E-service Based Information Fusion: A User-Level Information Integration Framework

Abdelsalam (Sumi) Helal and Jingting Lu

Computer and Information Science and Engineering Department,
University of Florida, Gainesville, FL32611, USA
helal@cise.ufl.edu
http://www.harris.cise.ufl.edu/projects/e-services.htm

Abstract. Managing fast-growing personal information is a time-consuming and laborious task that affects people's daily life. In addition, learning new skills is always an obstacle for end users, to fully take advantage of any new technology-based services. A framework is highly desired here to allow people to collect and fuse personal information without dealing with complex emerging technologies. In this paper, we introduce our e-service based Information Fusion framework for end-users. This framework enables end users to collect scattered information from diverse autonomous sources, and transparently create a repeatable process by which newer instances of the same information can be obtained in the future. By exploiting this framework, users won't need to repeat the manual information gathering task over and over again. We present our framework and provide some implementation details.

1 Introduction

With the fast pace by which we live today and with the information society we are becoming, people have more and more personal information concerning many aspects of their lives. This includes on-line brokering accounts, bank accounts, and credit cards, to mention just a few. People often spend substantial amount of time gathering and managing such information, every time this information or summary of it is needed. Usually, such information is scattered across different business sites. For example, many people have three or four bank accounts, more than one credit card, and several airline frequent flyer accounts. For regular online services, an end user needs to go to each individual web site to authenticate and manually fill in the necessary details to invoke a service and get the information (i.e. balance of a checking account). This is a time consuming process that will be repeated every time the end-user seeks a more up-to-date version of the information.

An alternative way to access information from different sources is using the emerging e-service technology. E-service is viewed as "any service or functionality that can be accessed by a business or a consumer programmatically on the Internet, using standard representation and protocols" [3]. It can greatly improve the efficiency of invoking and integrating services, including information. On the other hand, it involves fairly professional and complicated processes for end users, requiring them to have significant knowledge of e-service related specifications. Even for e-service specialists, it is their responsibility to modify the service requests correspondingly if a

particular service interface is changed later on, or to deal with status query directly for checking back the execution status of a long-running service. Regardless of its efficiency, the plain e-service framework is clearly a complex and inconvenient way of invoking services and gathering information.

The major goal of our E-service Information Fusion framework is to tame this complexity and simplify the use model of the emerging e-service standards so that services can be used transparently by non-expert-users. This can be done through a framework that involves a methodology at the e-service provider side, and a carefully designed, user-friendly interface that facilitates the access and integration of information on the internet, at the end-user side.

1.1 Motivating Scenario

Let's meet a software engineer here, Kin lee, who has three bank accounts in three different online banks: Bank One, Bank of America and Hong Kong Bank. Every time he checks his three accounts' balances, Kin spends a good deal of time on going into each individual bank's website and filling multiple forms to finally get the bank balances. Fig. 1.1 depicts this balance-querying scenario.

Before he can receive the balance at Hong Kong Bank, Kin needs to make n interactions with the web site to input his context such as account and password, etc. Similarly, m interactions are needed at American Bank, and k interactions at Bank One. Hence, Kin need to do a total of $(n + m + k)$ interactions in order to retrieve all three balances and finally calculating net worth, himself. Apparently, he has to repeat the $(n + m + k)$ interactions every time he wants to check the balances. And we haven't mentioned yet that he needs to memorize and put the accounts and passwords information somewhere safe. This repeated tedious process could turn into a nightmare and a waste of time for future web users. This can only exacerbate the more web users opt to the electronic statement option of the various businesses.

We know e-services can be invoked in a standard messaging through the Internet, facilitating different businesses to build their applications on different systems and technologies they prefer. Thus Hong Kong Bank, for instance, may implements the "balance" service as an e-service and publishes it in a public e-service repository to let more potential users (both businesses and individuals) use the service easily. The process that Kin will follow if he wants to exploit the e-service version of "balance" service is as follows. He first needs to refer to a public e-service repository to find the Hong Kong Bank's e-balance service description file, then generates the appropriate request message containing his context, and finally invokes the e-service to get the balance result. All exchanged messages should be based on some standard message format. And if the service interface is changed later on, Kin will get an error message when he tries to invoke the service again. Thus he must retrieve the service's new description file to change the service request accordingly. Obviously, this process is more suitable for businesses since they can invest much on implementing the corresponding software framework, whereas it is not necessary and practical for Kin to master the e-service related emerging technologies. A user-level e-service interface is clearly needed here.

