January 11, 2006
RT0638
Computer Science; Human-Computer Interaction; Web Service 10 pages

Research Report

Easy SOA: Rapid Prototyping with Web Services for End Users

Takayuki Yamaizumi, Takashi Sakairi, Masaki Wakao, Hideaki Shinomi, Samuel Adams

IBM Research, Tokyo Research Laboratory
IBM Japan, Ltd.
1623-14 Shimotsuruma, Yamato
Kanagawa 242-8502, Japan

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Easy SOA: Rapid Prototyping with Web Services for End Users*

Takayuki Yamaizumi
Tokyo Research Laboratory,
IBM Japan
1623-14 Shimotsuruma,
Yamato,
Kanagawa, Japan
zumi@jp.ibm.com

Takashi Sakairi
Tokyo Research Laboratory,
IBM Japan
1623-14 Shimotsuruma,
Yamato,
Kanagawa, Japan
sakairi@jp.ibm.com

Masaki Wakao
Yamato Software Laboratory,
IBM Japan
1623-14 Shimotsuruma,
Yamato,
Kanagawa, Japan
wakao@jp.ibm.com

Hideaki Shinomi
Yamato Software Laboratory,
IBM Japan
1623-14 Shimotsuruma,
Yamato,
Kanagawa, Japan
shinomi@jp.ibm.com

Samuel Adams
IBM Research
4400 Silicon Drive
Durham, NC 27713
ssadams@us.ibm.com

ABSTRACT

Wiki and blog which enable end users to do their works only with a Web browser, since those tools do not require end users to learn some special skills about HTML. Some of end users have also involved in "End-user programming", by writing some tools to change a behavior of an application to make their personal works more effectively. However, these tools are too difficult to be shared with other people generally, because a flexible application tends to be difficult to learn. Thus, an application development tool for end users should be easy to learn. It should not require neither to install more special additional softwares nor to add more special configurations on their personal computers.

This paper describes a rapid prototype tools for SOA based on Ad hoc Development and Integration Environment for End Users (ADIEU). ADIEU is a development tool for end users and works on a web browser by communicating with ADIEU server without installing any special development environment on a personal computer. With ADIEU, end users can prototype their application rapidly by placing some cards onto a development environment constructed on a Web browser and adding some connections between the cards. We also propose a prototype development model for Service Oriented Architecture (SOA) as well as Web applications and Web Services with ADIEU. The proposed development model consists of two parts: (i) End users import WSDL files from Web Services to represent methods defined in Web Services as cards on ADIEU environment. (ii) They place some cards and define the relationship between them on a Web browser. Because this environment model is quite simple and easy to be understood by end user, it will also encourage end users to adopt the SOA (Service-Oriented Architecture) by combining Web services with simple interfaces in the distributed environment.

Categories and Subject Descriptors

H.4.m [**Information Systems**]: Miscellaneous; D.2 [**Software**]: Software Engineering; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Service Oriented Architecture, End-User Programming

Keywords

Web Services, Service Oriented Architecture, Development Tool, End-User Programming

1. INTRODUCTION

Many end users edit and update their document with tools such as wiki[5, 6] and blog[4]. They also use these tools to communicate with each other only with a Web browser, since those tools do not require end users to learn some special skills such as HTML. Furthermore, they do not need to install any server application if they decided to use wiki and blog applications provided by an Internet service provider.

Some of end users have also involved in "End-user programming", by writing some tools to make their personal work more effective. These tools are written in Visual Basic, Excel Macro, and the other scripting languages. However, most of those tools are very difficult to be used and to be maintained by other people, since most of the end users write their applications without write their specifications and they often have to learn about applications on

*(Produces the WWW2006-specific release, location and copyright information). For use with www2006-submission.cls V1.4. Supported by ACM.

WWW2006, May 22–26, 2006, Edinburgh, UK.
.

which the tools would be run. Thus, the other users have to examine how the application works by running it and by observing how it works. Additionally, when end users ask application developers to develop some applications for their business with high quality and excellent user interfaces, end users often feel difficulty to explain to application developers what they want to direct a computer through the application, because their specification has become too large and too complex to understand for end users in general. This communication gap becomes increasingly prominent in web application development, because web application development usually starts from ill-structured and vague requirements[2, 3]. This communication gap also may cause a delay in development, which becomes unacceptable because developers are required to develop and to deliver applications in short period of time. Consequently, a collaboration method which involves end users in prototyping is needed to develop applications which end users want faster.

First, this paper describes Ad hoc Development and Integration Environment for End Users (ADIEU)[1]. ADIEU is a development tool for end users and works on a web browser by communicating with ADIEU server without installing any special development environment on a personal computer. as shown in Figure 1 With ADIEU, end users can

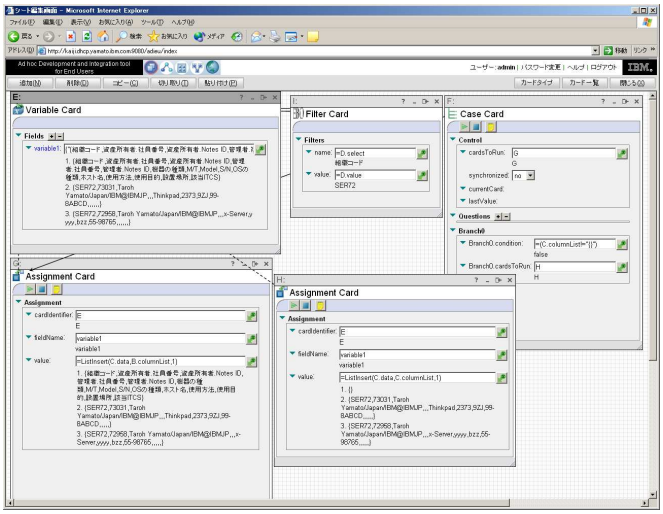


Figure 1: Sample snapshot of ADIEU

prototype their application rapidly by placing some cards into a sheet which is displayed in a web browser and represents an application. Each card has some input fields, so that relationships between the cards can be also defined through the fields. ADIEU also has a functionality to import external Web Services to create more cards to represent the imported Web services to enrich its assortment. Since Web service method can be imported by reading WSDL (Web Services Definition Language) files [19] and their interface is automatically generated on a card, connections between a generated card and the other cards are defined.

