A Framework for Context-Aware Services Using Service Customizer

Mehran Najafi, Kamran Sartipi

Department of Computing and Software, McMaster University, Hamilton, Ontario, L8S 4K1, Canada

najafm@mcmaster.ca, sartipi@mcmaster.ca

***Abstract*—** In order to call a context-aware service, a service requester has to reveal his contextual information to a service provider or a context manager, which may jeopardize his privacy and security. Moreover, if the client's context changes over time, a context-aware web service must be called frequently with the updated context values. In this paper, we propose a service customizer agent at the client's side and a service provider that generates general service responses and service customization knowledge. The customizer agent uses the knowledge to personalize the general service response based on the client's context. In addition to offering innovative context-aware services, the proposed approach improves the privacy and security features of web services.

Keywords: **Service-Oriented Architecture (SOA); Context-Aware Services; Software Agent; Knowledge Management; Service Customizer.**

I. INTRODUCTION

Service-Oriented Architecture (SOA) [1] is a high-level and technology-independent concept that provides architectural blueprints for enterprise systems. SOA based architectures focus on dividing the enterprise application layer, where its components (as services) have a direct relationship with the business functionality of the enterprise. Web service, which is based on message-exchange, is the most widely adopted SOA technology.

In service provisioning, *context* [2] refers to any information that can be used to characterize the situation of a service requester or provider such as client's identity, location, and resources, as well as service time. Context-aware services [3] are services that can adapt the provided results to changing context information. Since mobile devices (such as PDA and cell phones) have become more powerful, it is extremely important to include user's context in the service provisioning. Moreover, clients would like having personalized services such as adviser services that take client information and give proper advice.

Traditional context-aware frameworks include a context manager component that acquires contextual information from the service requesters and provides them to the service providers. Then, in order to receive personalized services, a service client has to reveal his context. Traditional approaches have several limitations to develop context-aware services such as follows.

- When a client's context is modified, a context-aware service must be called with the updated contextual information. Then, if the context is changing continuously (such as location), the service must be called several times that increases the network traffic as well as the cost of using the service.
- If a client's context is sensitive or confidential (e.g., client's personal information), sending them to a context manager (or a service provider) may violate user's privacy and security.
- When the requested contextual information is large, sending them to a context manager requires large messages.

Therefore, if a client does not want to (or can not) send his contextual information, the traditional approaches fail. To deal with this limitation, we propose a framework that allows clients to keep their context local and still receive personalized services. In this framework, there is a customizer agent at the client's side that has access to the client's contexts and customizes service responses based on them. In order to employ the service customizer, a service provider needs to generate general service responses as well as customization knowledge.

The organization of this paper is as follows. Previous work is discussed in Section 2. The proposed framework and its details are discussed in Section 3. A case study in the business domain is explained in Section 4. Finally, conclusions and future work are discussed in Section 5.

II. PREVIOUS WORK

A generic architecture for context-aware services is introduced in [4] (Figure 1). The specified architecture has four layers as follows. Physical entities layer includes a set of sensors and input devices to capture the context from the environment. Then, it classifies this information to allow the services to request context information. Finally, the context management layer stores the classified context in a database. The service-provisioning layer is responsible for the service

discovery, composition and execution based on the contextual information. The forth layer is application layer that supports users to develop context-aware services.

Based on the generic architecture, several context-aware service frameworks have been proposed. The Akogrimo framework [5] provides data, knowledge, and computational services for mobile users on the Grid. Based on a context model, CA-SOA [6] proposes different components to support context-aware discovery and access Web services. In [7], a framework that supports the development of context-aware adaptable Web services is introduced by separating clients/web services from the context framework.

