Personalized Access to Semantic Web Agents Using Smart Cards

Riza Cenk Erdur[1] and Geylani Kardas[2]

[1] Ege University, Department of Computer Engineering,
35100 Bornova, Izmir, Turkey
erdur@staff.ege.edu.tr
http://bornova.ege.edu.tr/~erdur
[2] Ege University, International Computer Institute,
35100 Bornova, Izmir, Turkey
geylani@bornova.ege.edu.tr
http://ube.ege.edu.tr/~kardas

Abstract. In this paper, we mainly focus on the integration of smart card based access to semantic web enabled multi-agent systems. Besides classical benefits such as smart card based authentication and authorization, integration of such a feature will make it possible for semantic web agents to take the personal knowledge stored as instances of a specific personal ontology in the smart cards into account and behave in a way that is more responsible to the individual requirements of the users. To integrate smart card based access to a semantic web agent, we need an agent plan specifically defined for that purpose. This plan will be responsible for both communicating with the smart card reader module and for semantically manipulating the personal knowledge that is transferred from the card. In the paper, we give the implementation level details for this plan. Another important aspect of the paper is that various alternatives for storing ontological knowledge on smart cards have been discussed based on some experimental results.

1 Introduction

Semantic web [2] aims to transform the World Wide Web into a knowledge representation system in which the information provided by web pages is interpreted using ontologies. This gives the opportunity for autonomous and interacting entities - semantic web agents [6] - to collect and interpret semantic content on the behalf of their users. On the other hand, smart card technology has paved the way for an individual to carry personal information in a small card with storage, data processing and security features [8].

By the marriage of agent, semantic web and smart card technologies, in addition to classical benefits such as smart card based authentication and authorization for accessing agent systems, semantic web agents can be accessed using smart cards that store users personal knowledge as instances of a specific personal ontology. By this way, agents can take the personal knowledge stored

J.C. Cunha and P.D. Medeiros (Eds.): Euro-Par 2005, LNCS 3648, pp. 1110–1119, 2005.

in the card into account in adapting their behavior to act in a way that is more responsible to the individual requirements of their users.

Here, we will describe a scenario to illustrate an example case where enabling smart card based access to semantic web agents may have potential benefits for the users: Let us think of a multi-agent system, which has been established to provide tourism related facilities in a specific geographic area. If smart card based access to this multi-agent system is provided via card terminals located in several places such as airports or various locations in city centers, then a newly arrived traveler without a pre-built travel plan can have the chance of discovering the hotel facilities that best matches against her personal knowledge stored in her smart card. In addition, after discovering the hotel facility, it will also be possible for the traveler to make an online reservation from the terminal using the credit card knowledge that is defined and stored as part of her personal knowledge. We believe that this scenario is valid, because there are always travelers around without a pre-built travel plan.

A semantic web enabled multi-agent system infrastructure that can be used in realizing the above scenario is shown in Fig. 1.

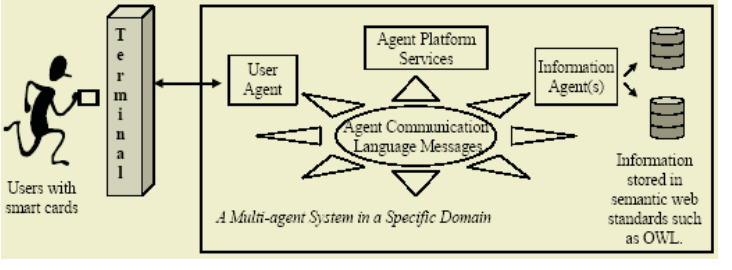

Fig. 1. The basic infrastructure for smart card based access to a multi-agent system

As shown in Fig. 1, the user agent in the multi-agent system must have the functionality to deal with the smart card based accesses. The user agent also must have the capability of understanding ontologies, since it has to understand the personal ontology instance that is transferred from the users smart card. In the multi-agent system, information is represented using semantic web standards such as OWL [12]. Hence, the information agents have the capability of manipulating semantic knowledge and answering queries over that semantic knowledge. Agent platform services, which are agent management, agent directory, and agent message transport services, are standard services that a platform, on which the multi-agent system is operating, has to provide. These services are usually provided as built-in services by the multi-agent development framework/ platform used.

In this paper, we mainly focus on the integration of smart card based access to the agents in a semantic web enabled multi-agent system. To achieve such kind of integration, we have to define a basic agent plan for that purpose. This

agent plan is needed both for communicating with the smart card reader modules and for manipulating the transferred personal knowledge. Manipulation of the transferred personal knowledge includes parsing it, constructing the ontology model in memory, and preparing requests for querying the knowledge stored in information agents.