Second, we also propose Easy SOA, which is a prototype development model for Service Oriented Architecture (SOA) as well as Web applications and Web Services with ADIEU. SOA is an emerging methodology and a framework to construct an enterprise system mainly with Web services, but it can hardly understand and confirm for end users how it works. The proposed development model consists of two parts as follows:

- End users import some WSDL files from Web Services to represent Web Services methods as cards on ADIEU environment.
- They place some cards and connects between them on a Web browser.

Because this environment model is quite simple and easy to be understood by end user, it will also encourage end users to adopt the SOA by combining Web services with simple interfaces in the distributed environment.

The rest of this paper is organized as follows: Section 2 describes the motivations for this paper. ADIEU is introduced in Section 3, followed by an explanation how ADIEU imports external Web service in Section 4. Section 5 illustrates our "Easy SOA" architecture and a development scenario based on SOA, followed by a "Historical Stock Quote in any currency units" development example. Finally, in Section 6, we discuss about our work and related works. The conclusion and future works are summarized in Section 7.

2. MOTIVATIONS

Many end users write some tools to make their personal work more effectively. However, most of those tools are very difficult to be maintained by other people, since an useful application tends to be difficult to learn and they cannot be expected to tolerate[10]. Additionally, when end users ask application developers to develop some applications for their business, such as a system to handle transactions, end users often feel many difficulties to explain to application developers what they want to direct a computer through the application, because their specifications are too large and too complex to understand for end users. As a result, developers illustrate a different specification as end users want to use. This communication gap and misunderstanding often mislead application developers to construct a different system from what end users had asked them to construct. The developer would try to modify the system to conform with end users' specification in some times. This iteration may also cause a delay in development, which becomes unacceptable since developers are asked to develop and to deliver more complex applications than those of previous version in a shorter period of time. For this reason, some prototyping tools are needed to encourage end users to participate in a system design.

Therefore, many software tools have developed for end user to design and to develop applications [31] by selecting items from graphical menu, placing text fields on a window, and by doing the other various graphical operations. However, to use these applications, end users have to install these applications on their personal computer and to learn how to use them. Some of them feel difficulies about the installation and usage. If an end user ever learns how to use them successfully, it is more difficult to find an usable component or user interface because those components or user interface must be developed by an user who has more skill, so that an end user must find such users, that is almost impossible. Therefore these tools cannot be adopted as development tools for end users. Consequently, there is

still some technical challenges in delivering a graphical development environment for end users and non-professional programmers. The card-based programming model is one of the solution, because it breaks down a traditional procedural programming language typically rendered and lines of text into separate interfaces for each effective function or "line of code". These separate interfaces become a kind of Integrated Development Environment (IDE) for each statement in a program, which provide a number of advantages in providing statement-specific guidance, help, and features such as activity logging and control flow-based execution tracing. Since each card represents a statement in the programming language of ADIEU, new cards, whether developed in ADIEU itself or in Java™,[1], effectively provide the ability to extend and reshape the programming language to better fit the problem domain which users concern.

On the other hand, some Web services are available to public through the Internet[12, 18, 13], although the number of public Web services has not increased dramatically. They are quite useful for skilled developers who can implement the interfaces for Web services with stub code because these Web services prevent them from "reinventing the wheel". Likewise, the public Web services can be accessed by end users who have neither any programming skill nor deep understanding of their specifications through the interfaces generated from WSDL files [19, 22]. However, since some of the Web services require a specifying a complicated data structure for the input data, users must have a deep understanding about the specifications of the Web services if they want to handle complicated data, and this should not be required to end users. Web services have become the open standard and can be used widely. Once a development tool which can import a WSDL file and utilize Web service methods described in the WSDL file, an user can develop an application with less cost and time to learn. ADIEU can provide simpler interfaces for Web services by representing a Web service method as a card and its data structure is displayed as a tree view.

Another motivation comes from SOA. SOA is a framework and an emerging methodology to construct enterprise systems [8], as well as to integrate the existing systems into a larger system. Web services now play an important role in SOA because most SOA systems will be constructed by combining Web services. However, since current tools to construct SOA needs huge resources to a personal computer and lacks a functionality to examine how a model defined on a tool actually works. Thus, rapid prototyping tool for SOA, i.e. Easy SOA will become important to construct an IT infrastructure. Users can define the relationships between any points in data structures which are used as inputs and outputs of a Web service even if their data structures are very complicated, as shown in Section 5.2.

3. AD HOC DEVELOPMENT AND INTEGRATION ENVIRONMENT FOR END USERS (ADIEU)

In this section, we will show the overview of ADIEU. Then we introduce a typical scenario to develop a Web application and a Web Service with ADIEU.

