END-USER-INITIATIVE APPROACH FOR TRULY USEFUL E-GOVERNMENT SYSTEMS

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

ABSTRACT

Applications for e-Government should be developed based on user-centered design. In the late 1990s, one-stop-service by governments became popular. In Japan, the e-Government service was promoted by priority policy programs since 2001. However, there have not been sufficient achievements. Many ministries in the central government developed Web applications with a lack of usability. On the other hand, local governments developed applications slowly because of lack of funding. The main problem is lack of end-user's point of view. This paper focuses on Web applications supported by the government because citizens will obtain a lot of benefit in daily life by using them. Such applications should be developed by business professionals themselves since applications must be developed agilely and modified frequently in order for a local government to support Web applications of high usability for citizens in a timely fashion with low costs. The effectiveness of this approach is confirmed by feasibility studies.

KEYWORDS

Usability, e-Government, application framework, Web service integration, end-user computing, form transformation

1. INTRODUCTION

The number of end-users using the Internet has been increasing. Usability of applications has become important. In particular, applications for e-Government should be developed based on user-centered design.

In the late 1990s, one-stop-service by governments became popular. For example, the US postal service's site supported many procedures together which followed a procedure of an address change. In Japan, e-Government was promoted by priority policy programs called "e-Japan" since 2001. However, such achievements have not been sufficient. Many ministries in the central government developed similar systems individually with sufficient budgets, many of which systems were not sufficiently used because of a lack of usability. On the other hand, local governments developed applications slowly because of lack of funding.

The main problem is lack of end-user's point of view, that is, why, what and how the application should be developed. In this paper, we focus on Web applications supported by governments because citizens will be benefited in their daily life by using them. Such applications should be developed by business professionals themselves since applications must be modified frequently. The following challenges are needed for the promotion of e-Government:

(1) Usability
(2) End-user-initiative development by business professionals
(3) Agile and low cost development and maintenance

This paper describes the past and present status of e-Government and end-user-initiative approaches in Section 2 and several solutions such as the agent technologies for (1) and the application framework and Web service technologies for (2) and (3) in Section 3.

2. PAST AND PRESENT STATUS

2.1 Issues in Japanese e-Government

In order to promote measures for forming an advanced information and telecommunications network society expeditiously and intensively, the Strategic Headquarters for the Promotion of an Advanced Information and Telecommunications Network Society (IT Strategic Headquarters) [9] was established within Cabinet in January 2001. The strategy called e-Japan, aims to make Japan the world's most advanced IT nation within five years. One of four priority policy areas is the realization of e-Government, which handles electronic information in the same manner as paper-based information, by fiscal 2003, and even expedites digitization of citizens and businesses widely.

However, e-Government was not sufficiently achieved. In fiscal 2001 and 2002, the more than 70 billion dollars budget was spent and how to use the tax was criticized. For example, the registration system of new cars started in Dec. 2005 and was used for 11,175 cars among about 1.54 million cars in 2007. The rate of use was only 0.7%. The filing system for a passport started in 2004 and stopped in 2006 because only 133 passports were issued. This implies that the cost for one passport is more than 100 thousand dollars.

In 2006, the new IT reform strategy started for realizing a ubiquitous and universal network society where everyone can enjoy the benefits of IT. In the pursuit of IT structural reform capabilities, one of the several areas responding to social issues that should be resolved is the world's most convenient and efficient e-Government for handling of 50% or more of all filings online and creating a small and efficient government. Furthermore, as for the development of IT Infrastructure, to realize of an IT society without digital divide, one of the targets is adoption of universal designs by promoting IT development that everyone can use safely and enjoy the benefits of IT.

However, it was insufficient to realize these goals of the second stage until now as well as the first stage of e-Japan. Recently, the Board of Audit of Japan conducted a survey of use rate of 49 developed systems and disclosed that the use rate of the 12 systems is lower than 10%. The use rate of seven systems among them is lower than 1%. Many systems were developed without needs.

In July 2009, the i-Japan strategy 2015 was presented by the IT Strategic Headquarters. As for e-Government, one of the targets is the promotion of one-stop-service with cloud computing. The e-Government usability guideline detailing items to be completed in the planning stage, design and development stage, the running stage and the evaluation stage respectively were also presented.

2.2 End-user Computing

There are several approaches to the end-user-initiative development. That is, the UI-driven approach makes it possible to develop applications easily for the UI-centered front-end subsystems. 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 the 80's. Some papers describe definitions and classifications of EUC [4] or the management of EUC [1]. A paper summarized the trends of end-user development without the assistance of IT professionals [14].