In smart card technology related literature, there are several studies covering the use of smart cards in Web applications, especially in medical healthcare systems [4]. However, there is no work discussing in detail either how smart card access support can be integrated to multi-agent systems or how ontologies are stored in smart cards, transferred from them and manipulated in agents. This paper aims to fill in this gap by discussing the implementation level details.

2 Architecture for Enabling Smart Card Based Access

In this section, first, the generic agent plan needed for smart card based access to a semantic web agent is explained. Then, details concerning the implementation of this plan are given. Finally, the personal ontology, which is used in representing the personal knowledge of each user, is given and various alternatives about storing ontologies in smart cards are discussed.

2.1 A Generic Agent Plan for Smart Card Related Behavior

To behave in a way to satisfy what is expected from them, agents formulate plans. Each plan consists of a number of tasks that are scheduled and executed. Plans are represented using a planning formalism. Hierarchical Task Network (HTN) is the most frequently used planning formalism in the planner modules [11] of agent development frameworks. Hence, the plan component defined for smart card access in this paper will be explained based on the HTN approach.

HTN structure consists of nodes that represent tasks. Since a task may be composed of subtasks, the plan structure may take the form of a tree-like structure. There are two kinds of links in a HTN representation. Reduction links describe the de-composition of high-level tasks to subtasks. Provision or outcome links represent value propagation between task nodes [11].

In an agent, smart card access related behavior can be modeled as a composite HTN task structure that consists of three subtasks, which are card reading and session opening subtask, the ontology manipulation related subtask(s) and card writing and closing the session subtask. Fig. 2 shows the HTN structure of this composite task.

As shown in Fig. 2, the smart card access task is a complex task that consists of three subtasks. The first subtask is responsible for waiting for the smart card to be inserted into card acceptance device and opening a new session when the card is inserted. After the session is opened, knowledge is read from the card and this knowledge is passed to the second subtask via the provision Knowledge_FromCard. The second subtask is a complex task that represents the behavior related with the manipulation of the semantic personal knowledge. After

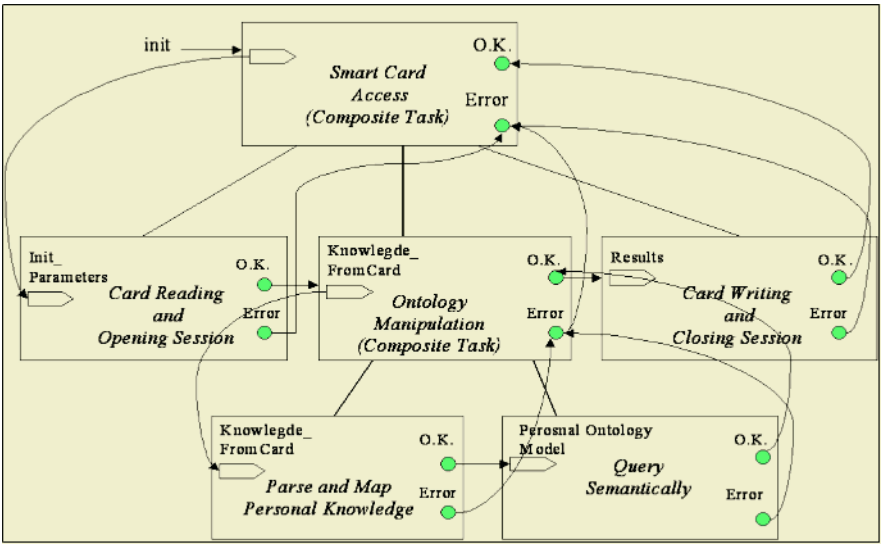

Fig. 2. HTN structure for smart card access behavior

the information agents are queried based on the concepts in the personal knowledge, incoming results are passed to the third subtask via the Results provision, which writes the necessary results back to the card and closes the session. We used a task for card writing, because we preferred the final results of a request to be written on users card so as to keep a history for the user.

2.2 Implementation Details for the Generic Plan

We need two basic components for implementing the smart card based access support plan completely. The first component is a middleware, which will enable an agent to communicate with the smart card. The second component is an API that is going to be used for the manipulation of ontological knowledge. In the following paragraphs, we discuss these two components.

In general, a smart card and an application communicate using a special protocol, which is called as Application Protocol Data Unit (APDU) and which is an application layer standard based on ISO 7816-4 [8]. In order to retrieve personal knowledge from the card, an agent needs to communicate with the smart card using APDU packages. However, we think that this task that is quite primitive and that integrating codes for pure smart card access will make an agents plan unnecessarily complicated. Hence, we have proposed a layered approach and using the OpenCard Framework (OCF) API [10] developed a middleware to handle smart card communications on behalf of the agents. OCF is an API that supports communication between the smart card host and the applications inside the card. It supports Java platform and is maintained by the OpenCard Consortium. Our middleware uses this API to control smart card events and manage

operations like card applet selection, cryptographic key exchange, receiving and sending of APDUs.