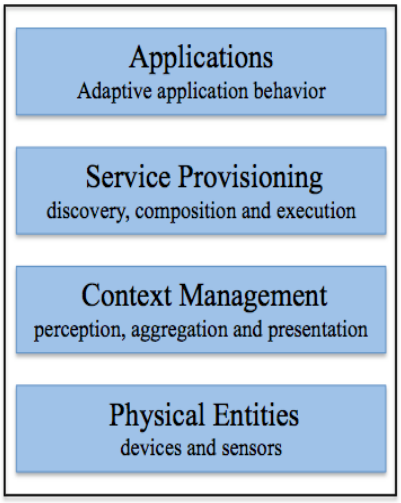

Figure 1. Generic architecture for context-aware service provisioning [4]

While security and privacy issues are highly related topics in context-aware systems, these issues have not been well addressed in context-aware Web services [8]. These issues are important when the context information is stored in centralized data store. Akogrimo and CA-SOA support security (based on authentication and authorization techniques), and limited privacy (privacy policy is used to indicate which consumer may require which types of contextual information). However most context-aware systems (e.g., CoWSAMI [9] and ESCAPE [10]) lack the security and privacy enforcement.

A software agent [13] is a piece of software that acts on behalf of an agency for a user. Software agents have been employed to facilitate developing context-aware services. ACAI [14] is an infrastructure that allows context information to be collected, processed, inferred, and disseminated by integrating software agents into context-aware services layer. The context management agent, the coordinator agent and the ontology agent are considered as the core of ACAI's multi-agent system. BerlinTainment [11] and AmbieAgents [12] are the other agent-based service frameworks proposed for context-aware service provisioning.

The discussed frameworks are based on context managers that reside outside service clients (i.e., context provider). As the result, they may violate client's privacy and security. In the proposed framework, we use a software agent that resides at the client's side to provide personalized services.

III. PROPOSED FRAMEWROK

The proposed approach works as follows: based on a received client's request, a service provider generates a general service response (i.e., not specific to a particular context) and relevant service customization knowledge. This information is sent to a service customizer agent at the client's side that has access to client's local data including client's contexts. Then, the customizer agent personalizes the service response based on the customization knowledge. The proposed framework whose components is specified in the following subsections is illustrated in Figure 3

A. Service Requester

Each service requester consists of a client application and a communication channel as follows.

Client application. It is a traditional client application that generates and sends request messages to service providers, where the request message does not include client's contextual information.

Communication channel. It allows the service customizer to access to the client's contextual information. In this framework, service providers should specify the required contextual information for their services, in the service registry. This channel consists of a number of ports, which are connection links to internal resources in the client application. In fact, a requester grants permission to the customizer agent to read/write a number of its resources through this channel. The ports can be input, output, or input/output. The agent sensors can read input ports and output ports can be written by the agent effectors. One instance of this channel is shown in Figure 2.

Figure 2. Example of communication channel

B. Service Provider

In contrast to existing SOA-based systems whose response messages have only one segment, the proposed model introduces service responses with two segments: general response and customization knowledge. Accordingly, the service provider consists of two layers that are designed to work independently and each layer is responsible for providing one segment of the response message.

Functional layer. It receives the request message; then the business engine, which applies business related functions on the business data (stored in the business data base), generates a general service response for the request. It should be mentioned that the proposed approach is useful in cases where the general service response (i.e., customizable based on a client's context) can be generated efficiently.

