of business rules [15] were adopted since it seemed general and reasonable for the case studies: {Fact, Constraints, Action Enablers, Inferences, Computations}. In this classification, there were some problems. For example, many rules were represented from the view of service clients. For the software requirement specifications (SRS), however, rules are represented from the view of the service providers or the support system. These case studies confirmed that the classified category of each rule varies by the expression of the rule which depends on the view of a client or the system.

In this section, business logic is defined from the view of the service providers or the support system. Furthermore, it is supposed that a Web application has the typical three-tier architecture of user interfaces (UI), business logic (BL) and database (DB) as shown in the upper part of Figure 4.

Next, the business logic from the view of the support system at the requirement specifications level is mapped into the combination of UI, BL and DB. The following template is introduced because the UI-driven approach is suitable for the end-user-initiative development:

(1) UI: The system gets a request from a client.
(2) BL: The system processes the request.
(3) DB: The system accesses the database.
(4) BL: The system processes the results.
(5) UI: The system displays the results.

This template, named UtoU, implies that the typical process for business logic is {UI > BL > DB > BL > UI} as shown in the lower part of Figure 4. It is easy for an end-user to understand this process because the end-user as a business professional or a domain expert is familiar with the following typical work flow such as getting a resident’s card: (1) A client fills out an application for the card and hands it to the service counter in the city office. (2) A clerk at the service counter checks the application and passes it to a computer operator in the back. (3) The operator enters the information about the client and gets the resident’s card. (4) The clerk receives it and confirms the contents. (5) The clerk hands it to the client.

### 4.2 Case study of business logic definition

Some examples for the reuse support system are shown. The first example is the requirement for identification and qualification of donors and donees. The following business rules are given:

1. If a citizen is a resident in the city and is more than 17 years old, the citizen can be registered.
2. Dealers cannot be registered.
3. If a citizen requests registration, then the identification must be checked.

These rules are merged into one complicated rule and the main process of the rule is defined as follows:

(1) UI: The system displays a form for registration and gets a request from a client.
(2) BL: The system checks the request according to these rules.
(3) DB: The system accesses the database for registration.

(4) BL: The system gets the results from the database.
(5) UI: The system displays the identification number.

In this process, some details are omitted such as error handling, identification check and identification number generation. The common error handling will be defined at the design phase. The method of the identification check depends on the status of e-Government. Citizens may already have an identification method via the Internet or they must visit an actual service counter once before the use of the support system. As for the identification number generation method, the system will prepare a common method such as a sequential number generator and sometimes may make a user select a form of the identification number.

The second example is the requirement for registration of items. The following business rules are given:

1. The system must require that the donor declares the item has been used in domestic life.
2. Large pieces of furniture are registered and kept at home.

If the reuse promotion services are limited to Web site services, the second rule is not necessary because it is assumed that every item will be kept at home. The main process is defined as follows:

(1) UI: The system displays a form for registration and gets a request from a client.
(2) BL: The system checks the request according to the rule.
(3) DB: The system accesses the database for registration.
(4) BL: The system gets the results from the database.
(5) UI: The system displays the results including the item registration number.

In this process, some details are omitted. The displayed form includes the check box for the declaration in addition to the information about the item.

These are examples of the case studies. Although it is supposed that there are a lot of variations in business logic, it is confirmed that the template is useful for defining the requirements based on the typical three-tier architecture of user interfaces (UI), business logic (BL) and database (DB). The definitions of business logic by using this template will promote the end-user-initiative development, especially when the domain-specific application framework and the domain-specific visual modeling tool are introduced. This is because it must be easy to understand the necessary facilities for business logic.

