End-user Initiative Requirement Definitions Based on Web Service

Takeshi Chusho, Noriyuki Yagi, and Katsuya Fujiwara

Abstract—It is expected that information technology (IT) will contribute to resource saving and environmental preservation for a sustainable society. For this purpose, application software is required and then the fund is needed for its development by IT professionals. However, the preparation of the fund must be difficult. The end-user initiative development of application software is indispensable for the solution of this dilemma. This paper describes requirement definitions based on abstract forms in a method that business professionals build applications by themselves. The abstract forms are considered as interfaces of Web services based on the simple concept that "one service = one form." Therefore, the business logic can be defined as the form transformation from input forms into outputs form by business professionals.

***Index Terms*—EUC, form-to-form transformation, visual tool, web service integration.**

I. Introduction

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 then the fund is needed for its development by IT professionals. However, the preparation of the fund 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 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 register goods to be reused or search the list of registered goods for their own use easily.

As for another example, let's consider service counters which exist everywhere. Although some service counters already support the Internet usage, many service counters have not yet done this because of lack of funds, not lack of technologies. If business professionals at a service counter can develop the application for a Web site, they will save resources because of the paperless system and reducing the cost of electricity by not using elevators when going to the actual counter.

Furthermore, let's consider online shopping. Some customers may want to buy goods from the nearest shop to reduce carbon dioxide (co2) emission in transportation. It must be useful to open a Web site where you can search several online shops for the specified goods and display the information alongside the transportation distance. This application may be developed by non-professionals of IT if Web service integration of online shopping sites and an online map service site is performed by using recent mash-up technologies.

There are several approaches for the end-user initiative development. That is, the UI-driven approach makes it possible to develop applications for the UI-centered front-end subsystems easily. It is strengthened by using framework technologies. The model-driven approach makes it possible to develop applications for the workflow-centered back-end subsystems easily. It is strengthened by using a visual modeling tool. Furthermore, the form-driven approach must be easier than the aforementioned two approaches for business professionals since they are familiar with forms in daily work. It is strengthened by the form-to-form transformation and Web service integration.

Terms for end-user computing (EUC) and papers on EUC often came out in 80's. Some papers describe definitions and classifications of EUC [7] or the management of EUC [2]. A recent paper summarizes the trends of end-user development without IT professionals' assistance [22].

There are some other works related to EUC. In the programming field, the technologies for programming by example (PBE) [16] were studied. The PBE implies that some operations are automated after a user's intention is inferred from examples of operations. The non-programming styles for various users including children and for various domains including games were proposed. In the database field, the example based database query languages [21] such as QBE (Query-By-Example) were studied. QBE implies that a DB query is executed by examples of concrete queries. User-friendly inquiry languages were proposed in comparison with SQL.

Our research target is different from these technologies and is for business professionals and business domains. The user's intention is definitely defined as requirement specifications without inference as business professionals with domain expertise develop software which executes their own jobs.

Therefore, this paper pays attention to 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 is supposed, which has been popular recently. Generally, there are three approaches corresponding to the user interface (UI), business logic and database (DB). In our studies, application frameworks, visual

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

T. Chusho and N. Yagi are with the Department of Computer Science, School of Science and Technology, Meiji University, Kawasaki, 214-0033, Japan.

K. Fujiwara is with the Department of Computer Science and Engineering, Akita University, Akita, 010-8502, Japan.

modeling tools based on components and form transformation tools for Web service integration were developed for EUC.

This paper presents Web application development technologies in Section 2, examples of applications in Section 3, issues on EUC in Section 4 and abstract forms and form transformation in Section 5.

II. WEB APPLICATION DEVELOPMENT

A. Basic Approaches

The approaches to the 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 any one of the UI (user interface), the model (business logic) or DB. These approaches are described in this section.

B. A UI-Driven Approach

Recently, a UI-driven approach has emerged as Web applications are increasing. A typical example of this approach is the Struts framework [1] which is an open source framework for building Web applications in Java. The visual forms are defined first and then components for business logic and access to the DB are defined. In this approach, it seems to be easier for the end-user to define the UI in comparison with definitions of the model or the DB.

