Web Personalizer as User Consultant

Kamran Sartipi[‡] and Mehran Najafi[♣]
DeGroote School of Business[‡], Department of Computing and Software[♣]
McMaster University
Hamilton, ON, L8S 4M4, Canada
{sartipi, najafm}@mcmaster.ca

Introduction

An increase in the level of abstraction and ease of use in web-based computing requires more complexity and sophistication at the lower levels of implementation. Service oriented architecture (SOA) has provided more independency for the web service users, where the user can focus on the business logic aspects of the service integration. While the SOA technology has advanced the development of complex and distributed systems, it is still applied at the development phase and requires certain knowledge of system design and messaging techniques. The emergence of social network environments, user centric concept in information systems, and web application integration technologies such as Mashups, are driving forces for computer scientists and IT professionals to provide a customizable and effective environment for utilizing the rich variety of Internet resources. In this short paper, we propose different research avenues towards the overall requirements of the smart Personal Web systems. We propose the concept "Web Personalizer" which is a collection of three generic agents that are deployed at different platforms. These agents are specialized using roles and training skills to represent delegates from service provider to the user, or from user to the service provider to offer high-level services and consultations to the user.

The current state of the web applications is represented by the services that require extra knowledge and expertise from a normal user to take advantage of the available features and operations of the services. Given the large variety of web applications and the users' tight time schedules, the users will have to limit themselves to a minimum set of available service features. This is also the case in using other types of computerized systems such as automobile gadgets, home appliances and entertainment centers. In other words, the proper and efficient use of computerized systems (embedded or software based) requires an extended level of knowledge in different application domains. The user interaction capabilities of these systems are already sophisticated and hence these systems act as effective "user assistants" by providing different types of information to assist the users in performing their tasks. However, domain knowledge and expertise are still requirements for the user. The next generation of these computerized systems should be even more sophisticated by incorporating the required expertise as part of the system's functionality. This means a shift from "user-assistant" to "user-consultant". Therefore, instead of expecting the user of a SOA service to know significant amount of details and operational steps of using a specialized web application, the web service itself acts as an expert that consults with the user to provide an effective and customized use of the operations according to the user's specific context information. This provides an opportunity for the user to employ an expert (as Web Personalizer) to manage their web assets and perform desired tasks with minimum effort and time. Such a Web Personalizer provides smart interactions and ease of use for the user. The "Web Personalizer" agents are resident (as opposed to mobile agents that can move among different platforms) with customizable architecture that receive a set of well-defined task information in order to become expert and serve the user. The proposed Web Personalizers will be additions to the currently available services (i.e., traditional services), which receive a client's request for a service, perform the service at the provider's platform and return the results to the user.

Architecture of Web Personalizer

A Web Personalizer is a collection of three generic agents, namely "*service-agent*", "*broker-agent*" and "*analyzer-agent*" that are used in the context of an extended SOA architecture. Based on the traditional SOA reference model, service providers publish the descriptions of their services in the service registry. The service clients search the service registry to discover and connect to the select services. In this context, a service-agent represents a real-world delegate from the business owner to serve the client; a broker-agent and an analyzer-agent represent the client's delegates that communicate with the service provider on behalf of the client. Before being operational, these generic agents must be customized using three-part messages as <model, knowledge, data>. The model part is in the form of a business process or guideline that is intended to specialize the agent to act a specific role (e.g., financial advisor, sales person, home nurse, tester, auditor, or trace collector). The knowledge part is in the form of business rules and actions that provide the required expertise for the agent to perform the assigned role. The data part provides contextual information to be used by the business rules and actions. The structure of a generic agent consists of blocks such as "task manager", "business process engine", and "communication gates". The task manager receives the model part (hence it adopts a specific role) to control the sequence of the assigned business process. The business process engine engages the expertise by applying business rules and performing business actions. The communication gates are used to import the user's context data and to return the results of the desired operation to the client. Such a layered configuration for a generic agent provides ease of use and flexibility of configuration to perform different tasks for different users. It is assumed that different agents of the Web Personalizer have already been deployed at the intended platforms, i.e., service-agent, broker-agent, analyzer-agent in the client, broker, and provider platforms, respectively. In the followings, the steps for utilizing the Web Personalizer are presented.

Step 1: *Identifying the context of the user*. In service provisioning, *context* refers to any information that can be used to characterize the situation of a service requester or provider. We define a context as a tuple: <User, Role, User Location, Server Location, Time of Day, Team, Delegation, Requested Profile Status, Service Invocation Type, Requested Data Type, Login/Logout Event>. This context information is monitored dynamically to feed a database of context-logs which will be used during the service selection.

