A Web Services Based Personal Information Integration Framework

Sumi Helal, Jingting Lu and Erwin Jansen
Computer and Information Science and Engineering Department,
University of Florida, Gainesville, FL32611, USA

ABSTRACT

Managing fast-growing personal web information is a time-consuming and laborious task that affects people's daily life. A framework is highly desired to allow people to collect and fuse personal information without dealing with complex emerging technologies. In this paper, we present 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 do not need to repeat the manual information-gathering task over and over again. We present our framework and provide some implementation details.

KEY WORDS
Web services, Information Fusion, Personal Information Systems.

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 on the web concerning many aspects of their lives. This includes on-line brokering accounts, bank accounts, and credit cards, frequent flyer programs, 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. An 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" [1]. It can greatly improve the efficiency of invoking and integrating services. 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 Based User-Level 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.

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 inter-actions 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 by him. Apparently, he has to repeat the $(n + m + k)$ interactions every time he wants to check the balances. Not to mention that he needs to memorize and put the accounts and passwords information somewhere safe. This 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 allowing more potential users (both businesses and individuals) to 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 he generates the appropriate request message containing his context, and finally he 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 would be very beneficial here.

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 $(n + m + k)$ inter-actions 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 hides 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 a lot of user intervention activities.

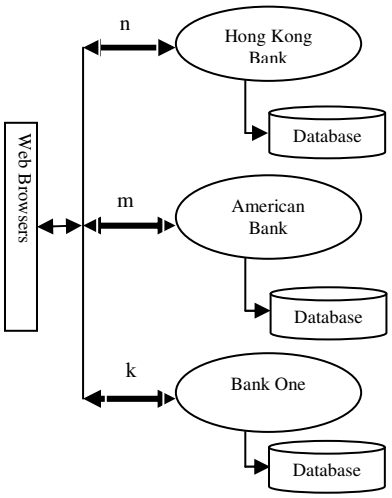


Fig.1.1. Kin needs to do ($n + m + k$) interactions to retrieve snapshots of balances from 3 different accounts every time.

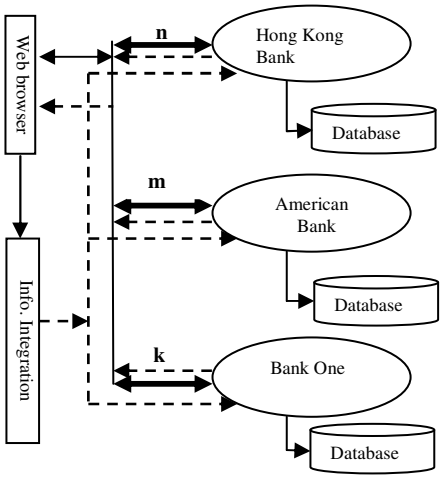


Fig.1.2. Kin made ($n + m + k$) interactions to retrieve snapshots of balances from 3 different bank accounts. Kin did this once, and in doing so, he extracted the process i.e. the e-service standard request message to an "Information Integration" module. (Dashed lines show the direct way for Kin to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

In effect, what Kin has accomplished is a user-level fusion of information from heterogeneous sources. Such a 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 the 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. But before we get into the details of our framework, we first introduce some related web service technologies we the describes the general idea of the framework. We proceed by discussing in details the design and implementation of the framework, respectively. Finally we show the use of the framework by a scenario analysis. The last section discusses conclusions and future work.

Related Web Service Technologies

The basic challenges are still focused on Web services modeling, interoperability, mathematical foundations for supporting federated Web services discovery, dynamic Web services composition, Web services monitoring, management, Web services security, privacy and Web services semantic computing.

SOAP

Simple Object Access Protocol (SOAP) is a high-level, lightweight protocol specification. It defines consistent message exchanging for structured and typed information in a decentralized and distributed environment [2]. It receives important support from industry giants Microsoft, IBM and SUN to small companies, and has become a widely adopted messaging technology for web services.

Similar to the way a web browser could open any web page as long as the environment can parse html, SOAP can invoke remote services regardless of the server's particular platform if an XML parser is provided at both servers and users ends. This desirable platform-neutrality feature is achieved by two facts: SOAP is an XML-based protocol, and it can bind with multiple Internet protocols. This appealing feature has been the driving force behind the fast development of SOAP as an alternative way to those existing RPC-based techniques.

WSDL

Web Services Description Language (WSDL)[3], an XML based language, specifies a mechanism to describe web services in consistent ways. IBM, Microsoft and Ariba deliver important support for WSDL development, and W3C is responsible for the WSDL standardization. Before businesses and programmers want to exploit third parties' applications, they need to know some essential information such as the location of third parties' services, what information needed to invoke the service and the underlying transport and message protocols etc. WSDL was introduced to meet this expectation by describing necessary technical information for invoking a remote service and can greatly reduce the interaction overhead between systems. WSDL achieves reusable message and protocol definitions by separating abstract definition of endpoints and message with concrete protocol and data format specification.

UDDI

The Universal Description, Discovery and Integration (UDDI) specification provides logically central but physically distributed databases, enabling any business at any-where to quickly, easily and dynamically find and integrate partners' services [7]. It is a cross-industry effort driven by platform developers, software developers and businesses [4]. More than 280 companies have become strong backers for UDDI specification.