[1]Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

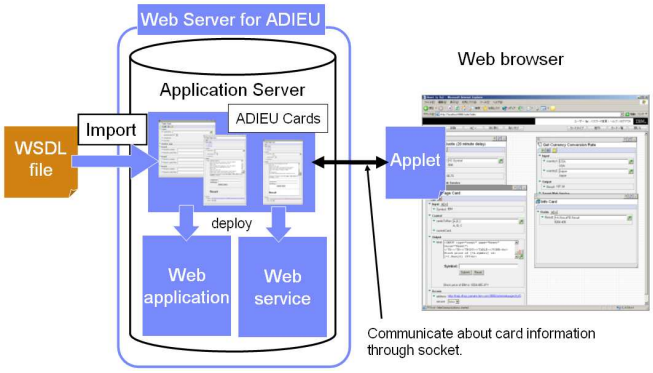


Figure 2: ADIEU architecture

3.1 ADIEU Overview

ADIEU is the programming environment which operates within the Internet Explorer Web browser. Its screenshot example is shown in Figure 1. ADIEU can be installed on WebSphere Application Server or Apache Tomcat. Users can develop and deploy Web applications and Web services without installing any software onto their personal computers. By using this tool, end users can develop Web services and Web applications without any programming knowledge, although this environment is written in Java, JSP, and JavaScript[15]. An application on ADIEU environment consists of two parts: sheet and card. These parts correspond to an application and a logic, respectively. Users can develop these applications by using collections of cards, each of which acts like single-function applications in a form-based, desktop-like environment. The data fields in cards can be used like cells in a spreadsheet and can contain either data or an expression that determines the data at run time. Cards can also run other cards; this capability provides the basic flow control necessary for programming concepts such as decision branching, sequences, and loops. With this capability, ADIEU can be regarded as a prototyping tool, because an user can examine how their application logic actually works on a Web browser.

Figure 2 shows the architecture of ADIEU. We developed user interfaces including cards, sheets, and the other components in Dynamic HTML(DHTML), JavaScript, JSP and Java. We also developed a persistent socket-based communications channel between the client interface and the ADIEU server. When users launch the ADIEU environment, an applet which creates this communication channel between a Web browser and the ADIEU server is download from the Web server to client automatically, so that an user do not need to take an extra work for that. This implementation provided a high degree of interactive response for the interface without requiring full redisplay of the programming environment. It also allows the user to interactively and in parallel execute multiple card actions.

Users can access their applications as Web applications by clicking the link on Web Page Card. With Web Service Card, users can also their applications as Web services without any special operations to deployment. Because all of informations about ADIEU environment stored in a Web application server, Users can develop Web application and Web services from any modern personal computers with a

Web browser.

3.2 Development Scenario Example

Suppose we asks end users to develop a Web application and a Web Service which calculates a summation of two numbers. End users can develop their application with adieu by following steps.

1. Users specify the URL which points the ADIEU environment.

2. Users creates a sheet for this application and give it a name.

3. Users open the sheet and place two Variable Cards. The ADIEU environment assigns their card IDs, such as A and B. End users can create Variable Card by selecting from menu. Variable Card has only one field (`variable1`) when the card has just been placed on the sheet.

4. Users place Web Page Card on the sheet and type a HTML snippet as shown in Figure 3. The ADIEU environment assigns C as its card ID. To refer a value which is held by a field in the other card, User insert a special anchor to point the field on the card in the following format (1):

```
<h1>Very Simple Calculator</h1>
<hr>
[=A.variable1] plus [=B.variable1] equals to
[=A.variable1+B.variable1].
```

Figure 3: A HTML snippet example for Web Page Card

$$\texttt{<}\textit{card id}\texttt{>.<}\textit{field name}\texttt{>} \quad (1)$$

For end users' convenience, this anchor can be inserted by clicking the "insert variable" button at the right side of a text area (Figure 4), and the field name can be selected directly from the menu list. In the example shown in Figure 4, `A.variable` is inserted into the html field.

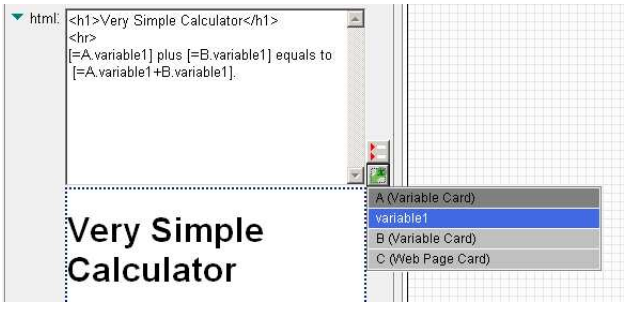


Figure 4: "Insert variable" button and a menu to select field

5. Users may confirm here how this application works by typing some variables in `variable1` fields on Variable Cards. The variables in HTML on Web Page Card are changed, accordingly.

6. Users click the form button to insert a form into html field and to create `variable1` and `variable2` input fields. This operation may be quite easy for end users, while expert users may type the same form manually.

7. Users set the values in `variable1` field on Variable Card A and B to `=C.variable1` and `=C.variable2` respectively. and type `A,B` to the cardsToRun field on Web Page Card C. At this step, user can finished the development of a calculator Web application.

8. To test this application, users simply click the hyperlink in address field on Web Page Card C to start a Web browser shown in Figure 5. Users can calculate with two numbers through this web page.

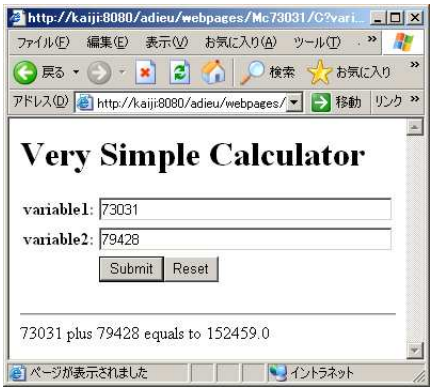


Figure 5: Running a Web application developed on ADIEU environment

From this Web application, a Web service can also be developed by adding Web Service Card on this sheet, and by adding some fields and configurations as follows:

- Users add Web Service Card, which ADIEU assigns card ID D.
- Add `variable1` and `variable2` fields which can store one integer value for each field by clicking the "Add field" button at the top of the "Input" fieldset view.
- Add `answer` field which can store an integer value by clicking the "Add field" button at the top of the "Output" fieldset view.
- Replace the values in `variable1` field on Variable Card A and B with `=D.variable1` and `=D.variable2`, respectively.
- Type A, B to the cardsToRun field on the Web Service Card D.