Step 2: *Selecting the required task*. The user asks for a specific task and the required expertise to assist her. Through mining of the context-logs (Step 1) and consulting with a web registry, a client proxy obtains a list of relevant services that provide different levels of expertise in that task, and generates a ranking lists of them with their capabilities and the associate charges. The user then selects an appropriate service, which best matches with their situation. In this context, the web registry should possess a list of application domains such as: banking, insurance, healthcare, telephone, airline, government, etc.; as well as a list of expertise within each domain, such as: mortgage consulting, car insurance negotiator, virtual nurse, TV technical support, on-line ticket reservator, PHR viewer, medication administrator, financial assistant, etc.

Step 3: *Delegating expertise to the client*. After selecting the required task, the client proxy retrieves the service descriptions of the selected service and invokes the service from the provider's platform. Instead of performing the requested task for the client, the provider will send a tuple of <model, knowledge, data> to the client where the generic service-agent will receive the tuple and customize itself (as discussed above) to become an expert consultant for the user to help them in an interactive mode of operation.

In the followings, the characteristics of the three agents of the Web Personalizer are discussed.

Expert Service Agent

The service-agent is a generic agent that is installed at the client's platform and is personalized as described in Step 3 above. It is the closest agent to the user in the proposed Web Personalizer and provides an expert service to the user. The service-agent ensures that the user will take advantage of the full capabilities of the available service functionality by adjusting its service locally and according to the user's context information. There are several advantages in processing a service locally as opposed to performing it at the service provider's platform: i) client data is confidential and revealing it to a service provider violates the client's privacy and security; ii) client data is usually large or changes over time, thus transferring it to a service provider increases the required network traffic; and iii) in the case of real-time or time-critical services, transferring client data to a service provider increase the response time. As an examples of a service-agent consider the case of a financial adviser in the context of stock market. To give proper advice, a financial adviser usually asks for personal information from their clients (e.g., client's portfolio or budget). By employing a service-agent to personalize a general advice for the client, the client does not have to reveal their personal information and hence their privacy is maintained [1]. Other example would be using a service-agent to assist a patient with a chronicle disease by following a flow-based guideline that has been prescribed by the physician.

Customizable Broker Agent

This agent is optional in the proposed Web Personalizer. It is meant to evaluate a set of candidate traditional web services in order to select for usage or aggregate with other services. The user will select from a list of high-level quality features, such as performance, security, or availability, which causes the required expertise be sent to the generic broker-agent. In this context, the web registry maintains a list of analysis expertise that is available for the user to choose from. The selected expertise will be sent to the generic broker-agent to customize it for the intended service evaluation operation. The broker-agent will then issue a number of service invocations to the candidate services that are provided by the user. The results will be sent back to the broker-agent, where it uses an objective function with parameters that are defined based on the client's contextual information. The broker-agent will then send back a list of ranked services to the user with a short report of the merits or drawbacks of each service. JaxView [4] provides a list of monitoring and security services that are based on monitoring the service traffic between the service client and service application. The broker-agent can enforce the business rules on the overall transactions between the client and server to maintain the client's integrity. This is done by ensuring the correct sequences of messages and checking the message contents.

Customizable Analyzer Agent

A generic analyzer-agent is intended to provide more in-depth analysis information for a customized broker-agent to perform sophisticated service quality analysis than those currently used for service selection and service aggregation. In this context, the broker-agent customizes the generic analyzer-agent, located at the service provider platform, to instrument the service application by embedding binary code into the service so that the analyzer can collect execution traces that belong to the broker-agent. The collected traces are returned to the broker-agent where different dynamic analyses can be performed on the execution traces such as security flaw identification or feature localization. For example, the analysis of patterns of execution traces using sequential pattern mining provides knowledge about feature scattering among the services. However, these analyses are intrusive and require the consent of the service providers. Such

analyzers can also be specialized to act as policy monitoring, auditing, and testing agents by returning a variety of information from the SOAP messages. We are currently pursuing a research that uses analyzer-agents to perform dynamic analysis of web services based on pattern mining of the execution traces.

Domain Interoperability and Cross-Domain Interoperability

An important requirement for the proposed Personal Web infrastructure is the capability of domain and cross-domain interoperability. Such an infrastructure will allow the proposed Web Personalizers to integrate web services from legacy and new applications, that are deployed in heterogeneous platforms, and are located in the same or different application domains. Such interoperability requires a common information domain and a common semantic terminology system to allow different applications communicate and understand each other.