For example, there are recent success stories on end-user computing. One is a paper that the European Union's SmartGov project transforms public-sector employees into developers of the government e-services used directly by the public [15]. An intelligent e-forms development and maintenance environment and associated framework are delivered.

Another one is performed at the office of Nagasaki prefecture in Japan [11]. The staff of business professionals designed and described the user interfaces without IT professionals' assistances. Furthermore, while the staffs specified requirements on business logic and DB tables, IT professionals described documents of design specifications. Based on these documents, the system was divided into subsystems as the development cost of each subsystem was less than about fifty thousands dollars and a small local IT company could undertake the small-scale subsystem development. As a result, the risk of ambiguous requirements was omitted, the cost was reduced to about 50%, and customer satisfaction based on usability etc. was improved. This story suggests that the UI-driven approach makes it possible for end-users to define the requirements of their own software.

We have also been studying this approach for several years [5]. The UI-driven approach is proposed for the front-end subsystem based on CBSE (Component-Based Software Engineering) [3], [8]. The systems are constructed by using UI-centered frameworks [10] and agent technologies [13]. The effectiveness of the UI-driven process is confirmed through experiences with the development of frameworks.

Business professionals define requirements for an application to be developed by using the framework as follows:

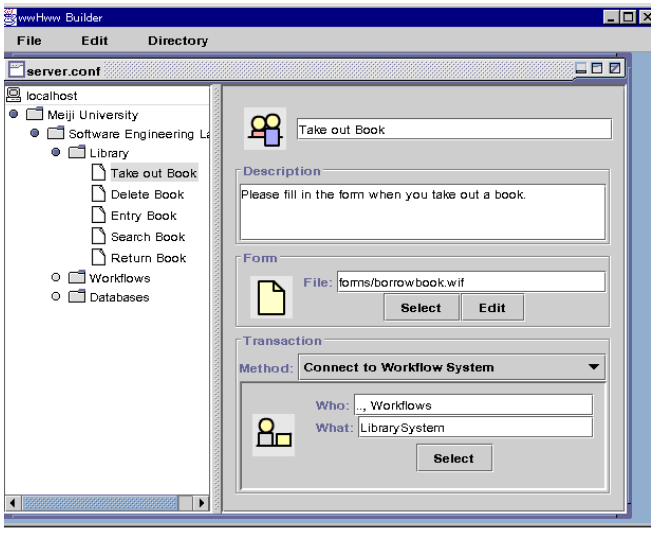


Figure 1. The browser for system definitions by end-users.

1. Service definitions : Services at the counter are defined.
2. Form definitions : Forms for these services are defined with navigational information.
3. Registration : These form definitions are registered into the corresponding servers.

An example of a browser defining the library system is shown in Figure 1. The left part implies a hierarchical directory. The right part implies definitions about the service for taking out books.

Intelligent navigation by agents is implemented in XML. The metadata for a window is described in an RDF (Resource Description Framework) style. While forms are defined in HTML in the conventional way, the semantics of forms are defined in RDF style also.

However, this framework for a service counter does not support the back-end subsystem with the workflow and DB. When another framework for a reservation task such as a room reservation system was developed, a visual tool for defining the DB table easily was developed simultaneously. Although end-users can use this tool, the target DB table is limited to a simple reservation table.

C. A Model-Driven Approach

Around the 90's, object-oriented analysis and design (OOAD) technologies came out and have become the major methodologies. Some of them match the waterfall model and others match the iterative and/or incremental development process [12], [14]. In the recent OOAD methodologies, the unified modeling language (UML) [20] is used for definitions of the system model. OOAD is a model-driven approach. In addition, UML2.0 requires more rigorous definitions of models for automatic generation of program codes based on the model-driven architecture (MDA) [19].

