A Form-based and UI-Driven Approach for Enduser-Initiative Development of Web Applications

Takeshi Chusho and Hiroshi Tsukui
Department of Computer Science, Meiji University
Kawasaki, 214-8571, Japan
chusho@cs.meiji.ac.jp

Katsuya Fujiwara
Department of Computer Science and engineering, Akita University
Akita, 010-8502, Japan
fujiwara@ie.akita-u.ac.jp

ABSTRACT

The number of end-users using the Internet inside and outside of the office has been increasing. The application development by the end-users has become important for the automation of their own tasks. In particular, Web applications should be supported by business professionals themselves since their specifications must be modified frequently. This paper describes enduser-initiative Web application development methodologies based on the 3-tier architecture. Basically, approaches 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), model or DB. In this paper, the UI-driven approach is proposed for constructing Web applications with frameworks. The reusability, the usability and the UI-driven process are confirmed through experiences with the development of frameworks.

KEYWORDS

Web application, end-user computing, framework, agent, object-oriented technology, CBSE

1. INTRODUCTION

The number of end-users using the Internet inside and outside of the office has been increasing. The application development by the end-users has become important for the automation of their own tasks. In particular, many Web applications should be developed on business professionals' own initiative because they frequently need to make modification to the specifications of the applications as the business world is constantly changing. This method is called enduser-initiative development in this paper.

For the typical Web application, we direct our attention to the application system for windows or counters in banks, city offices, travel agents, mail-order companies, etc. Some of the services provided at these windows have already been implemented in the Internet for practical use such as online shopping. In the near future, the information society will require such new technologies so that business professionals can automate their own work by themselves and almost all clients can operate computers at home or at the office without extra training or the help of others.

This paper describes enduser-initiative Web application development methodologies based on the 3-tier architecture. Basically, approaches 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), model or DB. The UI-driven approach is proposed for the front-end subsystem based on CBSE (Component-Based Software Engineering) [brown-96, crnkovic-02]. The systems are constructed by using UI-centered frameworks and agent technologies. The reusability with frameworks, the usability with a multi-agent system and effectiveness of the UI-driven process are confirmed through experiences with the development of frameworks.

2. ARCHITECTURE AND PROCESS MODELS

First, let's review several conventional process models and analysis/design methodologies with respect to the 3-tier architecture. Generally, development processes are classified into three categories of UI-driven, model-driven and data-driven processes based on the 3-tier architecture of the UI, model and DB.

In 70's and 80's, structured analysis (SA) was one of the typical methodologies under the waterfall model. In this method, the data flow diagram (DFD) is used for a definition of the domain model. The entity-relationship diagrams (ERD) are used for data model definitions. SA is a model-driven approach because the DFD is first constructed from the view of function decomposition. Around the 90's, object-oriented analysis and design (OOAD) technologies were introduced and have become one of the major methodologies. Some of them match the waterfall model and others match the iterative and/or incremental development process [jacobson-99, larman-02]. In the recent OOAD methodologies, the unified modeling language (UML) [omg-uml] is used for definitions of the system model. A class diagram of UML is used for data model. OOAD is also 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) [omg-mda].

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

Recently, a UI-driven approach has emerged as Web applications are increasing. A typical example of such approach is the Struts framework [apache-struts] as an open source framework for building Java Web applications. The forms as the UI are defined first and then components for business logic and access to the DB are defined. This approach is also suitable for IT professionals, but not for end-users that lack the programming skills. In this approach, however, it seems to be easier for end-user to define the UI than the model or the DB. We have been studying this approach for several years.

3. ENDUSER-INITIATIVE APPROACH

The Internet-based business is rapidly changing and thus requires shortening of the application development period from defining a new business model to releasing new services. Furthermore, after the release, the application should be maintained constantly as the business world changes. Conventional ways of system engineers developing and maintaining applications are no longer suitable because of the shortage of time.

Our approach to how to make Web applications is shown in Figure 1. The business model at the business level of the three horizontal layers is proposed by business professionals. Then, at the service level, the domain model is constructed while specifying the required services. That is, the requirement specifications of the application for the business model are defined. At the software level, the domain model is implemented by combining components. For enduser-initiative development, it is essential for end-users to construct the domain model.

In this approach, there are two technological gaps, that is, the granularity gap between components and the domain model and the semantic gap between the domain model and end-users. The granularity gap is bridged by business objects, patterns [gamma-95] and frameworks [fayad-97, johnson-97] based on CBSE. In particular, frameworks seem to contribute to enduser-initiative development because of the reusability. On the other hand, the semantic gap is bridged by multi-agent systems. Since usability is important in Web applications, navigation by multi-agent systems must be the solution for usability improvement. Business professionals pass their expertise to agents [bradshaw-97, jennings-01], and then the agents can assist users.