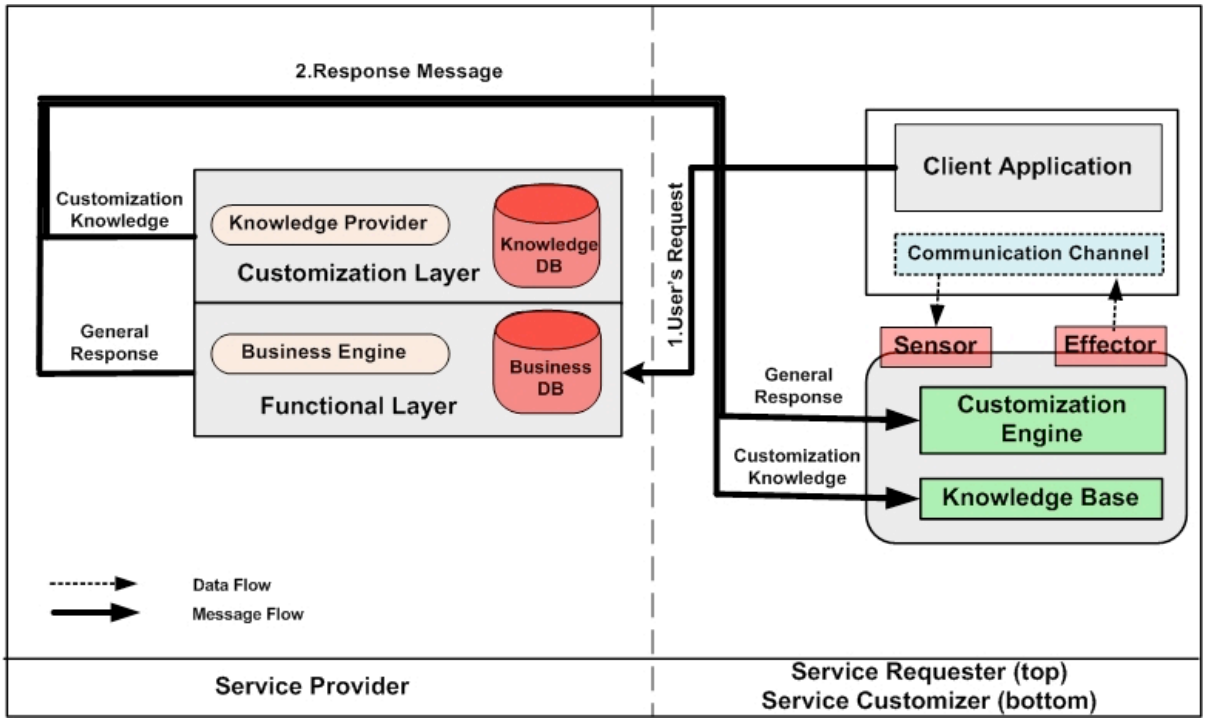

Figure 3. Proposed architecture: the client application sends a request message to the service provider; the service provider employs the service customizer to personalize its general service responses for the client.

Customization layer. It generates required customization knowledge for a service customizer agent to personalize the generated general service response. Customization knowledge is defined as a set of knowledge sentences where each sentence defines how a number of client's contextual information modifies the general service response. Different types of knowledge sentences can be used such as rule based, (if-then statements) and model based (mining-model parameters). For example, a rule-based knowledge sentence can be defined as:

If ***condition*** *(context)*
Then response' = ***modify*** *(response, context)*

This knowledge sentence states that if the defined condition among a number of contextual information (*context*) is true, the personalized response (*response'*) is obtained by applying the modification function on the general service response (*response*) and the client's context. In this case, the customization knowledge defines the condition and the modification function. As another example:

If ***relation*** *(response, context)*
Then response' = ***modify*** *(response)*

Similarly, it states that if a relation is true between general response and the contextual information, the modification function gives the customized service response. Finally, the generated knowledge sentences are put in the customization knowledge segment of the response message (in a defined order) and the message is sent to the service customizer.

C. Service Customizer

Service customizer is a software agent that has a one-way connection with the service provider, then it cannot transfer client's context and client's privacy is maintained. It has the following components.

Sensors and Effectors. Sensors provide client's context for the service customizer by reading the communication channel and effectors return the customized service response to the client application by writing into the communication channel. They will be connected to the corresponding ports, based on the channel description published in the service registry.

Knowledge Base. It receives and stores the customization knowledge segment of the response message to relive the service provider to send the knowledge in future calls. The customization engine retrieves its required knowledge from the knowledge base.

Customization Engine. It takes each received knowledge sentence (in the defined order) as well as the relevant client's context and then modifies the general service response. The final modified service response is returned to the client application as the personalized service response. For each type of knowledge sentences, one customization engine can be developed. For example, to work with the rule-based sentences, the customization engine must know how to take and apply if-then statements or model-based sentences can be applied by constructing the corresponding mining models in the customization engine.

Table 1. Customizing knowledge for personalizing general advice

#	Customization Knowledge
1	**If** Advcice . Type = Buy ∧ Advice . Price > Client.Budget **Then** Advice . Valid=False
2	**If** Advcice . Type = Sell ∧ Advice . Stock ∉ Client.Portfolio.Stocks **Then** Advice . Valid = False
3	**If** Advcice . Confidence < Client . RiskPreference **Then** Advice . Valid = False

IV. CASE STUDY

In this section, we present a case of 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 customizer to personalize general advice, a client does not have to reveal his personal information and hence his privacy is maintained.