We have also been studying this approach for several years [4]. The model-driven approach based on CBSE is proposed for the back-end subsystem which the main part is a workflow. Our solution is given as 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 a computation model based on an

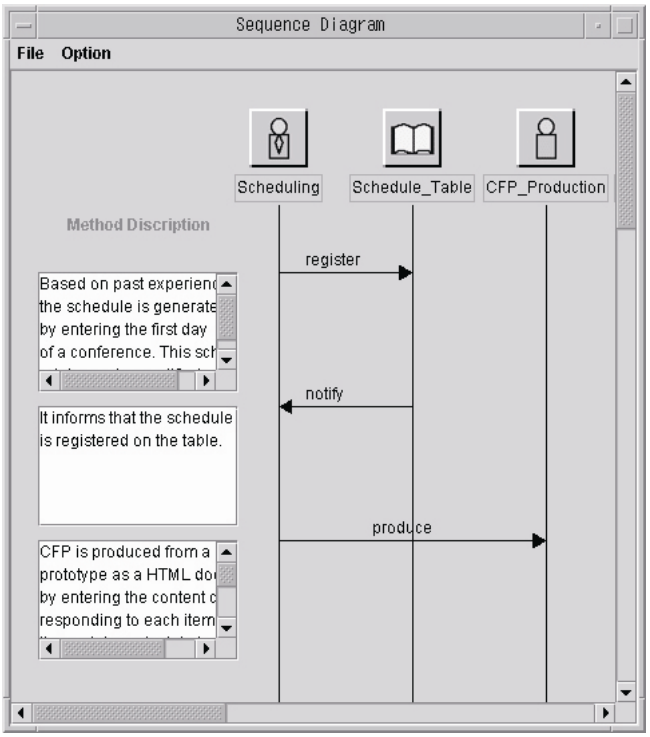


Figure 2. An example of the requirement specifications verified by modeling and simulation.

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. This process requires necessarily the fixed architecture and ready-made components such as business objects.

Our approach is different from most conventional object-oriented analysis and/or design methods which need defining an object model on static structure of objects prior to a dynamic model on interactive behavior among objects. At the first stage, the system behavior is expressed as a message-driven model by using a visual modeling tool while focusing on message flow and components. At the second stage, a user interface is generated automatically and may be customized if necessary. Then the transition diagram of user interfaces is generated automatically and used for confirmation of external specifications of the application. Finally, the system behavior is verified by using a simulation tool.

This component-based development process was confirmed by feasibility study on a given problem of the IPSJ(the Information Processing Society of Japan) sigRE group. The problem is how to define requirement specifications for a program chair's job of an academic conference. A dynamic model was constructed while introducing eleven kinds of objects. These objects are defined by drag-and-drop operations from the palette of icons. A message between objects is defined by drawing an arrow from the source object to the drain object.

In addition, branch conditions are described in rule expressions. For example, in the "produce" method of the CFP Production object, the following rules are described for branch conditions:

- if Printer = yes then print;
- if CFP Distribution = yes then distribute;

Furthermore, simulation is executed for validation of the requirement definitions as shown in Figure 2, both on the domain model and on the sequence diagram, while displaying traces of the message flows.

D. A Data-Driven Approach

As for a data-driven approach, a data-centered or data-oriented approach was introduced in the 80's. In this method, the data flow diagrams (DFD) are sometimes used for a definition of the workflow. The data model is defined with entity relationship diagrams (ERD). In many mission-critical applications, the DB design is the most important. Since data structures are more stable than business logic, the data model is defined prior to business logic.

However, the design of a large-scale DB is difficult for end-users. In our UI-driven approach and/or the model-driven approach, it is supposed that DB components are used. As for a small-scale DB, visual tools are introduced for defining the DB tables.

III. TYPICAL APPLICATIONS

These technologies for end-user computing will be applied to various applications for a sustainable society.

As for a Web application for a thrift store, the UI-driven approach for a front-end subsystem and the model-driven approach for a back-end subsystem are applied. If a Web site for a thrift store can be opened easily by business professionals, natural resources can be saved and the opportunity to reuse them great.