For the past several years, we took two approaches of the UI-driven process and the model-driven process [chusho-00]. In our UI-driven approach, the forms are defined first and the framework with components is used. The component for DB is used also while defining only table items. The business logic depending on the application is defined with the form definitions. The other logic is embedded into the framework. That is, the framework implies a reusable semi-complete application that can be specialized to produce a custom application.

On the other hand, in our model-driven approach for the back-end, the forms are generated automatically based on information on the model, which is defined in message flow diagrams by using a visual tool. In this

approach, there are two technical problems for enduser-initiative development. One is how to describe complicated business logic. Our solution is that the business logic depending on the application is defined in the rules with the message flow. The other problem is how to get all requisite business components. End-users may order new components. Therefore, the model-driven process looks more difficult for end-users in comparison with the UI-driven process.

In the remainder of this paper, our experiences with the UI-driven approach are described.

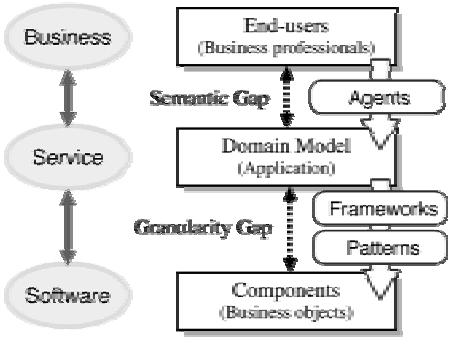

Figure 1. Technologies for enduser-initiative development.

4. FORM-BASED DOMAIN-SPECIFIC FRAMEWORK

4.1 Framework for window work

As a typical distributed information system, we direct our attention to application systems for window work. Such window work is not limited to actual window work in the real world. For example, in a supply chain management system (SCM), exchanges of data among related applications can be considered as virtual window work. Of course, many kinds of window work have already been put to practical use in the Internet and intranets. However, since these systems have been developed by IT professionals, not by end-users, they must have been quite expensive. Furthermore, although business professionals require to make frequent modification of specifications in these systems for service upgrade, it is difficult to modify these software timely since the business professionals can not maintain the systems by themselves and need to ask the IT professionals instead. Our goal is to allow end-users the opportunity to develop and maintain Web applications.

Generally, window work is considered as interaction between service requesters and service providers. The forms to be filled are considered as the interface between them. That is, the concept of "One service = One form" is essential for our approach.

The first version of the framework for window work, wwHww, was developed for a library system that was used in our laboratory. This Web application was helpful to us since there were no librarians in our laboratory. This system can be used in conventional libraries under the condition that only librarians can operate the system. The software architecture is similar to Figure 2, which will be referred to in the next subsection as a multi-agent framework. The architecture of the first version has no agent facilities. The wwHww browser on the client side is composed of two subsystems, the form browser and the directory browser. The wwHww server of the server side is composed of three subsystems, which are the directory server, the form server and the transaction server. The directory browser and the directory server are not necessary for individual applications but are used for searching of a target site among many service sites that are supported by Web applications for window work. This system was implemented in Java, and there are Java applet versions and Java application versions for two browsers.

The framework for window work includes the wwHww browser and the wwHww server but does not include the workflow system and the DBMS, which are also implemented in Java and Oracle. The wwHww framework is composed of 19 classes. The total number of Java source codes for the framework is 1,703 lines.

The reuse rate is 86% since 1,468 lines were the frozen spots reused. In addition, the workflow system is composed of 7 classes. The number of Java source codes for the workflow system is 353 lines.

In a framework, the frozen spot implies the fixed and reusable part that we need not customize. A high reuse rate reduces the end-user's work in application development. Conversely, the hot spot implies the flexible part that we need to customize. The two types of customization on hot spots are required for applying the framework to the application development: the use of plug-in components and the definition of properties.

The five forms defined correspond to the five services of registration, deletion, lending, return and search. With respect to window work, these five forms satisfied the system requirements although the DB tables were designed and implemented individually. As a result, it was confirmed that the UI-driven approach was suitable for such a system. Actually almost all codes for the customization of hot spots are required for the form implementation.