It's a time-consuming and non-trivial process for businesses to find potential partners before the existence of UDDI. Different companies implemented their applications using diverse programming languages on various platforms, leading to an incompatible services world. All kinds of marketplace, directory providers and business take divergent ways to establish connectivity with partners. Invoking remote services is required to agree to protocols offline and depend on manual documentations [5]. As a result, businesses are limited to exploit those compatible web services of the known partners. UDDI was introduced to provide a next-layer-up to XML and SOAP, which are not enough to form a full end-to-end solution for e-business [6].

EBXML

EbXML stands for electronic business Extensible Markup Language. It was initiated by OASIS (Organization for the Advancement of Structured Information Standards) and UN/CEFACT (United Nations Center for Trade Facilitation and Electronic Business).

By defining standard business process and messaging, ebXML achieves great inter-operability in heterogonous systems. "The vision of ebXML is to enable a global electronic marketplace where enterprises of any size and in any geographical location can

meet and conduct business with each other through the exchange of XML-based messages."([8]) The easy-installed nature of ebXML makes it possible for small and medium companies to join in the e-business world. EbXML also meets the compatible requirement by keeping XML good features of interoperability and at the same time providing a framework in which extensively existing EDI (Electronic Data Inter-change)-based applications can also fit.

E-Service Based User-Level Information Fusion Framework

Our E-Service Based User-Level Information Fusion Framework aims at providing Information Integration at the end-user level, hiding all data inputs, service implementations, and emerging standards from the users. A preliminary version of our frame-work was introduced in [9]. The basic functionalities implemented by the 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 with 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 in-formation integration across heterogeneous and autonomous sources.
- Hide from users changes made to the e-service interface over time.
- 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

E-Service Based User-Level Information Fusion Design

Fig. 3.1 depicts a process of how a user will exploit the E-Services Based User-Level Information Fusion framework. We present the process through the steps that take place during three phases.

Phase I: Using our methodology, the business provider creates e-services and publishes them in an e-service repository. 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 "informational e-services" that implement 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.

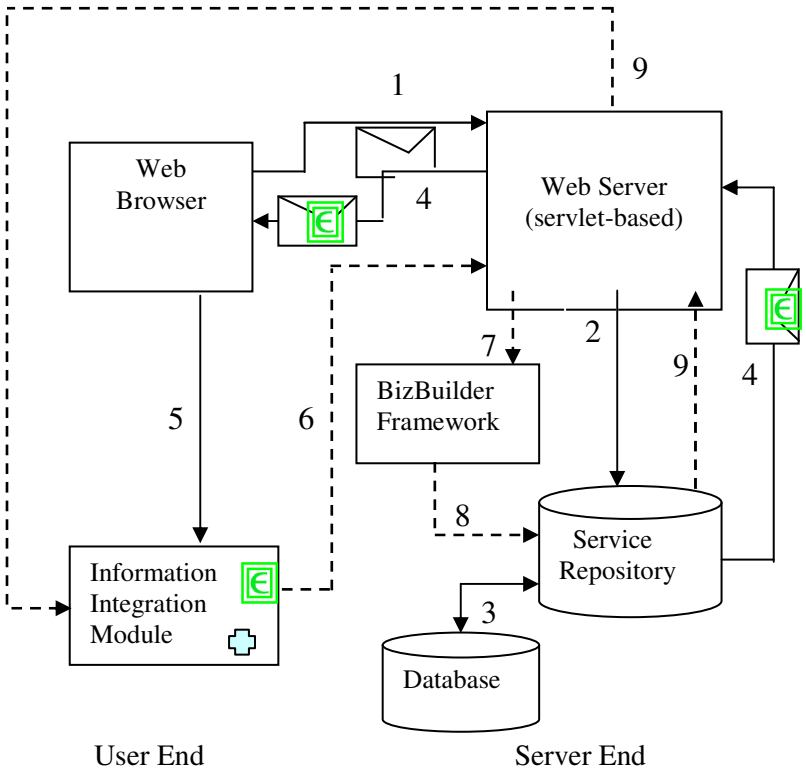


Fig.3.1. The use of E-service Based User-Level Information Fusion Framework (Solid-line represents the process for invoking legacy services; Dash-line represents the e-service invocation process through e-service Information Fusion framework; the green box represents "e-service binder"; whereas the plus represents simple manipulating functions).

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 data-base, 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 Integration Module) (step 5). The user can decide to cut and paste, and save, or may ignore the alert. Through the use of 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 the users side. Getting a complete SOAP request message from service implementers rather than generating the SOAP request at the user end is a good choice for making service implementations transparent from users.

Phase III. When a user wants to invoke the same service again, the user only needd 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 frame-work (explained later on), 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 "builds" 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.

Information Integration Module

The Information Integration module sitting at the users end is a core component in the information fusion architecture. Fig. 3.2 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 the "e-service binders" (step 1). The corresponding SOAP request and service URL is 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".