There are some other works related to EUC. In the programming field, technologies for programming by example (PBE) [11] 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 [13] 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, especially for e-Government. The user's intention is 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, in this paper, the three-tier architecture which is composed of the user interface (UI), business logic and database (DB) is adopted since it has been popular recently. In our studies, application frameworks, visual modeling tools based on components and form transformation tools for Web service integration were developed for EUC [2][3].

For example, there have been 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 [10]. An intelligent e-forms development and maintenance environment and associated framework are delivered. Another one is performed at the local government of Nagasaki prefecture in Japan [8]. The staff of business professionals designed and described the user interfaces without IT professionals' assistance. Furthermore, while the staff 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 thousand 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.

3. END-USER-INITIATIVE APPROACH

3.1 Basic Concept

Our approach as to how to make Web applications is shown in Figure 1. The left side implies the abstract level. The right side implies the concrete level with the related technologies. The business model at the business level is proposed by end-users who are business professionals. Then, at the service level, the domain model is constructed and the required services are specified [5]. That is, the requirement specifications of the application for the business model are defined. At the software level, the domain model is implemented by using components to be combined.

In this approach, there are two technological gaps, these being, 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 and application frameworks [6] based on CBSE(Component-Based Software Engineering). In particular, our previous studies verified the effectiveness of application framework technologies by development of service counter frameworks.

Conversely, the semantic gap is bridged by form based technologies and agent technologies. That is, an intelligent form with an agent which learns business logic, was introduced. In our recent studies on end-user initiative development, it is considered that a simple form is sometimes better for end-users.

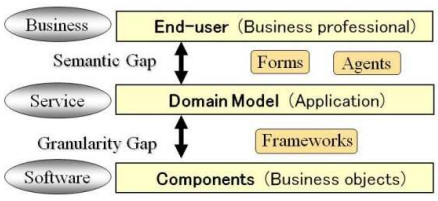

Figure 1. Technologies for bridging the two gaps.

We direct our attention to an application system for a service counter. Of course, many kinds of service counters have already been put into practical use on the Internet. However, these systems must have been developed by IT professionals, not by end-users, and are expensive. Furthermore, although the business professionals require frequent modification of specifications in these systems for service upgrades, it is difficult to modify software quickly because the business professionals need to rely on the IT professionals instead. This method causes budget and delay issues for practical use. Therefore, our goal is the development and maintenance by end-users who are business professionals in the local government.

In the remainder of this section, technologies on application framework, intelligent navigation by multi-agents, Web service integration and form-to-form transformation are described.

3.2 Application Framework

The first version of the framework for a service counter was developed for a library system. The framework for a service counter includes the browser and the server but does not include the workflow system and the DBMS. The 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 a service counter, these five forms satisfied the system requirements. As a result, it was confirmed that the UI-driven approach was suitable for such a system. The browser for system definitions by end-users is shown in Figure 3 and will be explained in 3.3.

In this framework, however, there were only five forms, and interaction among these forms was minimal because services by these forms are independent of each other. Therefore, the framework for reservation was developed 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 applied to a meeting room reservation system for our department in practical use. This application is similar to a Web application for reservation of facilities in a local government such as meeting rooms or athletic fields. This framework uses open source framework for building Web applications, Struts, and uses JavaServer Pages (JSP) for dynamic Web pages. 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 2,250 lines, including the XML document, were reused as frozen spots.

In this application, a lot of forms were developed since the usability was important. However, these forms required 550 lines of unreusable code 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 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 users is shown in Figure 2. 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 a requirement definition. 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. Most of the 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.

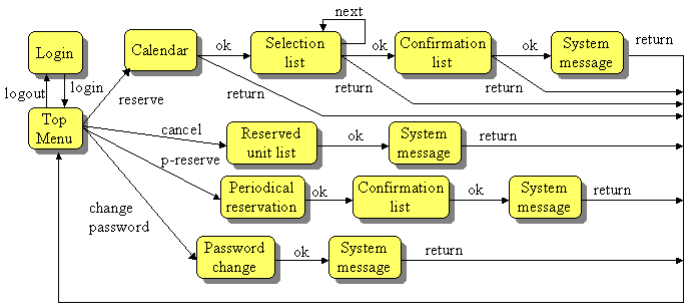

Figure 2. A UI transition diagram for reservation.