4.2 Multi-agent framework for navigation

The semantic gap between the domain model and end-users is bridged by multi-agent systems while the granularity gap between components and the domain model is bridged by frameworks as shown in Figure 1. The usability of Web applications, in particular, is improved by agent technologies. A customizable multi-agent system was developed by enhancing the framework for the window work mentioned before. Agent facilities were added to the previous architecture as shown in Figure 2. The electronic form processing is navigated by agents in both client terminals and server-at-windows. Clients can give plain text information such as their names, addresses and phone numbers generally used to fill out forms to their agents. Then their agents will fill out the form automatically for their client [chusho-02]. Business professionals can pass their expertise to their agents. Then the agents can assist clients in filling out forms and checking written forms.

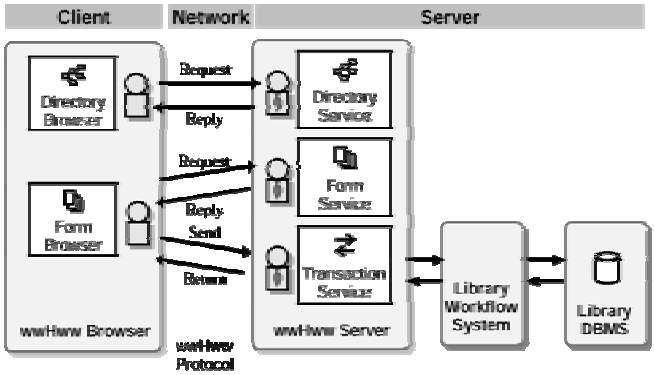

Figure 2. An application and the multi-agent framework.

Figure 3. The browser for system definitions by end-users.

Business professionals build expert agents by using the framework as follows:

1. Service definitions : Services at the window are defined.
2. Form definitions : Electronic forms for these services are defined with navigation 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 3. 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 meta data 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 an RDF style also. An example of a library is partly shown as follows:

```
<!DOCTYPE RDF [<!ENTITY site 'http://se.cs.meiji.ac.jp'>]>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:d="http://purl.org/dc/elements/1.1/"
 xmlns:w="http://wwhww.org/1.0/">
<w:Agent rdf:about="&site;/library/">
```

```
  <d:Title>Library</d:Title>
  <d:Description>The library system of the SE lab.</d:Description>
  <d:Text>The library system of the SE lab.&lt;br&gt;Book take-out service</d:Text>
  <w:name>/Meiji-U/CS/SE/Library</w:name>
  <w:service rdf:resource="&site;/library/takeout"/>
    …
</w:Agent>
<w:Form rdf:about="&site;/library/takeout">
  <d:Title>take-out</d:Title>
  <d:Description>A procedure for book take-out</d:Description>
</w:Form>
</rdf:RDF>
```

5. UI-DRIVEN APPROACH

One of the major functions for the front-end of a Web application is for the UI to facilitate interactions between the service requester and the service provider. We have had three experiences of UI-centered framework development. These experiences are analyzed in the three aspects of reusability, usability and the UI-driven process. In our first experience with the framework for window work, high reusability was confirmed. In the second experience with the multi-agent framework for navigation, usability was confirmed. In these frameworks, however, there were only five forms, and interaction among these forms was few because services by these forms are independent of each other. Therefore, there is not enough information to confirm the effectiveness of the UI-driven process. The third experience with the framework for reservation will be described next. This framework includes the business logic of the back-end subsystem and has a lot of forms that interact with each other on a UI transition diagram.

The framework for reservation was developed and applied to a meeting room reservation system for our department. This application is in practical use now. In comparison with the previous frameworks, this framework uses open source framework for building Web applications, Struts, and limits an application domain to reservation services. JavaServer Pages (JSP) is used for dynamic Web pages since the current state of reservation should be displayed. A system architecture based on the Struts framework is shown in Figure 4.

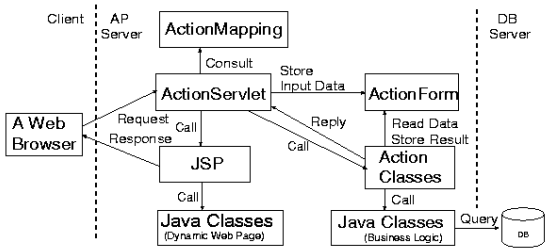

Figure 4. A system architecture based on Struts.

	Java classes (logic)	Action classes	Action-Form	JSP	Java classes (pages)	Action-Mapping	Total (lines)
Total (lines)	740	370	170	320	550	100	2250
Frozen spot	460	280	140	190	0	60	1130
Hot spot	280	90	30	130	550	40	1120
Reuse rate	62%	76%	82%	63%	0%	60%	50%
Num. of classes	6	6	3	5	2	-	22

Figure 5. Program sizes and reuse rates of the framework.