As for a Web application for a service counter, framework technologies are suitable because the domain expertise is embedded into a framework which is prepared in advance. As for the latest example, the framework for a reservation system was developed and applied to a meeting room reservation system for our department. This application is in practical use. In comparison with the previous frameworks, this framework uses the open source framework for building Web applications, Struts. JavaServer Pages (JSP) are used for dynamic Web pages since the current state of reservation should be displayed. The system architecture is based on the Struts framework. As a result, use of natural resources is reduced by a paperless system, and the energy for elevators, trains, cars, etc. is saved since it is not necessary to visit the actual service counter.

Furthermore, if an Internet shopping site for shopping and a map site are combined using Web service integration, clients can buy goods from the nearest shop and there will be a reduction of CO2 emission usually in transportation.

IV. ISSUES ON END-USER COMPUTING

In our experiences, sometimes end-users needed IT professionals' assistance. It is difficult for end-users to develop new components, to modify ready-made components for complicated business logic and to implement user interfaces in JSP.

However, if end-users can describe requirement specifications, some IT professionals may continue

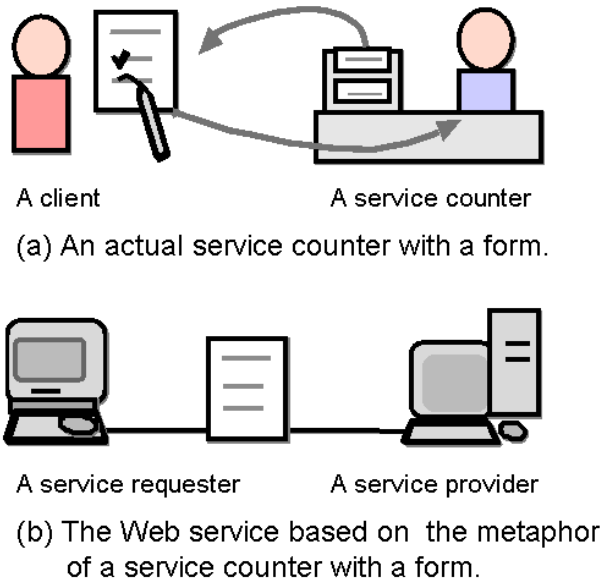


Figure 3: A service counter as a metaphor for Web service.

application development as volunteers or some IT companies may undertake application development at a low cost. Recently, the ratio of the requirement definitions cost to the total cost on software development has been increasing since the productivity of design, implementation and testing has been improved by using various tools and object-oriented platforms.

Furthermore, in the business world, the external specifications of application software are recently considered as services as shown in keywords such as ASP (Application Service Provider), Web service, SOA (Service-Oriented Architecture) and SaaS (Software as a Service) [9], [17], [18]. Our new approach to end-user computing is that end-users develop Web applications by service integration for both the front-end subsystem and the back-end subsystem because end-users consider their applications as a level of service, not as a level of software.

That is, the service counter is considered as a metaphor to describe the interface between a service provider and a service requester for Web services. Such a service counter is not limited to the actual service counter in the real world. For example, in a supply chain management system, data exchange among related applications can be considered as data exchange at the virtual service counter.

Generally, the service counter receives service requests from clients as shown in Figure 3. Forms are considered as the interface between them. That is, the following concept is essential for our approach:

”One service = One form.”

The integration of some individual Web services is considered as transformation from some input forms into some output forms. Although most of these forms are not visual forms, end-users can consider this form as a visual form for the requirement specification. Such a form is called an abstract form in this paper. Since end-users can consider such Web service integration as the workflow with visual forms which they are familiar with, IT skills are not required of end-users.

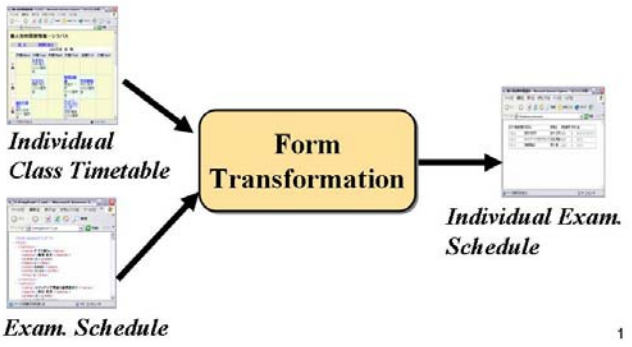


Figure 4: Form transformation for Web service integration.