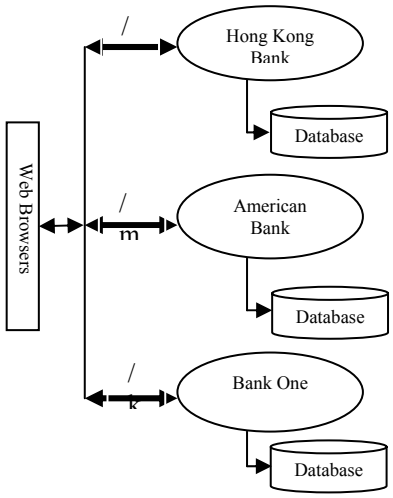

Fig.1.1. Kin needs to do ($n + m + k$) interactions to retrieve snapshots of balances from 3 different accounts every time. ("/n" represents n interactions,"/m" represents m interactions, "/k" represents k interactions*)*

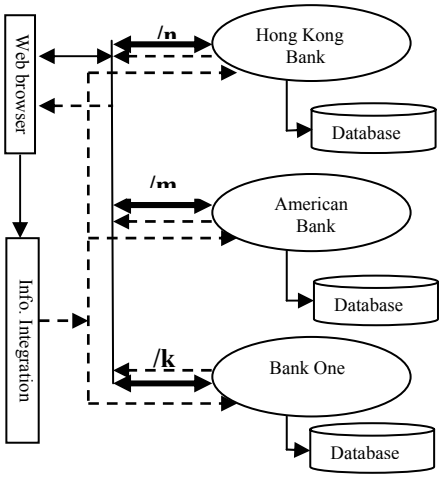

Fig.1.2. Kin made ($n + m + k$) interactions to retrieve snapshots of balances from 3 different bank accounts. Kin did this once, and in doing so, he extracted the process i.e. the e-service standard request message to an "Information Integration" module. (Dashed lines show the direct way for Kin to subsequently invoke the e-balance service).

Fig. 1.2 depicts our framework. It shows a more efficient process to query three different banks for three different balances from an "Information Integration" module without manually inputting any data by Kin. This process involves ($n + m + k$) interactions only the first time Kin queries the three banks. During these interactions, Kin transparently and reflexively extracts the user-service-based information gathering process such as the e-service standard request message. In the future, Kin will not visit each bank's web site and repeat the same interaction steps. Instead, Kin will use the information integration module to directly obtain the data he needs. In essence, the Information fusion framework creates a new meaning of e-service – "informational e-service", which encapsulates particular user-service-based information in an e-service invocation compared to regular e-services concept. As a result, Kin doesn't even need to search for the "balance" e-service description file in a public e-services repository to exploit the e-service. The information integration module handles all communication with the service provider and prompts Kin to input additional data only if necessary. This approach does not only save time, but also hide any changes made to the e-balance e-service interface, from Kin.

Furthermore, this framework provides a facility that can automatically evaluate all the embedded e-services binders at one time for users, reducing lots of user's intervention activities.

In effect, what Kin has accomplished is a user-level fusion of information from heterogeneous sources. Such problem has been a daunting task in the past. Today, with emerging e-service technologies such as XML, SOAP and WSDL, user-level integration is becoming possible.

Creating an information fusion engine that integrates users' scattered personal data, and transparently exploits e-services without requiring users to know or handle the technology and the implementation details is the motivation of our E-service Information Fusion Framework.

In the following subsections, we describe our e-service based framework that realizes the above pleasant vision. Section 2 introduces related works. In section 3, we discuss in details the design, architecture, and implementation of the e-service Information Fusion framework. Conclusion and future work are discussed in Section 4.