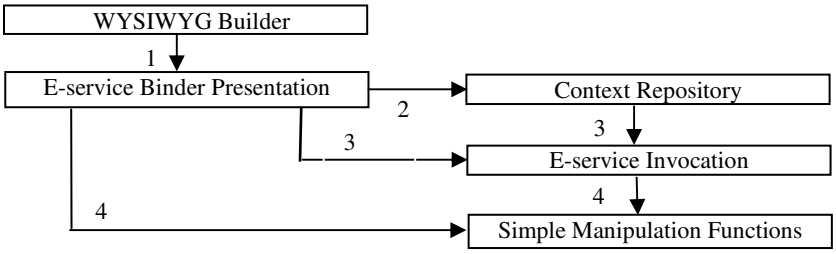


Fig.3.2. the components of Information Integration module

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. This is sent to the appropriate web server to invoke the e-service (step 3). After the user gets the service results, the user can further perform the defined simple functions based on the results (step 4), such as calculating balance summary. This module provides

simple steps to create their e-services binders' presentation, context repository, and invoke e-services directly to complete the defined functions.

BizBuilder Framework

The BizBuilder framework [1], currently implemented in Java, facilitates the invocation of e-services at a service provider side. This framework can convert legacy services (Java services particularly) to e-services and facilitate their invocation. Figure 3.3 briefly describes the partial components of the BizBuilder Framework. We extended the original BizBuilder with the new achievements of our Information Fusion framework.

We quickly describe how BizBuilder works here. For more details, the reader may consult[1]. Refer to Figure 3.3, after the servlet-based web server receives a SOAP request (step 0), ServiceInvocation refers to an "XML-SOAP parser" to parse the SOAP request (step 1), and performs "e-service mapping" (step 2) to find the corresponding service for the requested e-service (step 3). Then the service object instance is created and executed (step 4). The results along with an "e-service binder" are dynamically and unambiguously presented to the requesting web browser (step 5).

Since we take into consideration three situations of "informational e-services" invocation in Information Fusion framework, including synchronized services invocations, services interfaces changes and long running services invocations, we need to extend the ServiceInvocation Module to adapt these situations. If the e-service finishes immediately, the ServiceInvocation sends back the final results directly. Otherwise, ServiceInvocation returns a whole list of missing or wrong-type parameters for a particular service, indicating this is a special response expecting users to input more personal information. For long-running e-service invocation, ServiceInvocation returns a workflow SOAP query message to users, facilitating users to check the service status later on without knowing the interface of the extra workflow query e-service to users.

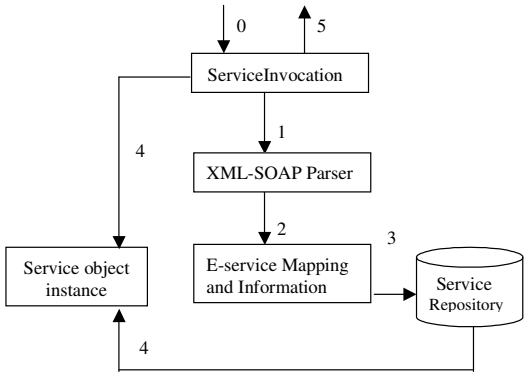
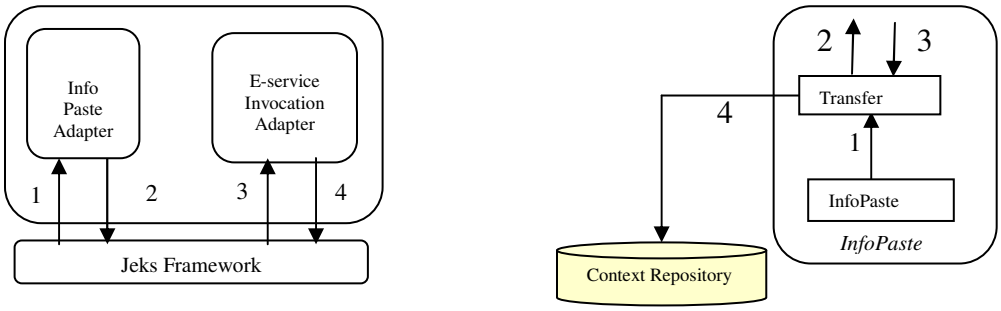


Fig.3.3 Partial components in the BizBuilder framework

Architecture of Jeks-Based Information Integration

Jeks [10] is 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." Figure 4.1(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 servers to accomplish their purposes.



(a) **(b)**

Fig.4.1. (a) The architecture of Jeks-based Information Integration module, (b) The InfoPaste Adaptor.

InfoPaste Adapter

The 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

A ServiceURL is the web address of a servlet-based e-service. A "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 side to the user side when it appears to do the same visual pasting as regular paste function.

Fig. 4.1(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.4.1 (a) step 1); it parsers the "serviceURL", "servicename" and "information" data out and passes them on to the Transfer sub-module (Fig.4.1 (b) step 1).

The Transfer module communicates with services servers to request transferring the exact service SOAP request message (Fig.4.1 (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 ac-count. 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. 4.1(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. 4.1(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. 4.1(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).

E-Service Invocation Adapter

The E-Service Invocation Adapter is the major component that accomplishes the e-service invocation process. Fig. 4.2 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.

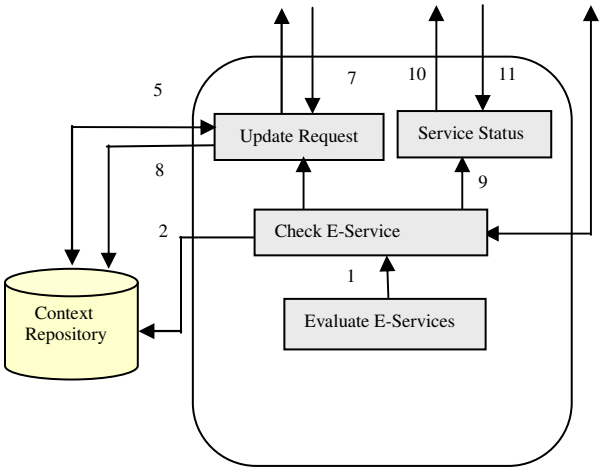


Fig. 4.2. The Architecture of E-Service Invocation Adapter

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. 4.1(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.