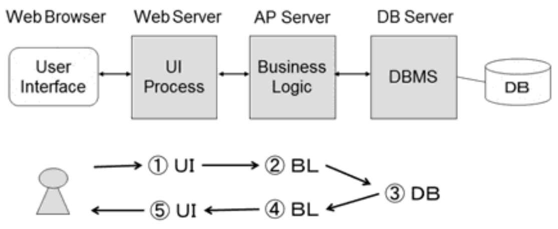


Figure 4. The UI-driven approach

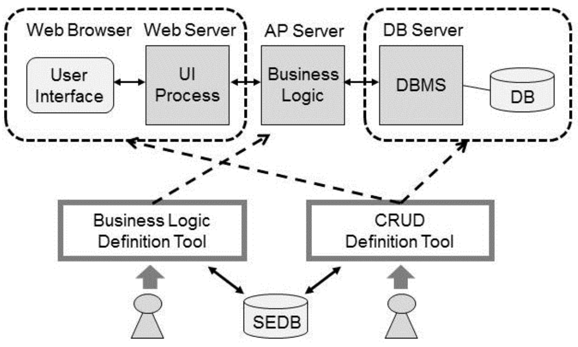


Figure 5. Web application generation process

## 5. IMPLEMENTATION TECHNIQUES

### 5.1 Web application generation process

As a result of the previous case studies, a Web application based on the typical three-tier architecture is defined by using a business logic definition tool and a CRUD definition tool as shown in Figure 5.

A Web application is defined by end-users as follows:

(1) The user interface is defined at the logical level.
(2) The DB table is defined at the logical level also.
(3) The business logic is defined based on (1) and (2).

It is important to describe these three steps at the same abstraction level. Let's consider the previous example for registration of items and apply this process to it. The first and second steps are necessary for the step of the business logic definition. Actually, these two steps are often performed simultaneously or the DB table is defined prior to the GUI definition

At the first stage of step (1), the first user interface of the UtoU template: { **UI** > BL > DB > BL > UI} is defined by listing all input columns at the logical level as follows: {the name of the donor, the member identification number, the name of the item, the details of the item, the number of photographs, the check box for a declaration that the item has been used in domestic life, the check box for a declaration that the item is not included in the list of items to be prohibited}. At the second stage of step (1), the last user interface of the UtoU template: { UI > BL > DB > BL > **UI**} is defined by listing all output columns at the logical level as follows: {the name of a donor, the item registration number}.

At step (2), the DB table of the UtoU template: { UI > BL > **DB** > BL > UI} is defined by listing all columns at the logical level as follows: {the item registration number, the name of the item, the details of the item, the number of photographs, the member identification number of the donor, the registration date, the status of "registration", "requested" and "deletion," }

At the first stage of step (3), the first business logic of the UtoU template: { UI > **BL** > DB > BL > UI} is defined by listing all input columns to be checked and internal processes as follows:

* All input columns are filled in.
* All input data are valid.
* Photographs meet the conditions if necessary.
* The identification of the registration date
* The generation of the item registration number

At the second stage of step (3), the last business logic of the UtoU template: { UI > BL > DB > **BL** > UI} is defined by listing all output columns and internal processes as follows:

* The name of the donor
* The item registration number

In this case study, exception handling is omitted to describe in detail.

As a result, it is possible that end-users of business professionals define requirement specifications of a Web application. Sometimes, they need the support of IT professionals for implementation of the complicated business logic if the component for the business logic has not been prepared in advance. One of problems for building the domain-specific framework is which components should be prepared. The range of applications is dependent on a library of components. For example, it is difficult to define the suitable specifications of a registration number generator which the system supports. This is because end-users prefer a simple function to a rich function. On the other hand, some business professionals in local governments may like to customize the functions since they do not like to change the conventional way.

## 5.2 Visual tools for modeling GUI, logic and data

Our recent work for a domain-specific framework was applied to the development of a reuse support system [8]. Version 1 was developed based on a simple framework of the JSP/Servlet model. Moreover, the SQL was created in advance, and was saved in the properties file. However, it is necessary for the end-users to define the domain-dependent portion since the source codes in JSP and Java are dependent on domains. It has been thought that such work is difficult for end-users. In version 2, we modified the first version to a component-based framework in order to construct an application not by programming but by defining the components. For end-user-initiative development, the visual tool was equipped with three basic functions of creation of DB, GUI and business logic.

Figure 6 shows the support system for the end-user. The right part is the generated web application. The gray boxes in the right part imply the framework. The left part is the visual modeling tool for application development. The end-user models DB, GUI and the business logic by using the visual modeling tool. These modeling results are kept in the JSON model, the CSS model, and the SQL statements. The framework runs applications by interpreting these models.