In general, Web service developers confirm the behavior of this Web application by creating stub application for it, by generating a Rapid Application Development (RAD) tool to generate HTML test files. Instead of those methods, ADIEU

users can confirm the behavior of this Web service by itself. Users can import the WSDL file which defines the Web service by selecting "Import New Web Service" menu and by typing the URL of the Web service, which is displayed in the wsdl field on Web Service Card, as shown in Figure 6. This Web service can be tested with the generated card (card *E* in Figure 6) by the import. This functionality can be also

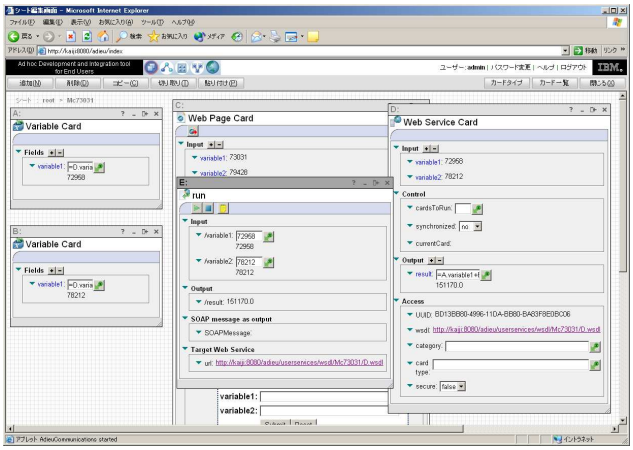


Figure 6: Specifying the URL of WSDL in the ADIEU environment itself

used to import external Web Service. We will discuss it in the next section.

4. IMPORTING WEB SERVICES INTO ADIEU

The cards which are available on an ADIEU environment can be divided into two types. The first one is the type of the original built-in cards, such as Variable Card, Web Page Card, Web Service Card and the others. These cards have one Java class for each card to handle events and data related to them. However, it is very hard to develop this type of cards even for a developer. It will only be able to represent very limited logic, even if these kind of card can be developed. The second one is the type of the cards whose specifications are determined when an ADIEU environment reads WSDL files on remove Web application servers. Users can use these card to compensate for the gap between the original functionality and users' typical requirements for their tools. User can import external a Web service through an ADIEU environment by specifying an URL of WSDL file, as shown in Figure 6. In this section, we discuss these generated cards with a sample Web service. This Web service has BankInfo, UserInfo and AccountInfo classes and BankInfo class has five methods which are exported as Web service methods, as illustrated in Figure 7.

4.1 Card Generation Example

ADIEU generate one card for each Web service method. In the Web service example as illustrated in Figure 7, five cards are generated, such as addAccountInfo card, addUserInfo card, getUserInfo card, getUserInfoList card and setUserInfo card. These generated cards completely can be used in the same way as the original cards. For example, users can refer some fields in these generated cards from the

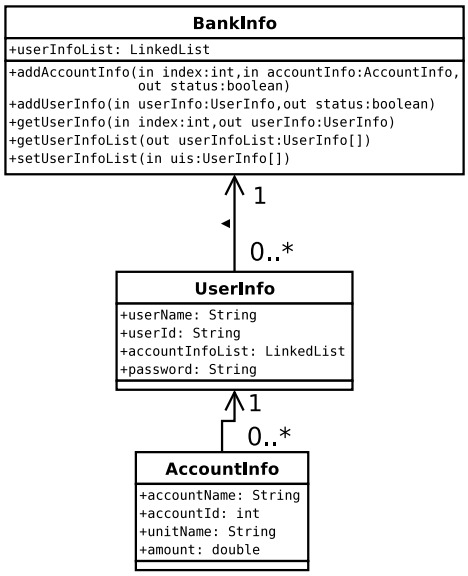


Figure 7: Class diagram of a Web service sample (internal operations are omitted)

other card, and vice versa. After ADIEU imports a WSDL file which includes the definition of Web service methods, it parses the WSDL file to extract the data structure, and finally generates the generic treeview-like interfaces as shown in Figure 8.

Each cards have treeview-like interfaces with input fields to help the understanding of an end user about the data structure, because Web service methods generally handle complexType data as their inputs and outputs. The number of fields on the card can be changed if the data include more than one array of complexType data by clicking the "Add element" button or "Delete element" button on the card. As a result, users can easily integrate external Web services and integrate them into their application. Users can also avoid additional and hard works which often result in reinventing the wheel.

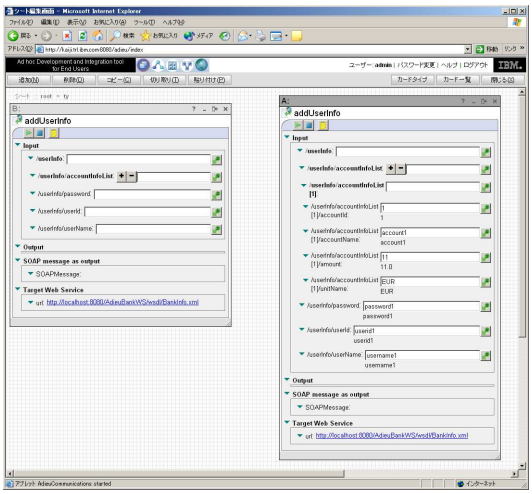


Figure 8: A screenshot example of generated cards

4.2 Support for complexType Data

In general, Web service methods are invoked with input data which are too complex for users to handle[28, 29, 30], because they must build SOAP [20] messages in the format which it is acceptable for a Web service. Output data from a Web service method, written in the SOAP format, are also too complex for an user to handle, while raw SOAP messages are useful for developers and programmers, because they can analyze to debug their software with that message. Accordingly, users need some tools which can support to handle both inputs and outputs data.