We implemented a prototype system of such a financial adviser, as follows. A client sends a request (without providing any personal information) to receive financial advice. The adviser's response contains two parts as follows.

- General financial advice
- Customization knowledge to personalize the general financial advice based on the client's personal information

A. Case Study Specification

The process of generating financial advice could be very complicated and is out of scope of our discussion. In this case study, we are interested only in the personalization procedure, as below.

Each general financial advice is in the form of a tuple with three elements <Buy/Sell, Stock-name, Confidence-level> where Confidence level is in the range [0.0 ... 1.0]. Also, related contextual information include: client's portfolio, client's budget, and client's risk preference, where the risk preference can be A, B, C, or D such that A and D represent client's maximum and minimum risks. The service customizer can modify the general financial advice for the client as follows.

- Eliminate a Sell advice whose corresponding Stock name is not available in the client's portfolio.
- Eliminate a Buy advice that requires more funds than the client's budget.
- Eliminate any advice whose Confidence level violates the maximum risk tolerance of the client.

B. Service Requester

The client application sends a request to receive financial advice. Moreover, it puts its contextual information and stocks price into the communication channel (Figure 4), based on the communication channel description published in the service registry. There is also a port to receive customized financial advice from the customizer agent.

Figure 4. Communication channel in the financial adviser

C. Service Provider

Functional layer is either an automated system or a financial expert who generates financial advice in the specified format of a tuple discussed in Subsection A. Moreover, a flag (valid) is considered for each general advice to show whether the advice is in the final set of personalized advice. This flag is initially set to true. These advice is put into the general response segment of the service response message. Customization layer puts the customization knowledge sentences (represented in table I) into the knowledge segment of the response message where their order is determined based on their number.

D. Service Customizer

Sensors are connected to the 'read' ports to provide client's contexts and resources as inputs for the customizer engine. The customizer engine takes the if-then rules (in the specified order) and applies to the received general advice. Then, it modifies the "valid" field of general advice. After applying the third rule, the agent's effectors writes advice whose 'valid' field is true to the customized advice port of the communication channel as the final service response.

E. Implementation Notes

We implemented a prototype of the financial adviser via NetBeans IDE 6.5, which uses GlassFish V2 as application server, and J2EE 1.4 to develop web services. The XML schema of messages, which are exchanged between the service requester and the service provider, are illustrated in Figures 5 and 6.

The request message is short and just contains the request message without including any contextual information. The response message contains both general advice and customization knowledge tags. By using these schemas, WSDL description of the web service was developed. In addition to providing a secure financial adviser, the implemented service customizer can be used by different service providers to offer other context-aware services when they provide customization knowledge in a well-defined format that is supported by the customization engine.