The user agent, which is the users entrance point to the multi-agent system, is responsible for manipulating the personal knowledge that is transferred from the smart card. Personal knowledge is an instance of the personal ontology. Personal knowledge is stored in the smart card in compliant with the semantic web standards, such as an OWL document. The user agents access this document over a secure and authenticated communication channel established between the smart card and the agent itself.

An agent handles the semantic knowledge by creating the resource model of the personal ontology and querying the model. For example, in tourism domain, a user agent using the personal knowledge should first determine its customer's preferences for hotel reservation and then based on these preferences prepare requests for querying the hotel information represented semantically in hotel agents. In our implementation, we have used JENA API [9] to supply reading, writing and semantic querying of ontological knowledge in agent components. JENA is a semantic web framework, which provides a programming environment for RDF, RDFS and OWL and includes a rule-based inference engine. It also ensures a query language for RDF called RDQL. Our agents use RDQL to query on the ontology model to obtain the desired semantic knowledge.

During the implementation of the multi-agent system, the generic HTN structure, for which the subtask implementation details are given above, is instantiated and executed based on the plan definition and execution model of the agent development tool that is being used. For example, if JADE [1] is being used as the agent development framework/platform, then the generic HTN structure will be implemented and executed in compliant with the behavior model of JADE. How the generic plan is instantiated using an agent development framework/platform will be clear in section 3, which includes a case study explaining the instantiation of this plan in JADE environment.

2.3 Personal Ontology Example and Storing Its Instances on the Card

As mentioned before, the personal knowledge that is stored in the cards should be an instance of a specific personal ontology. Such an example personal ontology consists of two main parts: The first part is the domain independent part, where user identification, contact and payment information are kept. The other part is the domain specific part and includes knowledge represented using a specific domain ontology. For example, if the smart card is intended to be used in tourism domain for travelers who want to discover hotels and make reservations, then the knowledge belonging to this part will be represented using a specific tourism ontology defining facilities about hotels and reservations. One such ontology is introduced in [7].

Various alternatives may be considered for storing ontological knowledge in smart cards. These various alternatives are discussed below for different cases by giving the possible advantages and disadvantages:

Alternative-1: Store the personal knowledge completely in the card.
Advantages for alternative-1:

i) The knowledge stored in the card can be transferred to the agent by taking the advantage of strong security and authentication features of smart cards.

Disadvantages for alternative-1:

i) The storage capacity of smart cards is limited. However, the personal ontology is not very complex in most cases and capacity would not be a serious problem with the application of the scaling down techniques. One such technique is introduced in [3].

ii) To investigate whether response time may be a problem as the ontology file sizes get larger, we have measured the time needed to read or write ontological knowledge and have seen that as the file sizes increase, the time needed to read from or write to the card increases linearly. Measurements have been realized using Gemplus GemXpresso 211/PK model Java cards with multi applet support and with 32K ROM, 32K EEPROM and 2K RAM. JavaCard Framework 2.1 has been used during on-card software development. Gemplus GCR410 serial card read/write device has been connected to a terminal PC with 9600 baud data transmission rate. The measurement results are shown in Fig. 3. As Fig. 3 shows, larger personal knowledge file sizes may cause delays in response time. One of the most important reasons for delays is that APDU packets are maximum 255 bytes length and this requires multiple packet exchanges for large file sizes increasing the response time. In addition, the time values in Fig. 3 also include the time needed to convert the byte stream to a Java file in the agent. Please note that the ontological knowledge is stored in byte-streams in the card, which then needs a conversion into necessary Java files/objects.

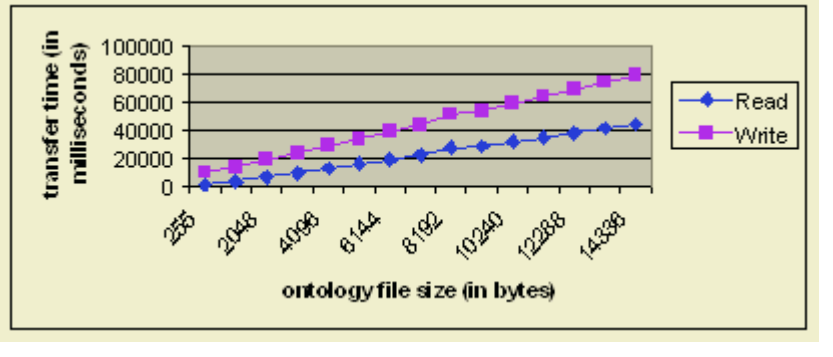

Fig. 3. Read/Write times for different ontology file sizes

Alternative-2: Store only the URL of the personal knowledge document in the smart card. In this case, the agent retrieves the URL of the ontology instance document rather than the document itself and then accesses the document over the Internet.