Furthermore, our previous two approaches are unified by these concepts. The UI-driven approach with frameworks for front-end subsystems is considered as the special case that a part of abstract forms are actually visual forms for interaction between the system and the external world. The model-driven approach with visual modeling tools for back-end subsystems is considered as the special case that the message flow is used instead of the form flow as the workflow. That is, cooperative work at an office is expressed by using a form flow model with the abstract forms.

The remainder of this paper describes requirement definitions by the abstract form and the form transformation procedure from inputs to outputs.

V. Abstract Forms and Transformation

A. Form Transformation in XSLT

The best solution is that end-users can get application software by form definitions and form-to-form transformation definitions. An application which generates individual examination schedules for each student has been selected for applying our solutions to practical Web service integration. Actually, the university supports the individual portal sites for each student. The student gets the examination schedule in PDF and the individual timetable for classes in HTML. In our experiment, an actual examination schedule in PDF can be transformed into an XML document manually.

The target application generates an individual examination schedule for each student from the individual timetable for classes and the examination schedule. The form transformation is shown in Figure 4.

One input is the individual timetable for classes in HTML which is extracted from the individual portal site for each student. This document includes information about subjects for each student, that is, subject names, instructor names and class numbers. This HTML document is transformed into an XML document by using the wrapping technology. The other input is the examination schedule in XML, which includes information about subject names, instructor names, dates and periods, and room numbers. These two XML documents are merged into the individual examination schedule in XML format for each student.

This individual examination schedule in XML is transformed into an HTML document which can be displayed on the Web browser of each student. There are some conventional tools used for this transformation. The XSLT

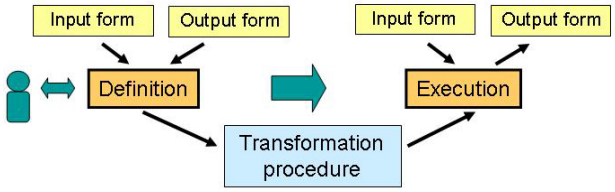


Figure 5. Form transformation by mapping.

stylesheet for this application is generated by using one of the conventional tools.

The key technology of this system is the form-to-form transformation from two XML documents into an XML document. The system administrator of this application is not an IT professional but a clerk in the university office. Such an end-user does not have the ability to perform programming, but needs to modify the system when the inputs change.

For the solution of this problem, basically, the procedure of this application is described in a script language. Furthermore, a visual tool supports the end-user. The system generates the XML document as the output by extracting classes which are included in the both input files. The early opinions on this approach are described in detail in [6].

B. Form Transformation by Mapping

One solution to the problem of the form transformation in XSLT is the form transformation by mapping from input forms to output forms. The end-users do not need to learn XML and XSLT technologies since they can define the form transformation procedure by only mouse manipulations to relate items in input forms to items in output forms. After the definition of this procedure, the form transformation from input forms into output forms is executed as shown in Figure 5.

For this study, a Web application for the reuse of laboratory equipment was selected. In the School of Science and Technology which we belong to, a lot of secondhand equipment such as PCs are thrown away although many of them can still be used. If the reuse site is open, the available but unnecessary equipment is registered there and someone can find and receive the reusable equipment easily. Therefore, our end-user initiative requirement definitions method is applied to such a system, ICRS (the Ikuta Campus Reuse System).

Main rules for this system are as follows:

1. Users are limited to members who have mail addresses which are managed by the university, that is, teachers, officers, students etc. for security check.

2. The equipment to be given should be free for avoiding illegal dealing of the university property.

3. The site administrator takes no responsibility for any troubles since this site is supported by volunteers.

Main functions are described in the usecase diagram of UML as shown in Figure 6. The user actor is the superclass of both the donor actor and the donee actor. The relations between actors and usecases are described as follows:

- The donor registers the unnecessary equipment.
- The donor replies to the donee.
- The donee receives the equipment for reuse.
- The donee requests the necessary equipment.