In the practical modeling process, first, the end-user needs to create three tables for management of members, objects, and donee's applications, respectively, while defining the name and the data type of each column in the tables on the data modeling of the visual tool. Next, the GUI is defined with the information including the items, item attributes, display style, an input/output, etc. When the end-user uses the visual tool, it needs to create the components and the pages because each page is composed of the components. The end-user creates a component by dragging and dropping the elements which are provided by the visual tool. Then, by setting up the attributes of an element, the component is saved to a JSON file as a domain component.

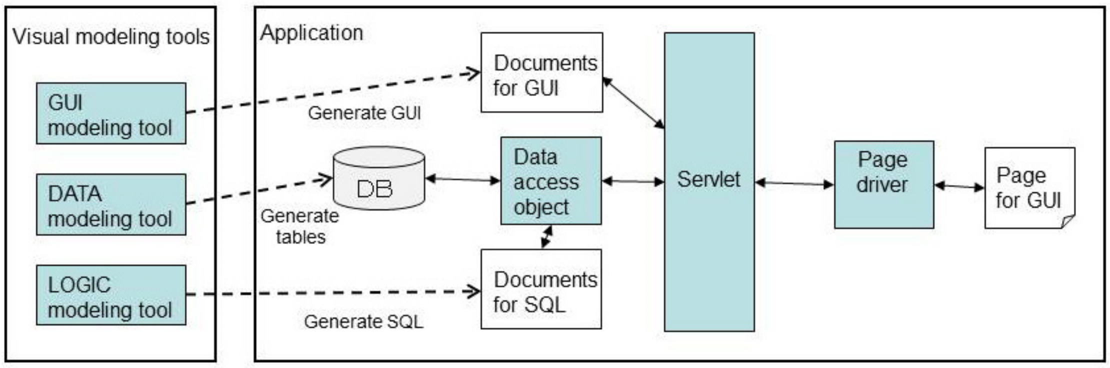
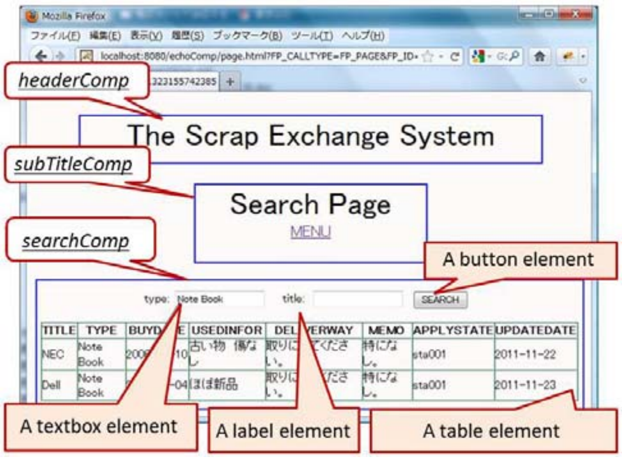


Figure 6. The visual tool and the domain-specific framework for an application

We explain the sample of creating the search component, named searchComp, which is shown in the Figure 7. It consists of two label elements, two textbox elements, a button element and a table element. The end-user adjusts the size and the position by dragging and moving these elements with the mouse. This component with these elements is saved as a JSON file named searchComp.json. The element's style information is saved as a CSS file named searchComp.css.

Next, a page is created by incorporating such components and saved as a JSON file. When the application is run, the page is displayed by calling the JSON files of the page and the related components. For example, the search page named searchPage consists of three components named headerComp, SubTitleComp and searchComp as shown in Figure 7 and is saved as a JSON file named searchPage.json. The components' style information is saved as a CSS file named searchPage.css.


Mozilla Firefox
ファイル(F) 編集(E) 表示(V) 履歴(S) ブックマーク(B) ツール(T) ヘルプ(H)
localhost:8080/echoComp/page.html?FP_CALLTYPE=FP_PAGE&FP_ID=
headerComp
subTitleComp
searchComp
A button element
A textbox element
A label element
A table element

The Scrap Exchange System

Search Page

MENU

type: Note Book title: SEARCH

| TITLE | TYPE | BUYD E | USEDINFOR | DEL ERWAY | MEMO | APPLYSTATE | UPDATEDATE |
|---|---|---|---|---|---|---|---|
| NEC | Note Book | 200 10 | 古い物 傷なし | 取りに てください。 | 特になし。 | sta001 | 2011-11-22 |
| Dell | Note Book | -04 | ほぼ新品 | 取りに ださい。 | 特になし。 | sta001 | 2011-11-23 |

Figure 7. An example of a page for the user interface

Finally, the business logic is modeled. The function for the business logic is classified into four categories: {validation, DB operation, page transition or navigation}. For example, the SQL statements for the business logic are generated by setting the reference between the data models, and the relationship of the data model and the GUI. These relationships are defined by connection lines between objects. Each column has the attribute of I or O which implies an input parameter or an output column respectively.