2 E-service Information Fusion

Our e-service Information Fusion framework aims at providing an Information Integration at the end-user level, hiding all data inputs, service implementations, and emerging standards from the users. The basic functionalities implemented by the E-service Information Fusion framework can be summarized as follows:

- Capture the context of a particular user-service-based interaction. Such context includes a service standard request message (SOAP message here) which contains a user's context and complies to the corresponding service interface definition, and a service URL from which the above request message can be captured and the e-service can be invoked later on.
- Provide an "e-service binder" which visually represents the new "informational e-services" and can be used by users through a tool to invoke a particular e-service directly.
- Provide a methodology for Information Fusion framework clients and service providers to follow. Such methodology will allow our framework to achieve information integration across heterogeneous and autonomous sources.
- Hide from users changes made to the e-service interface over time.
- Nicely convey long-running e-services to users
- Provides a facility to automatically evaluate all the embedded e-services binders at one time for users
- Provides simple manipulating functions on returned results, i.e. calculating summary of balance

2.1 E-service Information Fusion Architecture

Fig. 2.1 depicts the major components of the E-service Information Fusion architecture. We present the architecture through the steps that take place during three phases

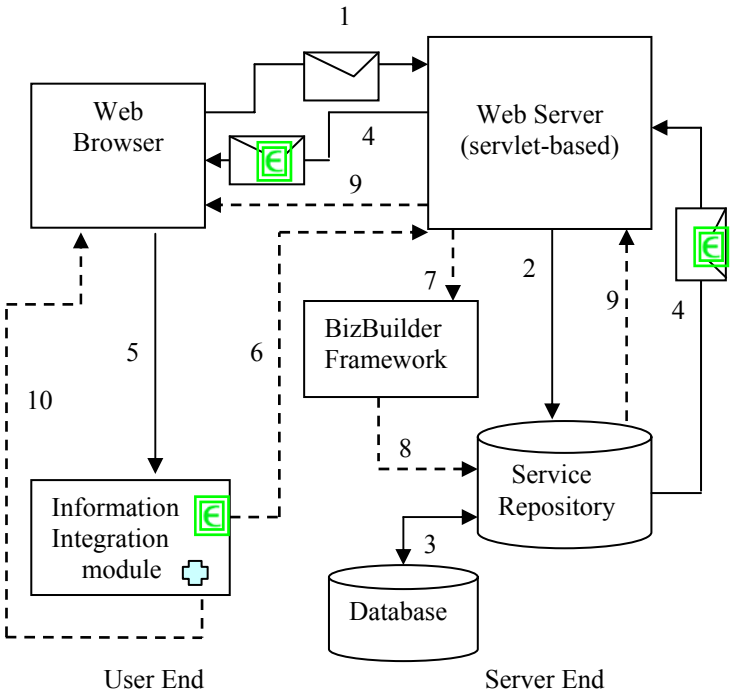

Fig.2.1. E-service Information Fusion Architecture (Solid-line represents the process for invoking legacy services; Dash-line represents the e-service invocation process through e-service Information Fusion framework; represents "e-service binder"; whereas represents simple manipulating functions).

Phase I. Using our methodology, the business provider creates e-services and publishes them in an *e-service repository* (Fig. 1.2 step 1). The business provider also links information that is dynamically generated (e.g. the service SOAP request including the particular user's context) to the e-service that computes this information. Our methodology is simple. It calls for structuring the applications behind certain information as e-services. It also calls for using a unique HTML representation to convey the fact that a piece of information in a dynamic web page has an underlying e-service implementation that is externalized to the end-user. Using our methodology, a business provider can decide on what information it wishes to externalize.

Phase II. A user accesses an online service for the first time, using a tool that follows our e-service framework. Such a tool is a standard browser connected to a "builder tool" that enables the user to "cut and paste" visual representation of the "informa-

tional e-services" that implement (compute) certain information. What the user sees during this browsing session is the data he needs, "shadowed" by visual indicators that information has externalized e-services available. A piece of information and the corresponding visual indicator will be unambiguously related be the way they are laid out next to each other.

The web server at a provider web site receives a user HTTP requests and extracts data from it (step 1). According to the invoked service name, a *service repository* will be searched to find the requested service (step 2). Since the service is implemented as e-service, at the point, the appropriate SOAP request is created for the use in a database, facilitating sending it back to the user when he cuts and pastes "e-service binder" to "builder tool" later on. The SOAP request is compatible with the corresponding e-service interface syntax, containing any authentication information and other parameters supplied by the user. The local service is finally invoked with the appropriate parameters, involving interactions with database (step 3). Along with the e-service results, part of the users' information (we name this critical *context*), the e-service URI, service name and provider name, all of which compose of a component named "e-service binder", will be presented back to the user's browser (step 4). The "e-service binder" is the visual cue that alerts the user that the "means" to obtaining this information again in the future can be cut and paste, and saved in the "builder tool" (we call this tool as Information Fusion Module) (step 5). The user can decide to cut and paste, and save, or may ignore the alert. By through the visual cut and paste operations, the corresponding e-service SOAP request message is being captured from the service side and saved along with e-service URL in context repository at the user end, accomplishing the "informational e-services" creation at users side. Getting complete SOAP request message from service implementers rather than generating the SOAP request at a user end is a good choice for making services implementation transparent from users.