Domain Interoperability [2]. The migration of legacy web applications to a new architecture requires data and service reverse engineering to understand the structure and organization of the information system, comprehend the available standards for data and services representations, and perform the actual mapping. To tackle the complexity of network-centric interoperability, the trend is towards ease of use, vendor / language / platform independency, and in general raising the level of communication abstraction. The available technologies at different levels of abstraction such as: RPC/RMI (low-level), CORBA / DCOM (proprietary), XML and web services (non-proprietary), and enterprise level (SOA) to a large extent have solved the problem of interoperability of heterogeneous distributed systems. However, another challenge to be tackled is the interoperability of terms and concepts between different information organizations, namely semantic interoperability. This necessitates a different type of communication abstraction, which reduces the task of users from making frequent agreements on different concepts and services that they use locally. This issue can be resolved through standard terminology systems and services for exchanging specialized applications and workflows. Such interoperability issues have been extensively dealt with in the healthcare domain. Due to sensitivity of information in healthcare systems (mostly patient data) and a huge collection of terms and concepts, achieving interoperability among healthcare systems faces problems beyond those in most domains. Moreover, the growing cost of healthcare in most countries has been the driving force to put a lot of efforts in this domain for defining standardization processes. These efforts have resulted in developing international standards in common terminology systems (e.g., SNOMED and LOINC) and in organizing and representing the whole body of domain information into class diagrams (i.e., HL7 RIM), as well as a large set of standard message interactions to allow seamless communication among applications (i.e., HL7 v3 messaging). Therefore, any new attempt for interoperability among legacy healthcare systems should conform to these standards to ensure its compatibility and maintainability in the future. However, there are some problems in adopting HL7 v3 messaging standards due to its complexity and large message sizes. We have developed a prototype environment (MacSeie: McMaster Service-Based eHealth Integration Environment) for integrating two systems: a prototype electronic medical record system (EMR) that we developed, and an existing clinical decision support system (CDSS) which provides a collection of algorithms that use the patients medical history, current medications, allergies, and vascular risk and generates messages which contain recommendations for the patients and physicians. The MacSeie environment is web service based and HL7 v3 compliant.

Cross-Domain Interoperability [3]. Currently, most research activities are focused towards standardization and interoperability among information systems within the same domain. However, an emerging challenge is to address the exchange of information among heterogeneous applications in different domains, such as healthcare and insurance. Cross-domain interoperability

requires extensive and in-depth analysis of the information domains and semantic models of the corresponding domains in order to develop a new set of standards for communication.
The Healthcare Development Framework (HDF) specification is a product of the HL7 Modeling and Methodology work group, and defines the general methodology for developing messaging standards. The HDF is aimed at achieving interoperability within the healthcare domain using a common set of message elements. While provision of semantic interoperability within the same domain is a nontrivial research problem, it is even more challenging to provide interoperability among systems in two or more application domains. We have proposed a framework to achieve message-oriented semantic interoperability across global application domains such as healthcare and insurance to ensure the conformance of the resulting common standard with the local domain standards. The proposed approach is an extension to the HDF framework and is based on: comprehensive information and messaging model of RIM; hierarchical concept and terminology representation of SNOMED; and message development framework of HDF, all under well engineered standards of HL7 v3. The proposed approach provides the detailed description of three phases of the semantic interoperability framework consisting of "collaboration point analysis", "design harmonization", and "dynamic design" processes. This approach allows the domain and technical experts to analyze and engineer two application domain standards in order to extract business rules and message elements of the portions of the two domains which are subject to collaboration. A core standard information model has been adopted as a common information domain (CIM) where the information and concepts from two domains will be translated into the CIM. The most challenging part of the semantic interoperability provision is to identify and map between message elements of two XML message schemas with different structures. This mapping algorithm is interactive and ensures a precise mapping between two XML messages in terms of semantics of their tree nodes.

References:

[1]. A Framework for Context-Aware Services Using Service Customizer. M. Najafi and K. Sartipi. IEEE International Conference On Advanced Communication Technology. (ICACT 2010), Volume 2, pages 1339-1344. February 7-10, 2010, Phoenix Park, Korea.

[2]. Standard-based Data and Service Interoperability in eHealth Systems. K. Sartipi and M.H. Yarmand. IEEE International Conference on Software Maintenance (ICSM 2008), pages187 - 196. Beijing, China. September 2008.

[3]. Cross-Domain Information and Service Interoperability. K. Sartipi and A. Dehmoobad. ACM International Conference on Information Integration and Web-based Applications & Services (iiWAS 2008), pages 25-32. Nov 24-26, Linz, Austria.

[4]. JaxView: http://managedmethods.com/