CheckE-Service Module: When invoked by EvaluateE-Serivces module for evaluation of one e-service binder in a cell (Fig. 4.2 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. 4.2 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. 4.2 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. 4.1(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. 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 up-dates 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. 4.2 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. 4.2 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. 4.2 step 7). UpdateRequest stores the updated user-service SOAP request to the context repository (Fig. 4.2 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. Based on BizBuilder framework, the status checking for long-running service is going through the workflow adapter. In other words, the status-query of synchronized service is viewed as workflow query in BizBuilder framework.

If the invoked e-service is long-time involving, CheckE-Balance will explicitly tell users it's a long-running one and send back a workflow SOAP query message at same time. ServiceStatus Module will temporally remember the workflow SOAP query message for the e-service (Figure 4.2 step 9). When the user wants to check whether the e-service has finished or not, ServiceStatus retrieved the workflow SOAP query message and send it to the e-service server (Figure 4.2 step 10). The e-service will either return final results or the same workflow SOAP query message again back to the user according to whether the service is completed or still in process (Figure 4.2 step 11). If the service is still in process, the user may repeat the same status-checking process later on until he finally gets the results. In a similar design logic with service SOAP request message, capturing the complete workflow SOAP query message from service implementers rather than generating the message at users ends is to make as much as possible services implementations transparent from users.

Implementation

In this section, we present details of the implementation of our Information Fusion framework.

SOAP Messaging, XML Parser and Servlet-based Server

Interoperability is a major contribution of e-services, and this desirable feature is achieved by using standard messaging. By non-property messaging, e-services can be flexibly invoked regardless of servers' particular platform. Both users and services can be guaranteed freedom to choose the technology they prefer, achieving low-cost services implementations and services invocation, and expanding potential markets in the mean time.

Simple Object Access Protocol (SOAP) is an XML-based, transport-independent and platform-neutral protocol specification. Hence, both services and users need to extensively use XML parser to parse the SOAP message. We use the Oracle XML parser in the Information Fusion framework.

There are two servlets installed at services sites. One is responsible for receiving requests from and returning results to the services' web interfaces. The other is listening to HTTP remote requests that come from end users.

Two special E-services

For e-services providers who want to comply with the Information Fusion frame-work, they need to provide two mandatory e-services with consistent services interface definitions: a "transfer" e-service and a "updateRequest" e-service. These can be viewed as special e-services for our Information Fusion framework. They have to be registered in a public e-services repository i.e. UDDI repository. Services providers can easily search for the e-services definitions in the public e-services repository. Implementing and exploiting these two mandatory e-services is one assumption of our framework. For simplicity, we define interfaces as simple as possible for these two special e-services

Transfer E-Service

The transfer e-service is the service requesting to transfer a particular user-service-based SOAP request message from services providers. The e-service name is defined as "transfer", and the interface has two parameters: the first is the actual e-service name with which users want to transfer the SOAP request, and the second is users' critical information. The combination of these two parameters is used as an index for a service server to search for the corresponding SOAP request in database (See following "BizBuilder Framework" section). Figure 5.1 shows an example of "transfer" SOAP request message.

The above request message indicates that the Information Fusion framework in-tends to invoke an e-service - "transfer" for a particular user, who was assigned account "12345" and wants to capture the SOAP request for the e-service "balance".

```
<?xml version = '1.0'?>
<SOAP-ENV:Envelope xmlns:SOAP-   ENV=
                "http://schemas.xmlsoap.org/soap/envelope/"
                SOAP-ENV:encodingStyle
                    ="http://schemas.xmlsoap.org/soap/encoding/">
          <SOAP-ENV:Body>
              <transfer>
                    <service>balance</service>
                    <index>12345</index>
              </transfer>
         </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 5.1. SOAP request message for the "transfer" e-service

UpdateRequest E-Service

The UpdateRequest e-service is for recapturing the updated SOAP request in the situation of a change of an e-service interface. The service name is defined as "updateRequest", and the service interface has two parameters: one is the actual e-service name that users want to update the SOAP request, and the second is a hash table merging new users' data with old but valid ones. Figure 5.2 shows an example of "updateRequest" SOAP request message.

```
<?xml version = '1.0'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=
                "http://schemas.xmlsoap.org/soap/envelope/"
                SOAP-
          ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <SOAP-ENV:Body>
            <updateRequest>
                <service>balance</service>
                <paraHashtable>
                        <account>12345</account>
                        <password>123</password>
              </paraHashtable>
          </updateRequest>
      </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 5.2. SOAP request message for the "updateRequest" e-service

This SOAP message requests to update SOAP request message for e-service "balance", by using a hash table that contains user's context: account "12345", password "123". The service server will regenerate the correct SOAP request message according to the updated users' context.