Phase III. The Information Fusion module can be made to appear as web-presence (step 10). When users want to invoke the same service again, they only need to click the saved "e-service binder", and the binder will retrieve the corresponding e-service SOAP request message and service URL in context repository and communicate with the correct web server (step 6). The web server interacts with the BizBuilder framework (explained in section 3.3) to find the appropriate local service for the requested e-service (step 7). The corresponding service in the service repository is executed (step 8 and step 3), and the result is sent back to the user (step 9). Finally, the user can invoke multiple e-services. For instance, if the user "built" a global bank statement by copying and pasting multiple e-services, provided by three different banks, in phase II, the result would be a personal information integration record with multiple e-service references (and of course a set of hidden contexts) with some other kinds of personal data. The Information Fusion framework provides a facility to automatically evaluate all the embedded e-services binders at one time, reducing lots of users' intervention activities. Users can also define some predefined manipulating functions to apply them for same categorized results.

2.2 Information Integration Module

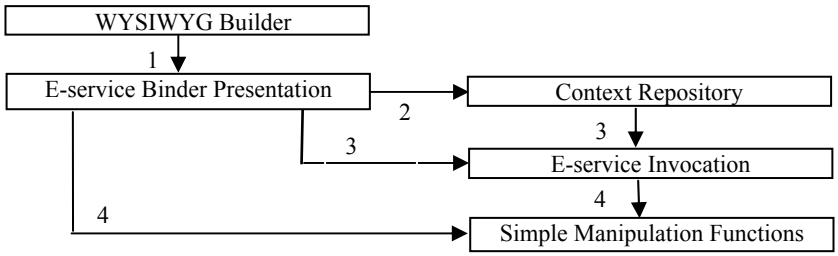

Fig.2.2. The components of Information Integration module (represents defined simple manipulating functions, represents "e- service binder")

The Information Integration module sitting at user's end is a core component in the information fusion architecture. Fig. 2.2 shows the sub-components of the Information Integration module.

In a neat WYSIWYG interface, users "cut and paste" the "e-service binder" to an Information Integration builder to create a record with reference to e-services using their "e-service binders" (step 1). The corresponding SOAP request and service URL are also "pasted" into a context Repository (step 2). By this mechanism, the "e-service binder" becomes a visual representation of so called "informational e-services".

By just clicking the "e-service binder", a standard SOAP request which contains the user's information is retrieved from the context repository and titled with the services URL, is sent to the appropriate web server to invoke the e-service (step 3). After users get service results, they can further perform the defined simple functions based on the results (step 4), such as calculating balance summary. This module provides simple steps for users to follow to create their e-services binders presentation and context repository, and invoke e-services directly and complete the defined functions.

We use the BizBuilder framework [3], which facilitates the invocation of e-services at the service provider side. This framework can convert legacy services (Java services particularly) to e-services and facilitate their invocation. Due to space limitation, we are unable to describe its details. However, the reader is referred to [3].

3. Design of the Information Integration Module

Our design extends a spreadsheet-like software named *Jeks*, which provides rich mathematical functions for users to manipulate resulting data. For space limitation, we present only the Jeks-based design.