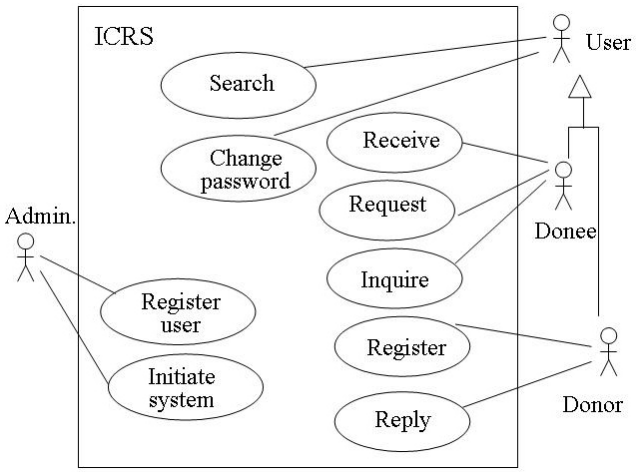


Figure 6. Usecases of the system for the reuse site.

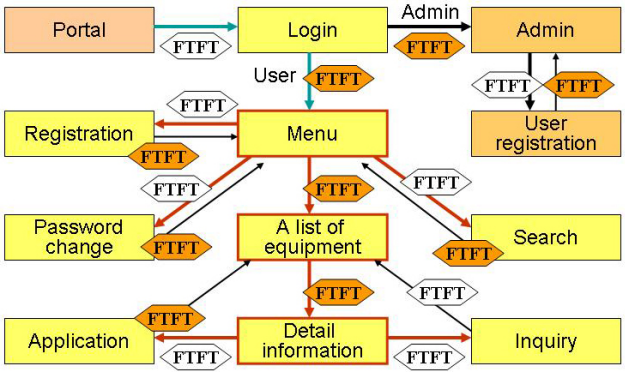


Figure 7. UI transition and two types of FTFT.

- The donee inquires about the registered equipment.
- The user searches a list of the registered equipment.
- The user changes his/her password.
- The administrator registers a user.
- The administrator initiates the system.

The user interfaces (UI) and the UI transition diagram are designed as shown in Figure 7. The business logic is specified by the form-to-form transformation (FTFT) with abstract forms. In this figure, the following two types of FTFT are discriminated:

1. FTFT with DB accesses via abstract forms.
2. FTFT between visual forms

The first item is marked with the FTFT by a gray hexagon and implies that a transformation between abstract forms or between abstract forms and visual forms are performed before moving onto the next user interface with a visual form. On the other hand, the second item is marked with the FTFT of a white hexagon and implies a transformation between visual forms.

Figure 8 shows a part of form flows and form-to-form transformations. The three forms of left-hand side are visual forms for actual user interfaces corresponding to Login, Menu and List windows. The four forms of the right-hand side are abstract forms for end-users support, which are not displayed visually at the application execution time. Such an abstract form is used under construction of an application by

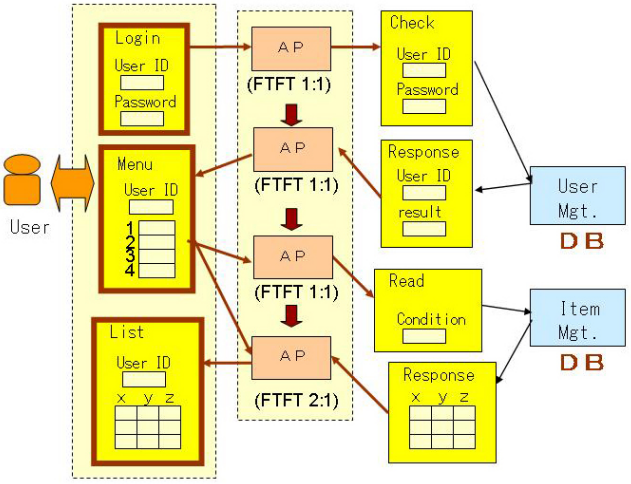


Figure 8. A part of form-to-form transformations and database accesses.

end-users. M:N of FTFT implies the transformation from M input forms to N output forms. First three transformations are 1:1 and the last transformation is 2:1.