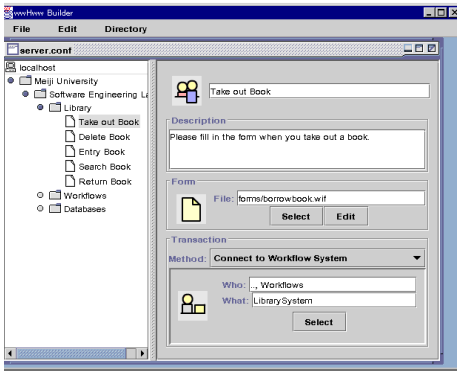

Figure 3. The browser for the system definitions by end-users.

3.3 Intelligent Navigation

The semantic gap between the domain model and end-users is bridged by multi-agent systems 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 service counter mentioned before. Agent facilities were added to the previous architecture. Electronic form processing is navigated by agents in both the client terminal and the server. 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. Business professionals can pass their expertise onto their agents. Then the agents can assist clients in filling out forms and checking written forms. This navigation facility is very important because citizens at home cannot ask business professionals at service counters about how to fill in the forms.

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 used 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, and the semantics of the forms are defined in RDF style also.

The form-based agent communication language was designed for communication between client agents and expert agents. This language is different from FIPA ACL [7] and is very simple because the communication is performed by sending or receiving written forms based on the concept of "one service = one form." The basic form is shown as (who, what, how, which). The who-parameter of a message receiver object implies a service counter where a written application is sent to. The what-parameter of a method name in the receiver object implies the title of the application form. The how-parameter of a set of parameters for a method invocation implies contents of the application form. The which-parameter of a message number implies a receipt number stamped on the received written application.

The states of values of these parameters affect the semantics of the message. If a value of a message parameter is unknown, the message implies an inquiry about the parameter. The semantics is quite different from conventional object-oriented programming languages because it is illegal not to determine the message receiver, the method name and the actual parameters for conventional message sending. This extension in this paper, however, produced an attractive effect. The actual end-user interface for filling in the form is different from the basic form which is the internal representation in the system. Examples of requests are given in the basic form for convenience, where the following notations are used:

a, b, ... : Parameters with known values.

?a, ?b, ... : Inquiries about the parameters with known values

x, y, ... : Parameters without values.

?x, ?y, .., : Inquiries about the parameters themselves

In the basic form, a word in which the first character is '?', is used for two different kinds of inquiries. The second use such as ?x and ?y implies requesting all possibilities. Several examples are given as follows:

(1) (?x, ?y = (a list of keywords), ,) : The list of titles of all application forms which relate to the list of keywords, is displayed with the names of service counters receiving them. The system retrieves forms, the titles of which include the keywords or help messages of which include the keywords.

(2) (a, ?b,,) : The explanation on the application form, b, to be sent to the service counter, a, is displayed.

(3) (a, b, ?x,) : The application form, b, to be sent to the service counter, a, is displayed. How to fill in the form is taught by an expert agent. Some typical items are filled in automatically by the client agent.

(4) (a, b, c, x) : The written application, b, with the contents, c, is sent to the service counter, a. A message number will be assigned to the variable, x, by the service counter receiving the message.

As an example of operations, let's consider a citizen who wants to get permission for parking at city hall and suppose that he or she does not know where or how it can be gotten. Figure 4 shows operations that can be done at home. The operations and the basic form to be sent at each step are described as follows:

(a) "City.*" is the input to the who-column and the keywords of "parking" and "Hall" are the input to the what-column in the initial screen, and then the basic form (City.?x, ?y = ("parking" "Hall"), ,) is sent. That is, the character, '*', implies a wild card and is transformed into '?x' of the basic form.

(b) The system displays “CityHall Sec.” in the who-column and “Parking form” in the what-column. By clicking on the value area in the what-column, the basic form (“CityHall Sec.”, ? “Parking form”, ,) is sent. The first character, ’?’, of the what-parameter with a known value implies the help message request.

(c) The system displays the help message on the parking form. After clicking the value area of a blank in the how-column, the basic form (“CityHall Sec.”, “Parking form”, ?z,) is sent.

(d) The system displays the application form. After filling in the two blanks for the ID no. and the date, the basic form (“CityHall Sec.”, “Parking form”, (“ID no.” = “M6507”, “date” = “June 26”, w) is sent. In equations of the how-parameter, the left sides are column names in the form and the right sides are input values of the columns.

(e) The system displays the message number in the which-column, while the value is assigned to the variable, w. Then the system terminates by clicking the close button.