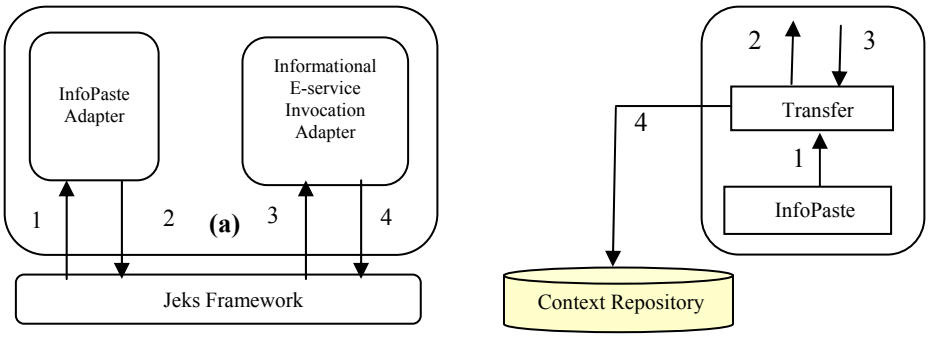

Fig.3.1. (a) The architecture of Jeks-based Information Integration module, (b) The InfoPaste Adaptor.

Jeks [6] is a GNU General Public License software created by Emmanuel Puybaret. It implements an extendable spreadsheet-like tool using Java swing Jtable, providing rich mathematical functions facilities. This spreadsheet-like interface is a nice workspace for our Information Fusion framework, since we can perform simple manipulating functions, especially aggregates. Furthermore, each individual cell is naturally a good placeholder for an independent "e-service binder". Fig. 3.1(a) illustrates the architecture of Jeks-based Information Integration.

There are two sub- adapters in the Information Fusion Adapter: InfoPaste and Informational E-service Invocation. They interact with Jeks framework, users' context repository and services server to accomplish their purposes.

3.1 InfoPaste Adapter

InfoPaste adapter is responsible for transferring a service context from the e-service server to the user's building tool. We borrow from the HTTP protocol and redefine a variant protocol format for the particular "e-service binder" component. This can be specified as follows:

http://serivceURL#providername#servicename#information

ServiceURL is the web address of a servlet-based e-service. "providername" is the name of a particular service provider like "Hong Kong Bank". "Information" contains minimum critical user information such as account, which can be used for the service server as index to search for the corresponding user-service SOAP request message in database.

Users can get the above important information by "copy and paste" the "e-service binder" component from web browser to the Jeks tool, utilizing the regular "copy shortcut" functions for the HTTP protocol and employing the special InfoPaste Adapter. The difference between special InfoPaste and regular paste function in the

Information Integration module lies in that InfoPaste can transfer the corresponding user-service-based context from the service's side to the user's side when it appears to do the same visual pasting as regular paste function.

Fig. 3.1(b) shows two sub-modules in the InfoPaste adapter architecture. When users "copy and paste" the "e-service binder" component from the presented web page to the Jeks tool, the InfoPaste sub-module is invoked (Fig.3.1(a) step 1); it parsers the "serviceURL", "servicename" and "information" data out and passes them on to the Transfer sub-module (Fig.3.1(b) step 1). The Transfer module communicates with services servers to request transferring the exact service SOAP request message (Fig.3.1 (b) step 2). After receiving the HTTP SOAP request, the servlet-based service server extracts critical information including the invoked service name like "balance", and user-unique context like account. Then it indexes them to its database to find the SOAP request message in per user-service base and send it back to the user (Fig. 3.1(b) step 3). Transfer module extracts the exact SOAP request message from the server response and save it along with the "serviceURL" to Context repository (Fig. 3.1(b) step 4). The row and column value of a cell will make the connection between an "e-service binder" in the cell and the corresponding user-service-based SOAP request in context repository. Finally, InfoPaste module presents the "e-service binder" with service provider's name into the cell that the user chooses to save the binder into (Fig. 3.1(a) step 2). The user can "copy" and "InfoPaste" multiple "service binder" from diverse services server, and save them to a file along with other categorized personal data and predefined manipulation functions, forming a personal information integration record (or information sheet).

3.2 E-service Invocation Adapter

E-service Invocation Adapter is the major component, which accomplishes the e-service invocation process. Fig. 3.2 shows four sub-modules comprising the E-service Invocation Adapter. These are *EvaluateE-serivces, CheckE-Serivce, updateRequest and ServiceStatus*. Due to space limitation, we only briefly describe these modules.

EvaluateE-Services module. Evaluates all embedded e-service binders in a given spreadsheet. The user opens his information integration file (sheet) and simply clicks the button for this function (Fig. 3.1(a) step 3). This module will scan each cell which contains an e-service binder one by one, invoking the *CheckE-Serivce* module for each e-service binder.