BizBuilder Framework Implementation

We extended the original BizBuilder to make it workable with the new achievements of Information Fusion framework. Here we will focus on describing the ex-tended implementation part, and the details of the BizBuilder framework implementation can be consulted with [1]. Generating SOAP request and presenting "e-service binder". When users first time invoking online services, for example the "balance" e-service, the service generates the corresponding user-service-based SOAP request (Figure 5.3) and saves it into a file named "balance_12345" if the user has an account "12345". Finally, the service presents the result to a web browser along with the visual "e-service binder".

Invoking an e-service: When the user invokes an e-service by clicking the "e-service binder", the sevlet-based server receives a HTTP request holding the complete user-service-based SOAP request. The XML parser will extract the complete SOAP request out and parse it to get the invoked e-service name and all the parameters information and invoke the appropriate java objects.

```
<?xml version = '1.0'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=
            "http://schemas.xmlsoap.org/soap/envelope/"
            SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
          <SOAP-ENV:Body>
            <balance>
                  <account>12345</account>
                  <password>123</password>
            </balance>
          </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 5.3. SOAP request message for a "balance" e-service

Before invoking the local service, ServiceInvocation will perform a match between the defined parameters in the service interface and input parameter from a user to make sure the user provides a correct version of a SOAP request in the case the service interface has changed. In addition, the match can reorder the input parameters values to let them in a consistent order with the one defined in the e-service interface.

If there is a parameter name defined in the e-service interface but not in input parameters list, it will be considered as a mismatch. The InvocateObject Module stores all those missing parameters in a string. In this step, we do not consider those extra input parameters not defined in the e-service interface as mismatched parameters any more. In other word, we consider the old SOAP request still a valid one in this situation. Only when the old parameters are not sufficient (including the situation of changes of parameters type), a new SOAP request is generated by the service for the user.

If there is any mismatch between two lists, the InvocateObject constructs a SOAP message with a node name "mismatchedPara" and a list of all missing parameters, and sends the complete SOAP message back. The service returns those only missing

parameters, trying to use existing but valid parameters as much as possible. Figure 5.4 shows an example SOAP response indicating one mismatched password parameter.

```
<?xml version = '1.0'?>
    <SOAP-ENV:Envelope xmlns:
          SOAP-ENV=http://schemas.xmlsoap.org/soap/envelope/
          SOAP-ENV:encodingStyle=
                "http://schemas.xmlsoap.org/soap/encoding/">
       <SOAP-ENV:Body>
            <mismatched1>password</mismatched1>
       </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
```

Figure 5.4. Example SOAP message indicating mismatched situation

If these two parameter lists match exactly, it means the e-service interface remains the same as before. The service object instance is created and executed. If the service is a synchronized one, results are returned back to the user immediately.

If the invoked e-service is a longterm service, the e-service generates a unique transaction ID and inserts it as a parameter in a workflow SOAP query message. The whole SOAP query message with a node name "WFGetStatus" is sent back to the user in the same mechanism as sending back a regular SOAP service request message. Figure 5.5 shows an example SOAP workflow query message with a unique id "T-1292318963".

```
<?xml version = '1.0'?>
     <SOAP-ENV:Envelope xmlns:SOAP-ENV=
               "http://schemas.xmlsoap.org/soap/envelope/"
               SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
          <SOAP-ENV:Body>
               <WFGetStatus querytype="workflow">
                    <txnid>T-1292318963</txnid>
               </WFGetStatus>
          </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 5.5. Example of a SOAP workflow query message

Transferring a user-service-based SOAP request. When the user infopaste the "e-service binder" into a cell in the Jesk, the servlet-based server will receive a HTTP request intended to invoke the special "transfer" e-service. The XML parser parses the request to get the actual e-service name with which the user wants to transfer a SOAP request, and the user's critical information. These two parameters are used to locate the corresponding files such as "blance_12345" in server's database. The "transfer" e-service retrieves the user-service-based SOAP request message from the appropriate file and returns it to the user.

Updating a user-service-based service request: After a service server receives an update service request, the XML parser parses the message to get the actual e-service

name with which the user wants to update the SOAP request, and a hash table that contains the user's updated context. The service will regenerate the user-service-based SOAP request and send it back to the user.

Handling with workflow query message: The status query of long-running service is viewed as workflow query in BizBuilder framework. If it detects a SOAP request is a workflow query, thr ServiceInvocation Module extracts the assigned transaction id to query the status of the object execution. If the service invocation has finished, the server sends back the final results directly. Otherwise, the server sends back the same SOAP workflow query message to the user.

InfoPaste Adapter

The InfoPaste Adapter is the core component to capture a user-service-based SOAP request message from an e-service server to the Information Integration builder tool for the first time.

Context Repository

File Name	File Contents Description
filename_eservices.txt	lists all the cells presenting e-services binders
filename_cell_providerName_serviceName.txt	list services providers names and services names for cells presenting e-services binders
filename_cell_servletAddress.txt	list servlet Address for cells presenting e-services binders
filename_row_column_request.txt	contains the complete SOAP request message and the service URL for a particular e-service presented in a cell
filename_row_column_query.txt	contains the complete workflow SOAP status request message for a particular long-running e-service presented in a cell

Table 5.1: Files in context recovery

Table 5.1 lists all the files in Context Repository, which are used by InfoPaste Adapter. "Filename" represents any name a user chooses to save his individual information file. E-services.txt is a log file which records all the cells presenting e-services binders, facilitating EvaluateE-Services Module to evaluate all the informational e-services at one time for users.