4.2.1 SimpleType and complexType

The format of a SOAP message is defined in a WSDL file in the XML schema format[21]. Developers must classify the data into two types; complex type and simple type which are called complexType and simpleType in the specification of XML schema, respectively. ComplexType data include another complexType data or simpleType data, while simpleType data include neither complexType data nor another simpleType data. If a data structure defined in a WSDL file is represented as a tree graph, this tree graph satisfies the following theorems[27].

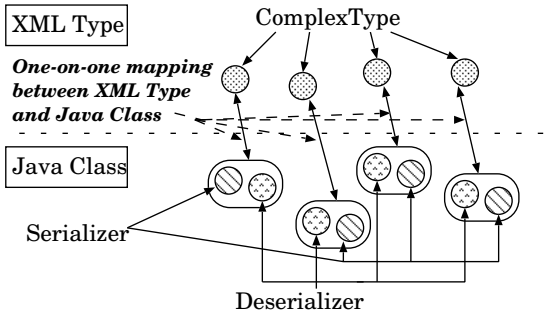
THEOREM 1. *Data in a stem node must be complexType data.*

THEOREM 2. *Data in a leaf node must be simpleType data or complexType data which have no child node, i.e. complexType data including no child data.*

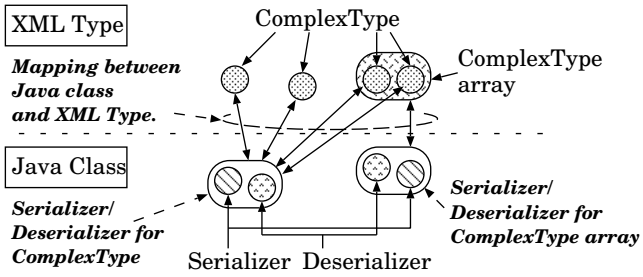
For this reason, we propose to display the data structure as tree view, because users can easily distinguish stem nodes which hold complexType data. Additionally, we define that an array of complex type consists of two types of stem node; one is an "element node" which is a stem node representing an element of an array and included by a "parent node", the other is a "parent node" which represents an array of complexType itself. Despite this definition, the above theorems still hold true if these nodes for an complexType array are regarded as complexType data. By this definition, an user can know a data type which is represented as a node.

4.2.2 Serialization and deserializaion model

Because there are some differences between the data models of all types of SOAP messages, which are RPC/encoding, RPC/literal and document/literal, although the tree view model we have proposed is easy for an user to handle, We have implemented a serializer and a deserializer to fill these gaps on the top of Apache Axis version 1.1 implementation[24]. The comparison between ADIEU's serialization and deserialization model and Apache Axis' original serialization and deserialization model is illustrated in Figure 9. An ADIEU environment handles data with two pairs of serialize and deserializer. The one is the pair which handles any complexType data, the other is the pais which handles any arrays of complexType data. This model is different from Apache Axis' serialization and deserialization model, sincee Apache Axis expects a developer to implement and to assign Java classes to serialize and to deserialize XML type data and store the relationship between Java classes and an XML type, and Apache Axis version 1.1 cannot handle relationships if more than one XML type share the generic serializer and deserializer for complexType data. For this reason,



(a) The model in Axis



(b) The model in ADIEU

Figure 9: The comparison of serialization and deserialization model between Apache Axis and ADIEU

ADIEU's serializer for an complexType array overrides ArraySerializer's serialize() method in Apache Axis version 1.1 to work an array of complexType data.

4.2.3 Data identification in XPath representation

Once a data structure is represented as a tree graph, users can point any data included in complexType data uniquely in XPath format, as shown in Figure 10[14]. To preserve

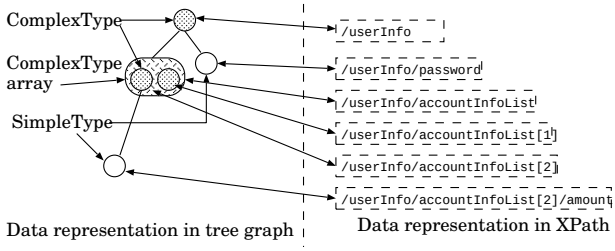


Figure 10: The XPath representation of a data structure in ADIEU

the uniqueness of names between the fields, when there are more than one card on a sheet and they have the same complexType data structure, users spetify a data node in complexType data in the following format (2):

#{<*card ID*>.<*data name in XPath format*>} (2)

Data name in XPath format follows a card ID which includes the data and the concatenated string is embraced by curly brackets to coexists with the other expression such as a mathematical expression by escaping slash characters which are used the sign of division in a mathematical expression. Since format (2) may be long in some cases, these data name in format (2) can be automatically put by clicking "Insert variable" button to select a suitable field in a menu as shown in Figure 4.

Figure 11 shows a prototyping example to work with a Web Service. This application gets the list of users from a Web service through card A and copy a data node in the data structure by pointing `/getUserInfoListReturn/UserInfo[1]/accountInfoList` node on the card A in the format (2) from card B. After an ADIEU environment copies the nodes on card A to card B, fields on card B to which the data are copied are not allowed user to modify the data.