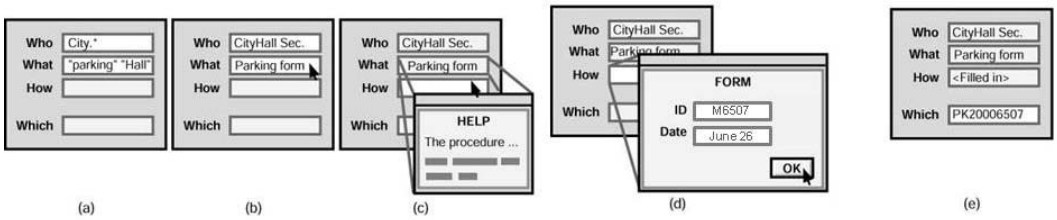

Figure 4. An example of operations based on the form-based agent communication language.

3.4 Web Service Integration

In order to apply our solutions to practical Web service integration [12], we select an application which generates individual examination schedules for each student [3]. In our university, actually, the examination schedule is notified on bulletin boards. Conversely, the university supports the individual portal sites for each student. The student visits the portal site which displays the individual timetable for classes. Here, we can foresee that the XML-based Web service for distributing the examination schedule to students will be realized in the near future. In our experiment, an actual examination schedule is transformed into an XML document manually. As for the individual timetable for classes, an actual HTML document is extracted from the individual portal site for each student. The target application generates an individual examination schedule for each student from the individual timetable for classes and the examination schedule.

The system is composed of three subsystems of HtoX, XtoX and XtoH. The HtoX transforms the individual timetable for classes in HTML into an XML document by using the wrapping technology. This HTML document includes information about subjects for each student, that is, subject names, instructor names and class numbers. The XtoX merges this XML document and the examination schedule in XML into the individual examination schedule in XML format for each student. The examination schedule includes information about subject names, dates and periods, and room numbers. The XtoH transforms this output in XML into an HTML document which can be displayed on the Web browser of each student.

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 only to modify the system when the inputs change. The procedure is described in a script language in order to solve this problem. Furthermore, a visual tool supports the end-user.

This procedure is described in XSLT(XSL Transformations). In this procedure, only the XSLT stylesheet is dependent upon the individual application, and must be described by the end-users. The other parts are automated. It may be difficult, however, for the end-users to describe the XSLT stylesheet even though the scripting is relatively easier than programming. This is because the end-user must understand the concepts of Xpath and templates.

We developed a visual tool which generates the XSLT stylesheet. At first, file names are inputted into the text input areas. Then, two input XML documents are transformed into HTML documents in which checkboxes are located at the front of each node, and are displayed. The user selects the parent node of the element to be compared. Then, the user further selects the elements to be compared. That is, in this application, first the subject is selected, and then the subject name and the instructor name are selected. In the third step, the selected elements are displayed as follows:

F0 /examine[1]/subject[n]/name[1]
F1 /examine[1]/subject[n]/instructor[1]
S0 /personaltimetable[1]/subject[n]/name[1]
S1 /personaltimetable[1]/subject[n]/instructor[1]

F0, F1, S0 and S1 are symbols for the corresponding XPaths. The user can define conditions for comparison by using these symbols as F0 = S0 and F1 = S1

This process can be applied to other general domains. For example, let us consider that a citizen wants to reserve a meeting room and a wheelchair together at the Web sites of a local government. Usually these services are supported by different Web sites. One Web site supports reservation of public facilities. The other site supports reservation of rental equipments. Therefore, the citizen must reserve them individually.

These services must be much convenient if they are integrated by using the merging method. In this case, the merging processes as shown in Figure 5. At first, when a citizen specifies a period to be reserved, this information is sent to two sites. Then two lists of available time and date are returned from these sites. Then, a list of times and dates with both the available meeting rooms and the available number of wheelchairs, are extracted by merging the returned lists. After this, the citizen can selects the best one among them.

This merger is extended for dealing with complicated business logic. That is, this basic merger is applied repeatedly in the multistage merger in our experiment of developing a system to advise students on graduation requirements. This system advises a student to take specific subjects necessary for graduation from inputs of both his/her school report and a manual on graduation requirements. The school report shows a list of subjects and the credits the student has already earned. The manual shows conditions for graduation, which are considered as complicated business rules. That is, subjects are categorized into several groups which compose a hierarchy, where each category has individual conditions. Finally, in this application, the seven stages were necessary for the confirmation of conditions required for graduation.