Row_column_rueqest.txt contains critical information in context repository. It holds a complete SOAP request message for a particular "informational e-service" presented in a cell with the corresponding row and column value. By transferring and saving a whole SOAP request message, users don't need to have any knowledge about an e-service interface definition. Instead, they just "blindly" retrieve the whole SOAP request message as a regular string from the appropriate row_column_request.txt file and wrap it inside SOAP body in a SOAP message.

Row_column_query.txt holds a complete workflow SOAP status request message for a particular long-running e-service presented in a cell with the corresponding row and

column value. Similar with regular SOAP request service message, by transferring and saving a complete workflow SOAP status request message, users' ends don't need to have any knowledge about the interface definition of this extra e-service. When a user uses the framework for the first time, the adapter will create a root directory "fusion" under the current executing directory, and a sub-directory "info" under the root directory.

The sub-directory "info" holds e-services.txt, cell_providerName_serviceName.txt and cell_servletAddress.txt files. In addition, the adapter will generate sub-directories for each categorized e-service and group all the row_column_request.txt and row_column_query.txt files of that particular categorized e-service under the corresponding sub-directory. For example, if there are two categorized e-services "balance" and "loan" contained in Jeks information sheet, two subdirectories named "balance" and "loan" will be dynamically created under the "Temporary" directory respectively. The "balance" directory groups all the row_column_request.txt and row_column_query.txt files of those cells that contain "balance" e-services binders. The "loan" directory is for "loan" e-services binders.

Transfer Module

This module exploits one of the special e-services for the Information Fusion framework: "transfer" e-services. It constructs a SOAP request as shown in Figure 5.1, and sends it as a HTTP request to the servlet-based service server. After receiving the service HTTP response, the Transfer Module extracts the complete user-service-based SOAP request and save it into the corresponding row_column_request.txt file. This module also updates cell_providerName_serviceName.txt and cell_servletAddress.txt files accordingly.

InfoPaste Module

When users "copy and paste" an "e-service binder" component from a presented web page to a cell in Jeks tool, the InfoPaste sub-module will parse the "serviceURL", "servicename" and "information" data out and pass them to transfer sub-module. After the Transfer Module finishes its functions, InfoPaste Module will update the eser-vice.txt files accordingly, and present the visual e-service binder to the cell showing the service provider name and the service name at the same time.

E-Services Invocation Adapter

There are four modules contained in this adapter: EvaluateE-Services Module, CheckE-Service Module, UpdateRequest Module and ServiceStatus Module.

EvaluateE-Services Module

This module provides a scanning facility that can evaluate all the e-services at one time, reducing lots of users' intervention activities. The eserivce.txt file is the central information, which logs all the cells containing "e-services binders". Going through this

information record, the EvaluateE-Services Module will pass the row and column value of each listed cell to CheckE-Service Module to evaluate each e-service binder, providing a visual one-time evaluation effect for users.

CheckE-Service Module

Based on the row and column value passed in, the CheckE-Service Module locates the corresponding row_column_request.txt file and retrieves the user-service-based SOAP request message and find out the serviceURL in cell_servletAddress.txt file in the mean time. Then the CheckE-Service Module sends an HTTP request containing the SOAP request message to the service server to invoke the e-service.

After receiving the service HHTP response, CheckE-Service can differentiate three invocation situations and consequently redirect the flow of action, based on the node name specified in the SOAP response.

If the node name in response is "mismatchedPara", that indicates the e-service interface has been changed and users are expected to input more personal information. CheckE-Service Module will remember the list of missing parameters extracted from the SOAP message and send it to UpdateRequest Module.

If the specified node name in response is "WFQuery", it means the user is invoking a long-running e-service and the service server has sent back a complete workflow SOAP status query message to facilitate the user to check the service status later on. The workflow SOAP status query message contains a unique transaction ID assigned by the service. CheckE-Service Module will temporally remember the workflow SOAP status query message to the corresponding row_column_query.txt file and explicitly tell users it's a long-running e-service and they can check the service status back after a while. If the e-service has completed immediately, the expected results will be directly presented back into the cell.

UpdateRequest Module

After it gets the information from the CheckE-Service Module, the UpdateRequest Module tokenizes the information to get the list of missing parameters' names. It then prompts the user to input his information one by one and remembers the pair of parameter names and values into a hash table. After collecting all the new information, The UpdateRequest Module will retrieve and parse the old SOAP request message to get old parameters information and appends them to the same hash table. The UpdateRequest Module will create a SOAP request based on the merging parameters information to invoke the special "updateRequest" e-service. In the situation that some old parameters are no longer used in the new service interface, it will be the service responsibility to filter out the extra parameters and regenerates the user-service-based SOAP request

message for the user. After UpdateRequest gets the updated SOAP request from service response, it uses this new SOAP request to update the row_column_request.txt file.

ServiceStatus Module

The ServiceStatus Module is responsible for checking a long-running service status for users. When the user wants to check whether an already-invoked e-service has finished or not, ServiceStatus retrieves the workflow SOAP query message from the row_column_query.txt file and send a HTTP request to the e-service server. If the e-service is completed, the e-service returns final results to the user. If the e-service is still in process, the same workflow SOAP query message is returned back and saved by the framework to the row_column_query.txt file again.