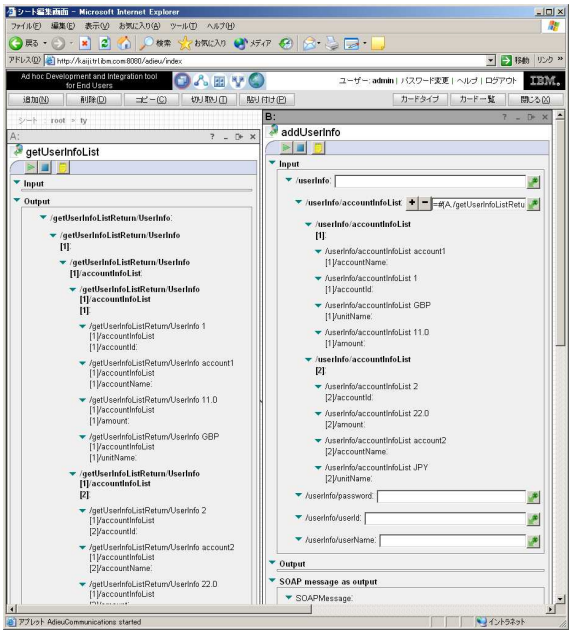


Figure 11: Working with an external Web Service through a generated card

ADIEU can parse SOAP headers and can generate generic interfaces for an ADIEU user, because some Web services authenticate users with data in SOAP header and send a client the result of authentication[12]. SOAP headers can be handled by the same way as SOAP message bodies can be handled.

5. PROTOTYPING ON EASY SOA

Users can build an application in a very primitive style of Service Oriented Architecture, because Web services can be connected in ADIEU on a Web browser, as we discussed in the previous section. We now propose to name this Easy SOA.

5.1 Infrastructure in detail

This architecture is quite simple, as illustrated in Figure 12. Basically, ADIEU itself is not changed as a system, whereas there is a big change in a development scenario. Users import some Web services which provide what they want to integrate their system. They orchestrate the Web service methods by defining some relationships between them with simple sets of mouse operations. Built-in cards which are supplied with an original ADIEU environment mainly help the orchestration of the generated cards. For example, Database Select Card selects some data from a database by keywords which are served by an external Web service and passes them to another Web service.

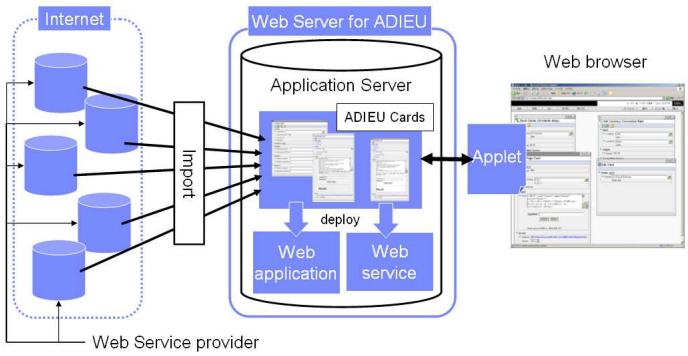


Figure 12: Easy SOA environment

5.2 Development example on Easy SOA: Historical Stock Quotes in Any Currency Units

For example, suppose we have decided to construct Historical Stock Quotes Web application as shown in Figure 13, which can serve a stock price on any past days. This Web application also displays the result in Japanese Yen (JPY) by calling a Web service which can convert the currency unit at the rate on the past day to check the benefit which we gain from our stock. We can construct the service by following steps:

1. Users import a Web service and generate a card which can convert a currency unit on any past day[23], and put a card which represents a suitable method[2]. ADIEU environment assigns A as the card ID.

2. Users place Web Page Card (card B) onto the sheet and create a date field, symbol field and a HTML snippet to display the result on the html field.

3. Users place Assignment Card (card C) to concatenate Month, Day, Year into one string to fit with the input field on card A.

4. Users place Variable Card (card D) which multiplies a foreign exchange rate between U.S. dollar (USD) and JPY by a stock price

5. Users import another Web service and generate a card (card E) which can serve a stock price on any past day[12], and put a card which represents a suitable method.

[2]At the time of writing this paper, because there is a limit in size of string which can hold in a text field on an ADIEU card, We implement Web service to convert data which are a part of XML document string to complexType data

6. Users define the relationships between the cards by typing a sequence of evaluation such as `C,A,E`.

We can examine how it works at each step in development, so that we can easily test the application. Figure 13 shows a typical screenshot after users have been finished the above steps. There are five cards placed in a Web browser.

By clicking the link on a Web Page Card, Historical Stock Quotes application has started, as shown in Figure 14.

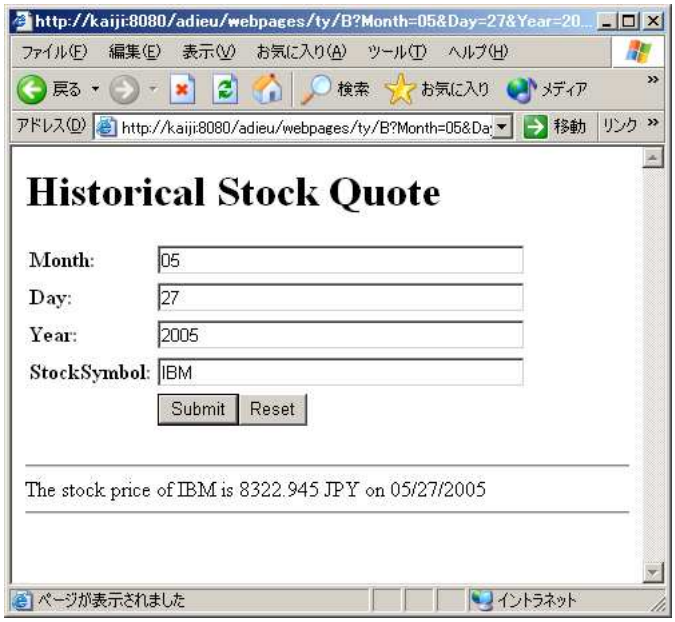


Figure 13: Screenshot of"Historical Stock Quote" Web application

Since the foreign exchange Web service provides many exchange rates at noon on the specified date and the Historical Stock Quote Web service provides a stock price at the end on the specified date, this Historical Stock Quote Web application provides a value in rough figure as a result. However, this value may be still important for end users, because they sometimes want to check whether their stocks are profitable, especially if they hold foreign companies' stocks.