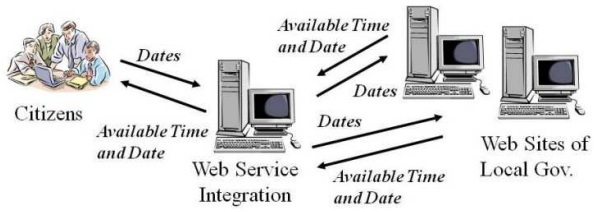

Figure 5. Web service integration.

3.5 Form-to-form Transformation System

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 need not 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 defining this procedure, the form transformations from input forms into output forms is executed [2].

Let's consider a Web site of a lending library supported by a local government, where a citizen can reserve a book. Furthermore, let's suppose that the local government has a delivery service of documents such as certificates or applications for handicapped people. Some citizens may require that the reserved book is delivered to their home by paying a fee. For integration of these two services, business professionals need to define the form-to-form transformation method as shown in Figure 6. This scheme implies the form-to-form transformation from one input into two outputs.

At the definition stage, after the book reservation & delivery form is defined, the transformation from this integrated form into the book reservation form and the delivery form is defined while mapping columns in the input form into columns in the two output forms.

Then at the execution stage, after a citizen fills in the input form for the book reservation & delivery service, the book reservation form and the delivery form are generated. Then the system fills in these two forms in accordance with the definition of the form-to-form transformation and sends the book reservation form to the Web site of a lending library, and the delivery form to a Web site of a delivery service.

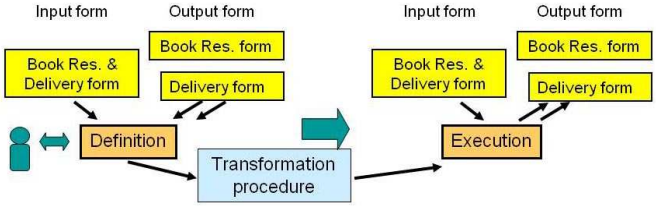

Figure 6. A form-to-form transformation system.

Tools for definitions and executions of the form-to-form transformations were developed. The user interface was implemented in HTML and JavaScript. The generated transformation procedure in XML is sent to the server and is stored there. The interpreter of this procedure in XML was implemented in Java. At the definition stage, the input forms and the outputs form are displayed on the left. The palette with operators, variables and functions is displayed on the right. Whenever a column of forms or an operation item of the palette is clicked on, the order and the operation item are displayed below for confirmation. For example, when the value of the name-column in the input form is copied into the name-columns in the two output forms, the name-column in the input form, the "=" operator in the palette, the name-columns in the first output form and the name-columns in the second output form are clicked on successively.

4. CONCLUSION

The effectiveness of some end-user-initiative approaches for e-Government was confirmed by feasibility studies. In particular, these technologies are indispensable in order that a local government supports Web applications of high usability for citizens in a timely manner with low costs.

REFERENCES

[1] Brancheau, J. C. and Brown, C. V., 1993. The Management of End-user Computing: Status and Directions, *In ACM Computing Surveys*, Vol.25, No.4, pp. 437–482.

[2] Chusho, T. and Yagi, N., 2008. Modeling by Form Transformation for End-user Initiative Development, *Proceedings of IEEE Computer Software and Applications Conference (COMPSAC 2008)*, pp.331-334.

[3] Chusho, T. et al, 2006. A Form-based Approach for Application Development by Web Service Integration, *Proceedings of IADIS Applied Computing 2006*, pp.600-605.

[4] Cotterman, W. W. and Kumar, K., 1989. User Cube: A Taxonomy of End Users, *In Communications of the ACM*, Vol.32, No.11, pp. 1313-1320.

[5] Elfatatry, A., 2007. Dealing with Change: Components versus Services, *In Communications of the ACM*, Vol. 50, No. 8, pp. 35-39.

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

[7] FIPA, 1999. Agent Communication Language, FIPA Spec 2-1999, Draft ver.0.1.

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

[9] Japanese IT Strategic Headquarters, http://www.kantei.go.jp/foreign/policy/it/index_e.html.

[10] Lepouras G. et al, 2007. Domain Expert User Development: the Smartgov Approach, *In Communications of the ACM*, Vol. 50, No. 9, pp. 79-83.

[11] Lieberman, H. (Ed.), 2000. Special Issue on Programming by Example, *In Communications of the ACM*, Vol.43, No.3, pp.72-114.

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

[13] Ozsoyoglu, G. and Wang, H., 1993. Example-Based Graphical Database Query Languages, *In IEEE Computer*, Vol.26, No.5, pp.25-38.

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