The Login form is transformed into the abstract form of 'Check' and it is sent to the user management DB. Next, the abstract form of 'Response' is transformed into the Menu form for selection of the next operation from a display of a list of the registered equipment, registration of unnecessary equipment, search of registered equipment or password change. If the user selects a display of a list of the registered equipment, the menu form is transformed into the abstract form of 'Read' and it is sent to the Item management DB. Then two inputs of the Menu form and the Response form are merged and transformed into the List form.

C. A Visual Tool for FTFT

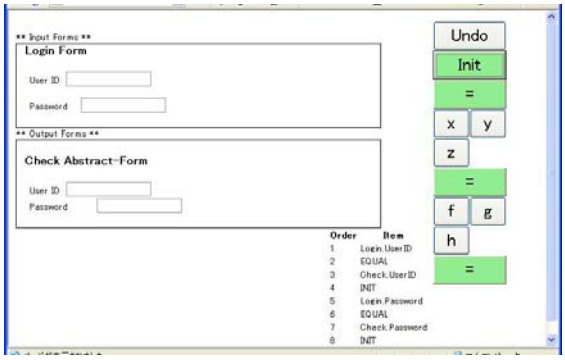
A tool for defining the form-to-form transformation was developed. The user interface was implemented in HTML and JavaScript. The generated procedure in XML is sent to the server and stored there. The interpreter of this procedure in XML was implemented in Java.

Figure 9 shows examples of the form-to-form transformation. The input form and the output form are displayed on the left-hand side. The palette with buttons for operation items is displayed on the right-hand side. Whenever a column of forms or an operation item of the palette is clicked, the order and the name of the clicked item are displayed below for confirmation.

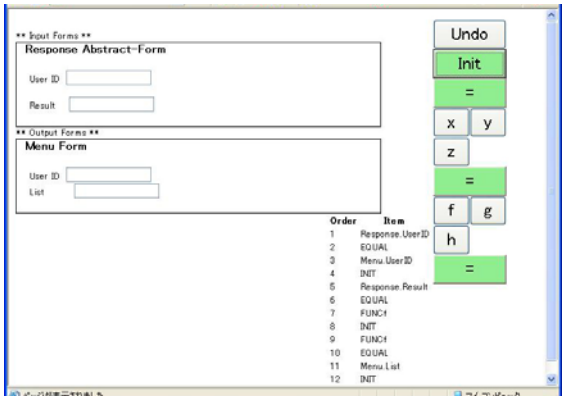
The transformation from the Login form into the Check form in Figure 9 (a) is defined as follows:

1 Login.UserID
2 EQUAL
3 Check.UserID
4 INIT
5 Login.Password
6 EQUAL
7 Check.Password
8 INIT

The first four operations define that the value of the User ID column in the input form is copied into the User ID column in the output form while clicking the mouse button in order of



(a) Transformation from the Login form to a Check form



(b) Transformation from the Response form to a Menu form

Figure 9. Examples of FTFT by using a visual tool.

{Login.UserID, =, Check.UserID, INIT}. The INIT operation implies the initialization as the previous execution result is not used. The following four operations define that the value of the Password column in the input form is copied into the Password column in the output form. This example is very simple.

The transformation from the Response form into the Menu form in Figure 9(b) is defined as follows:

1 Response.UserID
2 EQUAL
3 Menu.UserID
4 INIT
5 Response.Result
6 EQUAL
7 FUNCf
8 INIT
9 FUNCf
10 EQUAL
11 Menu.List
12 INIT

The first four operations define that the value of the User ID column in the input form is copied into the User ID column in the output form. Next, one of the functions of f, g and h is used for complex business logic. That is, the following four operations define that the value of the Result column in the input form is the input of the f function while clicking the mouse button in order of {Response.Result, =, f, INIT}. The last four operations define that the output of the f function is

assigned to the Menu column in the output form likewise.