6. RELATED WORK

To develop the enterprise system based on SOA, developers can define Web services' behaviors in business process with the Business Process Execution Language for Web Services (BPEL4WS[16], WS-BPEL[17]). BPEL4WS (WS-BPEL) is one of the core technologies to realize SOA [16, 17]. It is possible to use this to define an executable business process which determines the nature and sequence of Web service interactions. Tool support has been provided to make the definitions in BPEL4WS easier. However, these tools are not usable by end users, and developers must have deep understandings and analyze their business processes and the interaction of the Web services which the developer wants to define in the BPEL file. It is also difficult for developers to examine how it works before the actual system has been constructed. End users can illustrate what they want with ADIEU and can share them with developers in the prototyping and designing phase in development.

In contrast, the most primitive approach to work with Web services are to implement an application to send SOAP messages between Web services [20]. Most of the work, especially the implementations for communication, are eased by a tool which can generate stub code by parsing a WSDL file[11]. However, users still must have some programming skills to work with Web service because the user must code some user interfaces. Users can develop Web service on ADIEU environment without writing any codes.

The WSDL file which defines the interface of a Web service includes an XML schema [21] in which the data structure for Web service is defined. Hence, a Web interface can be generated by reading the data structure definitions by reading a WSDL file on the Web service provider [22] and users can confirm and test the functionality of the methods in the Web services. However, this test service [22] lacks the functionality to develop an application by defining a relationship between Web service method.

Fischer et al. discussed which domain should be selected for end-user development in [26]. They concluded that it is safe to adapt end-user developments to less complex domains. Thus, the end-user development approach can be adapted to Web services which are open to the public, although management and security issues still remain to be solved. End-user programming is recognized as an important issue because an interface should be customized to the need of particular users, although there are likely to be generic structures, e.g., in an email filtering system [25] which could be provided as a Web service, because it can be shared.

Kelleher et al. describe their intensive survey on a Programming Environments and Languages for Novice Programmers[31]. They mentioned that if the population of people creating software is more closely match to the population using software, the software designed and released will better match users needs. However, they also proposed that the people who is familiar with computer science should encourage novice users to program by breaking mechanical barriers and sociological barriers. We can avoid such a time-consuming approach with ADIEU, bacause users do not need any programmings on ADIEU environment.

7. CONCLUSIONS AND FUTURE WORKS

In this paper, we have described ADIEU first, which enables end users to develop Web services and Web applications with a minimal effort to installation, study and development by placing cards on the sheet which is displayed in a Web browser, as well as by defining relationships between cards with minimal mouse operations and keyboard typings. For this reason, we conclude that this tool can be used most effectively in a prototyping phase of development, because end users can participate in the design process of an application development and communicate with developers about their application with their artifacts on ADIEU environment. Collaborative development may contribute a great deal to shorten the development period for application and to result in great satisfaction for end users. Unlike the other development tools for end users, ADIEU can import an external Web service and can generate some cards which enable end users to integrate them into their own application on ADIEU environment. Thus, end users can concentrate

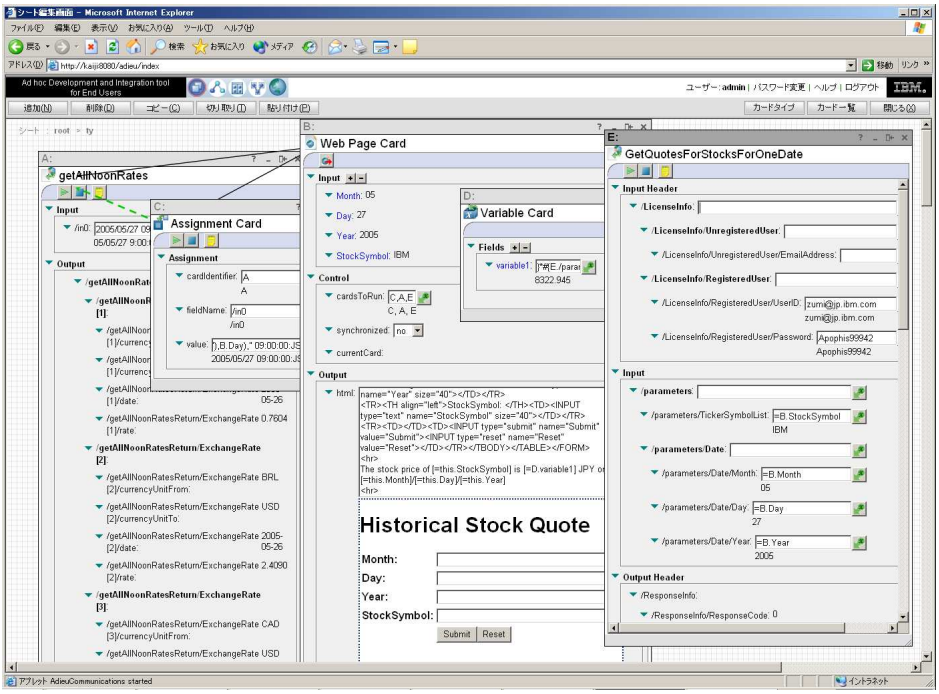


Figure 14: "Historical Stock Quote" Web application

on their own works without reinventing the wheel. Because many Web services can handle a complexType data, generated cards on ADIEU also can handle them, although we have defined two pairs of generic user interfaces, serializer and deserializer for complexType data and an array of complexType data. This approach with generic implementations is completely different from the other existing approaches and liberates end users from the maintenance work of Web client codes for their business.