SCENARIO ANALYSES

In this section, we will take a look at how the E-Service Based User-Level Information Fusion framework can be used by Kin Lee to construct his personal information sheet. In the motivation scenario section, we understand either the time-consuming or new technology involving process Kin Lee has to face when he wants to retrieve his balance information every time. Based on the same scenario, we will see that Kin needs only to perform the trivial manual procedure one time and gets his information sheet by using the Information Fusion framework.

Creating an e-service binder. Kin first goes to the Hong Kong Bank web site and fills in all the required information to get his balance just like he always does. Hong Kong Bank will return the balance result as well as an "e-service binder" component. Kin then uses the "copy shortcut" function of the web browser to copy down the "e-service binder" component virtually.

Pasting an e-service binder. In the Jeks Excel-like sheet, Kin chooses a cell and then clicks the "InfoPast" item from menu to paste the "e-service binder" virtually to the cell (Figure 6.1 and Figure 6.2).

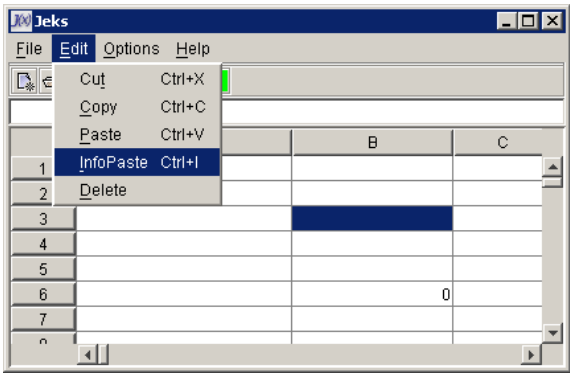


Figure 6.1. Snapshot of the "InfoPaste" menu item

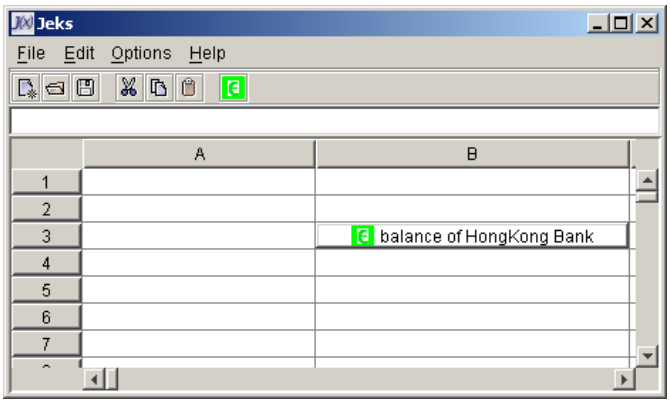


Figure 6.2 Snapshot of the e-service binder for the Hong Kong Bank's balance ser-vice.

Kin goes to Bank One to go through the similar procedure to generate the corresponding "e-service binder" for the balance e-service of Bank One in another cell in Jeks sheet (Figure 6.3).

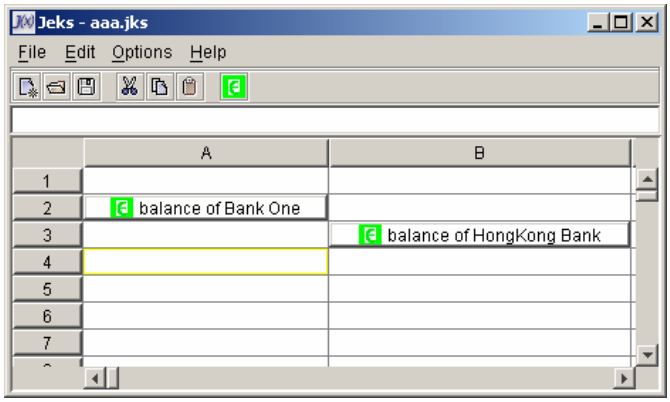


Figure 6.3. Snapshot with two "e-services binders"

Defining simple manipulated functions: balance summary. In the Jeks sheet, Kin can define a simple sum mathematic function in a cell to indicate that the sum will calculate results from certain cells. Figure 6.4 shows a sum function in cell B6 will calculate sum of balance in cell A2 and B3.

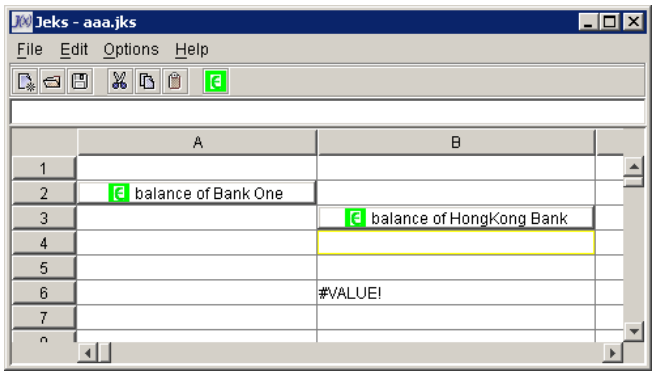


Figure 6.4. Snapshot with a predefined sum function

Evaluate all e-services binders. Kin clicks the button, and Jeks will evaluate all the e-services binders for him. After the evaluation process finishes, Kin can re-evaluate the cell defining the sum function, and the cell will show the balance summary result (Figure 6.5)