A Web application for the reuse promotion service was developed with the domain-specific framework, while creating three tables for management of users, items and the donee's applications. The dependence on the domain could be limited to the JSON model, the CSS model, and SQL statements. Then, it was confirmed that the domain-specific framework based on the ABC model was effective for the end-user-initiative approach.

## 6. CONCLUSION

This paper described the modeling techniques for the end-user-initiative development of Web applications with the three-tier architecture of the user interface, business logic and database. The conceptual model is based on CRUD operations. For definitions of various kinds of business logic, this paper introduced the ABC model: application = business logic + CRUD. After the DB table definitions and GUI definitions, business logic is defined by using the UtoU template of {UI > BL > DB > BL > UI}. This approach is applied to Web applications in practical use with case studies. This method was supported by domain-specific application framework and visual modeling technologies.

## ACKNOWLEDGEMENT

This work was supported in part by KAKENHI of Grant-in-Aid for Scientific Research.

## REFERENCES

[1] Chusho, T., 2012. Classification and definitions of business logic for end-user-initiative development. Proc. The 11th International Conference on Software Methodologies, Tools and Techniques (SoMeT_12), Genoa, Italy, pp. 41-56.

[2] Chusho, T. , Zhou F. and Yagi, N., 2011. End-user-initiative development with domain-specific frameworks and visual modeling. Proc. The 10th International Conference on Software Methodologies, Tools and Techniques (SoMeT_11), Saint Petersburg, Russia, pp. 57-71.

[3] Chusho, T. and Yagi, N., 2010. Visual modeling and program generation for end-user-initiative development. Proc. The 9th Joint Conference on Knowledge-Based Software Engineering (JCKBSE'10), Kaunas, lithuania, pp.102-115.

[4] Chusho, T., Tsukui, H. and Fujiwara, K., 2004. A form-base and UI-driven approach for enduser-initiative development of Web applications. Proc. Applied Computing 2004, Lisbon, Portugal, pp.II/11-II/16.

[5] Fayad, M. and Schmidt, D. C. (Ed.), 1997. Object-oriented application frameworks. *In Communications of the ACM*, Vol. 39, No. 10, pp. 32-87.

[6] Fischer, G., Nakakoji K. and Ye, Y., 2009. Metadesign guidelines for supporting domain experts in software development. *In IEEE Software*, Vol. 26, No. 5, pp. 37-44.

[7] Ko, A. J., Abraham, R., Burnett M. M. and Myers, B. A., 2009. Guest editors' introduction: End-user software engineering. *In IEEE Software*, Vol. 26, No. 5, pp. 16-17.

[8] Li, J. and Chusho, T., 2012. A Web application framework for end-user-initiative development with a visual tool. Proc. 2012 IAENG International Conference on Software Engineering (ICSE'12), Hong Kong, China, pp. 816-822.

[9] Lieberman, H. (Ed.), 2000. Special issue on programming by example. *In Communications of the ACM*, Vol. 43, No. 3, pp. 72-114.

[10] OMG, OMG Model Driven Architecture, http://www.omg.org/mda/

[11] OMG, Unified Modeling Language, http://www.uml.org/

[12] Sinha, A. P. and Jain, H., 2013. Ease of reuse: an empirical comparison of components and objects. *IEEE Software*, Vol. 30, No. 5, pp. 70-75.

[13] Sprinkle, J., Mernik, M., Tolvanen J. and Spinellis, D., 2009. Guest editors' introduction: What kinds of nails need a domain-specific hammer?. *In IEEE Software*, Vol. 26, No. 4, pp. 15-18.

[14] Sutcliffe, A. and Mehandjiev, N. (Guest Ed.), 2004. End-user development, *Communications of the ACM*, Vol. 47, No. 9, pp. 31-32.

[15] Wiegers, K. E., 2003. *Software requirements (Second Ed.)*. Microsoft Press.