Second, we also proposed the concept of Easy SOA by connecting the interface between cards which represents Web service methods on a Web browser with simple operations. XPath representation style for complexType data structure will provide a starting point to enhance this representation to a simpler format even if another representation is required for end users.

We have demonstrated two developed scenario, one is the very simple calculator Web application and the other is the Historical Stock Quotes service by orchestrating two Web services which are deployed by the different organizations. This indicates that ADIEU environment can be realize SOA in the primitive manner.

On the other hand, some technical challenges which are likely to be future works still remain. In this paper, we do not discuss about the user interface of ADIEU card, especially whether a tree view is the best interface for complex type data, because a tree view is used to select a few items from the huge number of candidates, although most of Web services may not work if an user gives input data to most of data nodes. We have solved this problem partially by adding a functionality to copy a node between the different complex type data with simple mouse operation. XPath presentation of node is a convenient method for a computer, while it is not for an end user. YAML[32] or another simplified representation may be a solution, however, the investigation will be one of the important future works. Currently, ADIEU cannot deploy a Web application and Web Services. If ADIEU can deploy or export Web application and Web Services, end users and developer will be able to work closer in an application development.

8. ACKNOWLEDGEMENT

Dr. Yuichi Nakamura in Tokyo Research Laboratory, IBM Japan kindly suggested us to write this paper. Dr. Tsutomu Kamimura in Yamato Software Laboratory also carefully reviewed our implementation and suggested us to improve it. We would like to express our deep thanks to both of them.

9. REFERENCES

[1] Ad Hoc Development and Integration Tool for End Users (ADIEU), `http://www.alphaworks.ibm.com/tech/adieu`

[2] J. Zhang, C. K. Chang, and J. Y. Chang, "Mockup-driven Fast-Pprototyping Methodology for Web Requirements Engineering", *Proceeding of the 27th Annual International Computer Software and Applications Conference (COMPSAC'03)*, 2003.

[3] Y. Deshpande and S. Hansen, "Web Engineering: Creating a Discipline among Disciplines", *IEEE Multimedia*, Apr-Jun. 2001, pp. 82-87.

[4] B. Nardi, D. Schiano, M. Gumbrecht, and L. Swartz; "Why We Blog", *Commnunications of ACM*, December 2004, 47(12), pp.41-46.

[5] Wiki: "What Is Wiki", `http://wiki.org/wiki.cgi?WhatIsWiki`

[6] Wikipedia, `http://www.wikipedia.org`

[7] Endrei, M. et al, Patterns: Service-Oriented Architecture and Web Services, http://www.redbooks.ibm.com/redbooks/pdfs/sg246303.pdf

[8] Service-Oriented Architecture, http://www.ibm.com/software/info/openenvironment/soa

[9] S. M. Kim and M. Rosu, "A Survey of Public Web Services", In *Proceedings of the 13th Internaional World Wide Web Conference 2004*, pp.312-313, Chiba, Japan, 2004.

[10] A. Sutcliffe and N. Mehandjiev, "END-USER DEVELOPMENT", *Commnunication of ACM* 47(9):31-32, September 2004.

[11] IBM Rational Applcation Developer for WebSphere Software, http://www-306.ibm.com/software/awdtools/developer/application/index.html

[12] StrikeIron, Your Trusted Web Services Marketplace, http://www.strikeiron.com/

[13] The Web Service Club, http://objectclub.esm.co.jp/webservice/home.html (contents are described in Japanese)

[14] W3C, XML Path Language (XPath) Version 1.0, http://www.w3.org/TR/xpath

[15] ECMA International, Standard ECMA-262, *ECMAScript Language Specification*, 3rd edition, http://www.ecma-international.org/publications/standards/Ecma-262.htm.

[16] Business Process Execution Language for Web Services version 1.1, ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf

[17] OASIS Web Services Business Process Execution Language (WSBPEL) TC, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel

[18] XMethods web site, http://www.xmethods.net/

[19] W3C, Web Services Description Language (WSDL) 1.1, W3C Note, 2001.

[20] W3C, SOAP - Simple Object Access Protocol, http://www.w3.org/TR/SOAP

[21] W3C, XML Schema, http://www.w3.org/XML/Schema

[22] Mindreef: Comprehensive Web services diagnostics and testing, http://www.mindreef.com/

[23] Federal Reserve Bank of New York, Pilot Noon Foreign Exchange Rates Web Service, http://www.ny.frb.org/markets/pilotfx.html

[24] Web Services - Axis, http://ws.apache.org/axis/

[25] B. Myers, S. E. Hudson and B. Pausch, Past, Present, and Future of User Interface Software Tools, *ACM Transactions on Computer-Human Interfaces*, 7(1):3-28, March 2000.

[26] G. Fischer, E. Giaccardi, Y. Ye, A. G. Sutcliffe and N. Mehandiev, META-DESIGN: MANIFESTO FOR END-USER DEVELOPMENT, *Commnunication of ACM* 47(9):33-37, September 2004.

[27] R. Diestel, *Graph Theory*, Springer-Verlag New York, 1997.

[28] Amazon Web Services, http://www.amazon.com/gp/browse.html/002-0381178-1342459?%5Fencoding=UTF8&node=3435361

[29] Google Web APIs, http://www.google.com/apis/index.html

[30] eBay Developers Program, http://developer.ebay.com/soap/

[31] C. Kelleher and R. Pausch, Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers, *ACM Computing Surveys*, Vol. 37, No. 2, June 2005, pp. 83-137.

[32] YAML Ain't Markup Language (YAML) Version 1.1, http://yaml.org/spec/current.html, 2004.