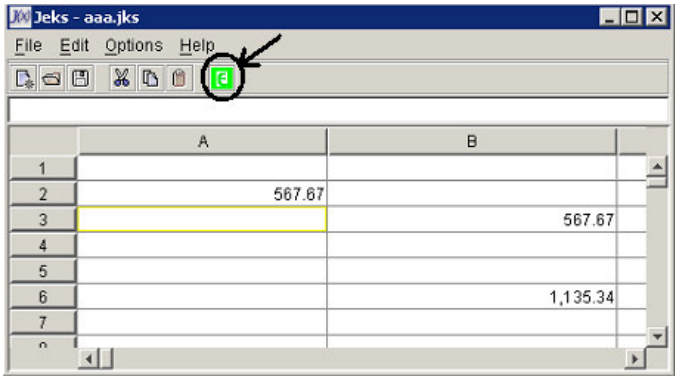


Figure 6.5. After Kin reevaluate the cell defining the sum function.

There are three services invocations situations:

- If e-services complete and return immediately, Figure 6.5 shows the resulting snapshot.
- If the e-services interface has been changed since Kin "copied and pasted" "e-services binders" last time, Jeks will pop up textboxes to prompt Kin to input more personal information (Figure 6.6). After collecting all the additional information and communicating with the services providers, Jeks will update the corresponding "e-services binders". Kin can reevaluate the cell to get the proper results.

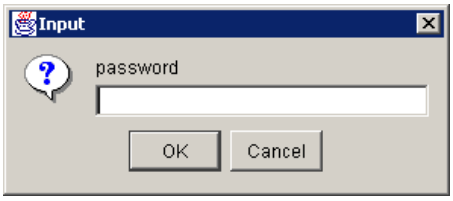


Figure 6.6. Snapshot of a popped up textbox for a mismatched situation

- If the invoked e-services are long running ones, a pop up message will inform Kin about the nature of the services (Figure 6.7, we use an async_balance to simply represent a long-running one). Kin can click the cell to get the execution status of the e-services anytime later on.

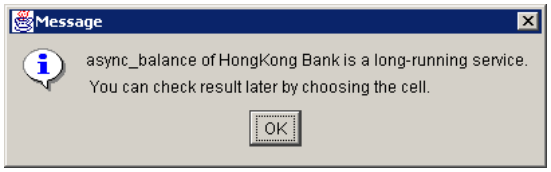


Figure 6.7. Snapshot of a pop up message for a long-running e-service

Conclusion and Future Works

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

Early on during our research, we realized that the user interface design is very critical for the success of this framework. The framework provides an easy "copy and paste" function between the builder tools we have implemented and a general browser featuring contents from services providers who use our E-Services Based User-Level Information Fusion Framework.

In the future, there are three considerations for the framework development. The first consideration is to achieve intelligent information sheets. We can apply events or rules to an e-service binder, and the events will be triggered automatically before a user invokes an e-service. The second consideration is to achieve a more appealing "drag and drop" effect between the builder tool and a general browser. The "drag and drop" will deliver neat and complete visual transfer effects for "e-services binders" for end users. The third consideration is to achieve online accessibility, which is an appropriate direction to make the framework more popular and transparent for end users. At the same time, users will be guaranteed more freedom to integrate the kinds of personal information they are interested on the web sheet. Wrapping the whole framework, as a java bean component and embedding it in web pages could be a suit-able way to make our framework online.

References

[1] R. Krithivasan and S. Helal, *BizBuilder — An E-Service Framework Targeted for Internet Workflow*, Proceedings of the third Workshop on Technologies for E-Services (TES'01), vol. 2193, 2001.

[2] "Simple Object Access Protocol (SOAP) 1.1", The World Wide Web Consortium, http://www.w3.org/TR/SOAP/ (current Aug. 2002).

[3] "Web Service Description Language (WSDL) 1.1", The World Wide Web Consortium, http://www.w3.org/TR/wsdl (current Aug. 2002).

[4] "UDDI Executive White Paper", www.uddi.org, http://www.uddi.org/pubs/UDDI_Executive_White_Paper.pdf (current Aug. 2002).

[5] B. Sleeper, "Why UDDI Will Succeed, Quietly: Two Factors Push Web Services Forward", http://www.stencilgroup.com/ideas_scope_200104uddi.html (current Aug. 2002).

[6] "UDDI Technical White Paper", www.uddi.org, http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf (current Aug. 2002).

[7] R. Karpinski, "Inside UDDI", http://www.internetweek.com/transtoday01/ttoday060701.htm

[8] "ebXML Technical Architecture Specification v1.04", www.ebxml.org, http://www.ebxml.org/specs/ebTA.doc (current Aug. 2002).

[9] S. Helal and J. Lu, *E-service Based Information Fusion: A User-Level Information Integration Framework*, Proceedings of the Fourth Workshop on Technologies for E-Services (TES'02), In conjunction with VLDB 2002, Hong Kong, August 2002.

[10] E. Puybaret, http://www.eteks.com/jeks/en/#Wh-atJeks (current Aug. 2002).

ABOUT THE AUTHORS

Sumi Helal is professor of computer science and engineering at the University of Florida. His research area span mobile an dpervasive computing and Internet computing. He is a senior member of the IEEE and a member of the ACM and USENIX Association. Contact him at helal@cise.ufl.edu.

Jingting Lu obtained her Masters of Computer Engineering from the Computer and Information Science and Engineering at the University of Florida. She can be reached at lujingting2001@yahoo.com.