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://xml.netbeans.org/schema/RequestMessage"
        xmlns:tns="http://xml.netbeans.org/schema/RequestMessage"
        elementFormDefault="qualified">
   <xsd:element name="RequestMessageDocument" type="tns:RequestMessage"/>
   <xsd:element name="RequestMessage" type="xsd:string" fixed="daily financial advice" />
</xsd:schema>
```

Figure 5. The request message schema

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://xml.netbeans.org/schema/ResponseMessage"
        xmlns:tns="http://xml.netbeans.org/schema/ResponseMessage"
        elementFormDefault="qualified">
<xsd:element name="ResponseMessageDocument" type="tns:ResponseMessage"/>
<xsd:complexType name="ResponseMessage">
  <xsd:sequence>
   <xsd:element name="GeneralResponse">
    <xsd:complexType>
     <xsd:sequence maxOccurs="unbounded">
      <xsd:element name="Type" type="xsd:string"></xsd:element>
      <xsd:element name="Stock" type="xsd:string"></xsd:element>
      <xsd:element name="Confidence" type="xsd:double"></xsd:element>
      <xsd:element name="Valid" type="xsd:boolean" fixed="True"></xsd:element>
     </xsd:sequence>
    </xsd:complexType>
   </xsd:element>
   <xsd:element name="CustomizationKnowledge"type="tns:RuleBase">
    <xsd:complexType>
     <xsd:sequence>
      <xsd:element name="Rule1" type="xsd:string" order="1"
      fixed="if advice.type=buy and advice.price gt client.budget then advice.valid=false">
      </xsd:element>
      <xsd:element name="Rule2" type="xsd:string" order="2"
      fixed="if advice.type=sell and advice.stock not member client.portfolio.stock then advice.valid=false">
      </xsd:element>
      <xsd:element name="Rule3" type="xsd:string" order="3"
      fixed="if advice.confidence less client.riskpreference then advice.valid=false">
      </xsd:element>
     </xsd:sequence>
    </xsd:complexType>
   </xsd:element>
  </xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

Figure 6. The response message schema

V. CONCLUSIONS AND FUTURE WORK

As opposed to using external context manager, the proposed framework keeps the client's context at the client side. Moreover, in contrast with the existing implementations of SOA services that require sending the entire agent (e.g., by using serialization techniques in mobile agents) to the requester site, our approach requires a short message with a minimum amount of knowledge to customize the service responses.

For future work, we will apply the proposed framework on more dynamic contextual information (e.g., location) while we are concentrating on mobile devices as service clients. We are also working to provide a formal model for the proposed service customizer. And finally, in addition to improve privacy and security features of services, the proposed idea introduces innovative context-aware services that finding and developing them will bring more opportunities for e-services to be used by enterprise companies.

REFERENCES

[1] D. Krafzig, K.Banke, and D.Slama. "Enterprise SOA: service oriented architecture best practices," Prentice-Hall, 2005.

[2] A. Dey and G. Abowd, "Toward a better understanding of context and context- awareness," Technical Report, Georgia Institute of Technology, 1999.

[3] G. Kapitsaki, G. Prezerakos, N. Tselikas, and I. Venieris, "Context-aware service engineering: A survey", Journal of Systems and Software, vol 82, Issue 8, pp. 1285-1297, 2009.

[4] S. Mostafaoui, and B. Hirsbrunner, "Context-aware service provisioning," IEEE/ACS International Conference on Pervasive Services, pp. 71-80, 2004.

[5] F. Solsvik. "D4.2.3: Final integrated services design and implementation report Akogrimo", .http://www.akogrimo.org, 2006.

[6] I. Chen, S. Yang, and J. Zhang, "Ubiquitous provision of context aware web services". Proceedings of the IEEE international Conference on Services Computing, pp. 60-68, 2006.

[7] M. Keidl, and A. Kemper, "A framework for context-aware adaptable web services". Advances in Database Technology - EDBT 2004, pp. 826-829, 2004.

[8] S. Dustdar, and H. Truong, "A survey on context-aware web service systems", International Journal of Web Information Systems. Vol. 5, no. 1, pp. 5-31. 2009.

[9] H. Truong, L. Juszczyk, A. Manzoor, and S. Dustdar, "ESCAPE - an adaptive framework for managing and providing context information in emergency situations," Second European Conference on Smart Sensing and Context, pp. 207-222, 2007.

[10] D. Athanasopoulos, A. Zarras, V. Issarny, E. Pitoura, and P. Vassiliadis, "CoWSAMI: Interface-aware context gathering in ambient intelligence environments," Pervasive Mobile Computing, Vol 4, no 3, pp. 360-389, 2008.

[11] J. Wohltorf, R. Cissee, A. Rieger, and H. Scheunemann, "BerlinTainment - an agent-based serviceware framework for context-aware services," 1st International Symposium on Wireless Communication Systems, pp. 245-249, 2004.

[12] T. Lech, and L. Wienhofen, "AmbieAgents: a scalable infrastructure for mobile and context-aware information services," In Proceedings of the Fourth international Joint Conference on Autonomous Agents and Multiagent Systems, Netherlands, pp. 625-631, 2005.

[13] H. Nwana. "Software agents: an overview", Knowledge Engineering Review, Vol 11, no 3, pp. 205-244,1996.

[14] M. Khedr, and A. Karmouch, "ACAI: agent-based context-aware infrastructure for spontaneous applications," Journal of Network and Computer Applications, Vol. 28, no 1, pp. 19-44, 2005.