The body of the function, f, will be implemented in a scripting language later. The variables of x, y and z are used for temporary stores of execution results.

VI. Conclusion

The end-user initiative requirement definitions are necessary for developing application software for a sustainable society. This paper described the requirement definition method based on abstract forms since business professionals are familiar with visual forms. The business logic can be defined as the form transformation from input forms into output forms. Our experiments confirmed the effectiveness of this approach.

References

[1] The Apache Software Foundation, Struts, http://struts.apache.org/

[2] J. C. Brancheau, and C. V. Brown, "The management of end-user computing: status and directions," *ACM Computing Surveys*, vol.25, no.4, pp. 437–482, 1993.

[3] A. W. Brown(Ed.), *Component-Based Software Engineering*. IEEE CS Press. 1996.

[4] T. Chusho, H. Ishigure, N. Konda, and T. Iwata, "Component-Based Application Development on Architecture of a Model, UI and Components," *Proc. APSEC2000*, IEEE Computer Society, pp.349-353, 2000.

[5] T. Chusho, H. Tsukui, and K. Fujiwara, "A Form-base and UI-Driven Approach for Enduser-Initiative Development of Web Applications," *Proc. Applied Computing 2004*, IADIS, pp.II/11-II/16, 2004.

[6] T. Chusho, R. Yuasa, S. Nishida, and K. Fujiwara, "Web Service Integration Based on Abstract Forms in XML for End-user Initiative Development," *Proc. The 2007 IAENG International Conference on Internet Computing and Web Services(ICICWS'07)*, pp.950-957, 2007.

[7] W. W. Cotterman, and K. Kumar, "User cube: a taxonomy of end users," *Communications of the ACM*, vol.32, no.11, pp. 1313-1320, 1989.

[8] I. Crnkovic, et al., "Specification, Implementation, and Deployment of Components," *Communications of the ACM*, vol. 45, no. 10, pp.35-40. 2002.

[9] A. Elfatatry, "Dealing with change: components versus services," *Communications of the ACM*, vol. 50, no. 8, pp. 35-39, 2007.

[10] M. Fayad, and D. C. Schmidt, (Ed.), "Object-Oriented Application Frameworks," *Communications of the ACM*, vol.39, no.10, pp. 32-87, 1997.

[11] N. Hirooka, "Nagasaki Prefecture," (in Japanese), *Nikkei Computer*, No.2007.7.25, 2005.

[12] J. Jacobson, et al., *The Unified Software Development Process*. Addison-Wesley, 1999.

[13] N. R. Jennings, "An Agent-Based Approach for Building Complex Software Systems," *Communications of the ACM*, vol. 44, no. 4, pp.35-41, 2001.

[14] C. Larman, *Introduction to Object-Oriented Analysis and Design and the Unified Process*. Prentice-Hall, 2002.

[15] G. Lepouras, C. Vassilakis, C. Halatsis, and P. Georgiadis, "Domain expert user development: the smartgov approach," *Communications of the ACM*, vol. 50, No. 9, pp. 79-83, 2007.

[16] H. Lieberman, (Ed.), "Special issue on Programming by example," *Communications of the ACM*, vol.43, no.3, pp.72-114, 2000.

[17] T. Margaria, (Ed.), "Guest Editors' Introduction:Service Is in the Eyes of the Beholder," *IEEE Computer*, vol. 40, no. 11, pp. 33-37, 2007.

[18] O. Nano, and A. Zisman, (Ed.), "Guest Editors' Introduction: Realizing Service-Centric Software Systems," *IEEE Software*, vol. 24, no. 6, pp. 28-30, 2007.

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

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

[21] G. Ozsoyoglu, and H. Wang, "Example-Based Graphical Database Query Languages," *IEEE Computer*, vol.26, no.5, pp.25-38, 1993.

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