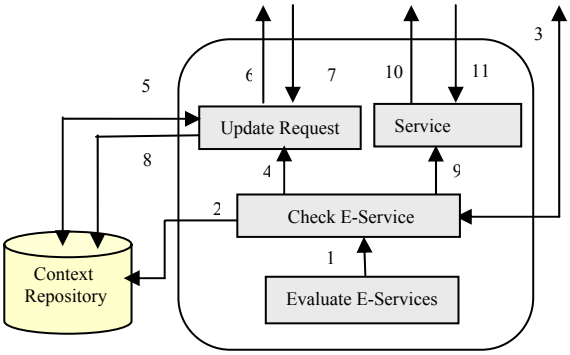

Fig. 3.2. The Architecture of E-service Invocation Adapter

CheckE-Service Module. When invoked by *EvaluateE-Serivces* module for evaluation of one e-service binder in a cell (Fig. 3.2 step 1), *CheckE-Serivce* module retrieves the corresponding user-service-based SOAP request message and serviceURL from a context repository indexed by the row and column values of the cell (Fig. 3.2 step 2). Then *CheckE-Service* sends the HTTP SOAP request to the service server to invoke the e-service and get the results back (Fig. 3.2 step 3). E-service Invocation Adapter can redirect the flow of action based on the returned results. If the e-service has been invoked without errors and the expected result is returned i.e. the balance, *CheckE-Service* can directly present the result back into the cell (Fig. 2.3(a) step 4).

One important achievement of the Information Fusion framework is graceful recover from errors due to change to the e-service definition. If this is to ever happen, the Information Fusion Adapter tries to reuse as much context as possible before it uses the new e-service definition to prompt the user for input (according to the new definition). In this case, the service will send back a SOAP request message containing a list for all missing or wrong-type parameters for the new e-service interface, with a particular method name such as *"mismatchedPara"*. After the new context is constructed and sent back to the service server, the Information Fusion recaptures the new SOAP request message generated by the service and updates the "e-service binder".

UpdateRequest module. Recaptures the new user-service-based SOAP request in case an e-service interface is changed over time. For example, HongKong Bank at first implemented the interface of "balance" e-service with two input parameters: account and password; now they want users to provide one more input: social security number to enhance security. The *UpdateRequest* module is invoked with all required additional data extracted from a service response by *CheckE-Service* module (Fig. 3.2 step 4), and the user is prompted to input additional data, achieving a change-tolerant end-user interface.

After *UpdateRequest* collects the desired new information, it merges the new data along with old but valid data from the context repository into a SOAP request and sends it over to the service server (Fig. 3.2 step 6). The e-service will regenerate the

appropriate SOAP request message according to the information sent by the user, and returns it back (Fig. 3.2 step 7). *UpdateRequest* stores the updated user-service SOAP request to the context repository (Fig. 3.2 step 8). Subsequent invocations of the e-service will retrieve the correct SOAP request message.

ServiceStatus module. ServiceStatus module is responsible for checking long-running service status for users.

4. Conclusion

We believe that achieving user-level information integration is becoming possible thanks to the emerging e-services technology. Our E-service Information Fusion Framework enables end users to construct their own information pages (or sheets) that integrate multiple autonomous information sources, all without having to learn or understand the complexities of the e-service technology themselves.

Early on during our research, we realized that the user interface design is very critical for the success of this framework. Specifically, our goal is to provide an easy "copy and paste "or "drag and drop" functions between a builder tool we are implementing and a general browser featuring contents from service providers that use our e-service based information fusion framework.

References

[1] http://java.sun.com/products/hotjava/3.0/, java.sun.com

[2] Kent Brown, "SOAP for Platform-Neutral Interoperability", http://xmlmag.com

[3] R. Krithivasan and A. Helal, "BizBuilder - An e-Services Framework Targeted for Internet Workflow," Proceedings of the third Workshop on Technologies for E-Services (TES'01), Springer Lecture Notes in Computer Science series, VOL. 2193. In conjunction with VLDB 2001, Sept 2001, Rome, Italy

[4] Richard Karpinski, "Inside UDDI", Transformation Today, June 7, 2001

[5] Yasser Shohoud, "Introduction to WSDL", http://www.devxpert.com

[6] Emmanuel PUYBARET, http://www.eteks.com/jeks/en/#WhatJeks