The first version of the framework for reservation was implemented as shown in Figure 5. The frozen spot implies reusable lines of the source codes. The reuse rate is calculated by {the frozen spot / the total }. The implementation of the ActionServlet is included in the Struts. The ActionMapping is described in XML. Consequently, in an example of applying this framework to a meeting room reservation system, the reuse rate is 50% since 1,130 lines among the total of 2,250 lines, including the XML document, were reused as the frozen spots.

In this application, a lot of the forms were developed since the usability was important. However, these forms required 550 lines of unreusable codes as the Java classes for dynamic Web pages. To solve this problem, a visual tool for defining the dynamic Web pages has been developed. As a result, 240 lines are generated automatically among 550 lines. The reusability was improved to 61%.

A total of 23 forms were developed in this system - 10 forms for end-users and 13 forms for the administrator. Among them, 20 forms were dynamic Web pages. The UI transition diagram on the 10 kinds of forms for end-users is shown in Figure 6. The transitions by trivial input errors are omitted. The form

design and the UI transition definition were performed concurrently because each link button on a form implied a transition to another form. In our experience, form design is considered as requirement definitions. Almost all functions are directly mapped into some forms. The functional sufficiency and the usability can be evaluated and improved easily since all usecases are simulated on the forms and the UI transitions.

As for the back-end subsystem corresponding to the workflow of the domain model, it is generally excluded from our frameworks. Although the framework for lending service and the framework for reservation include DB access, they require only DB table definitions for the application development. Furthermore, they do not include the complicated workflow or the complicated business logic. Therefore, most of requirements for applications that were developed by using these frameworks were satisfied by defining the forms. As a result, it was confirmed that the UI-driven approach was suitable.

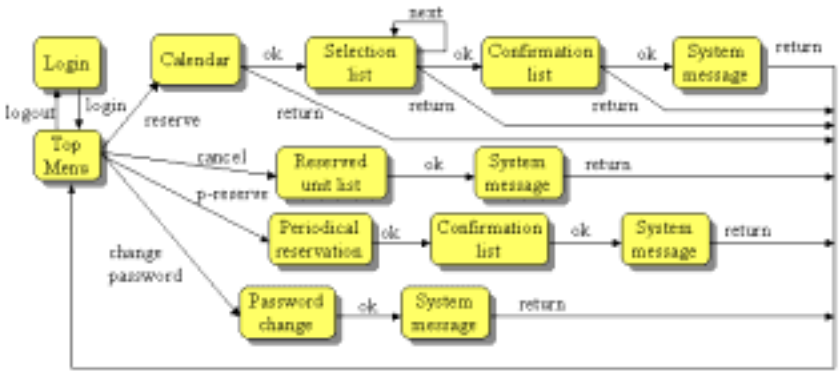

Figure 6. A UI transition diagram on an application for reservation.

6. CONCLUSION

This paper described enduser-initiative Web application development methodologies based on the 3-tier architecture. A UI-driven approach was proposed for the front-end subsystem, which is constructed by using a UI-centered framework. The effectiveness on the reusability, the usability and the UI-driven process were confirmed through experiences with the development of frameworks,.

REFERENCES

The Apache Jakarta project, STRUTS. http://jakarta.apache.org/struts/.

Bradshaw, J. M., 1997. An Introduction to Software Agent. *Software Agent*, MIT Press, pp.3-46.

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

Chusho, T. et al., 2000. Component-Based Application Development on Architecture of a Model, UI and Components. *Proc. APSEC2000*, Sigapore, IEEE Computer Society, pp.349-353.

Chusho, T. et al., 2002. Automatic Filling in a Form by an Agent for Web Applications. *Proc. APSEC2002*, Gold Coast, Australia, IEEE Computer Society, pp.239-247.

Crnkovic, I. et al., 2002. Specification, Implementation, and Deployment of Components. *In Commun. ACM*, Vol. 45, No. 10, pp.35-40.

Fayad, M. and Schmidt, D. C. (Ed.), 1997. Object-Oriented Application Frameworks. *In Commun. ACM*, Vol. 39, No. 10, pp. 32-87.

Gamma, E. et al., 1995. *Design Patterns.* Addison-Wesley.

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

Jennings, N. R., 2001. An Agent-Based Approach for Building Complex Software Systems. *In Commun. ACM*, Vol. 44, No. 4, pp.35-41.

Johnson, R. E., 1997. Frameworks = (Components + Patterns). *In Commun. ACM*, Vol. 40, No. 10, pp.39-42.

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

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

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