Advantages for alternative-2:

i) Obviously this approach decreases smart card communication time and shortens card sessions.

Disadvantages for alternative-2:

i) The security feature of smart cards cannot be used. The agent itself should manage the accessing privileges and security over the Internet. This may be a problem for the private part of personal knowledge, such as the payment information.

Conclusion: We have preferred the first alternative in our implementation although storing ontologies completely in the card may cause delays for especially large ontology files and especially for write operations. The reason for preferring the first alternative is that we think of taking the advantage of smart cards as a means of secure, portable and cheap personal knowledge storage media for accessing semantic web enabled multi-agent systems. In addition, ontology file transfer delays can be overcome using more efficient connection technologies between the card acceptance device and the host. In our experiment, the serial COM port has been used, which is another potential cause for the delays in transferring large ontology files.

3 Case Study: A Multi-agent System for Tourism Domain

As a case study, we have implemented a prototype semantic web enabled multi-agent system for realizing the infrastructure of the scenario given in the introduction section. There exist two types of agents in the developed system: hotel agents and customer agents. A hotel agent is an information type agent that represents a hotel in the system. A customer agent is an interface type agent that is responsible for searching suitable hotel rooms and making reservations based on the personal knowledge read from the card. The overall architecture of the system is shown in Fig. 4. The agents in the system have been developed using JADE [1], which currently is one of the most widely used Java based multi-agent system development framework/platform in the multi-agent literature.

The implementation consists of three Java packages, which are named as "`CustomerCard`", "`CustomerCardClient`" and "`Tourism`" respectively. The "CustomerCard" package includes on-card Java application, which is developed using Java Card Framework (JCF) [5]. This package is related with the smart card reader/writer unit connected to the host PC via COM port as shown at the left side of Fig. 4. The "CustomerApplet" class in this package is extended from the "javacard.framework.Applet" and it processes command APDUs sent from a customer agent. The "CustomerCardClient" package behaves like a middleware between the on-card applet and the customer agent. As it is shown in the middle part of Fig 4., this package resides on the same host with the customer agent that has smart card access and it supports functionalities such as smart card applet selection, APDU communication and hex-coded data package handling. The third package, which we named as the "Tourism" package, contains the agents in the developed multi-agent system, including the hotel agents and

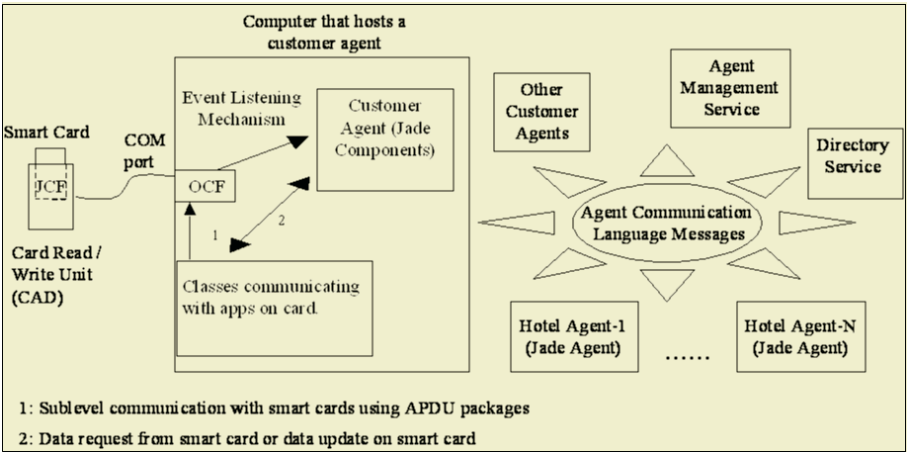

Fig. 4. A multi-agent system with a smart card access integrated customer agent

the customer agents. The agent management service maintaining life cycles of agents in a multi-agent system and the directory service providing yellow pages services to other agents are already supported by JADE platform.

The hotel agents and the customer agent are instances of the class "HotelAgent", and "CustomerAgent" respectively. These classes are derived from jade.core .Agent class and these agents behaviors are implemented using the behavior model of JADE. During initialization, a hotel agent registers itself to the platforms directory service as a Hotel Service provider. So, when a customer agent looks for a specific hotel room, it gets the hotel agents description from the directory service and can interact with this hotel agent. To be semantic web enabled, the RDQL [9] query language is supported. As mentioned before, JENA API has been used in semantic knowledge manipulations in all kinds of agents.

3.1 Instantiating the Generic Smart Card Access Plan in JADE Environment

The generic HTN structure given in section 2 can be modeled as a finite state machine (FSM) with transitions between subtasks. In JADE, there is a composite behavior subclass "jade.core.behaviours.FSMBehaviour" [1], which can be used to model behaviors as a finite state machine. For this reason, we have used JADE's FSM behavior to model the customer agents behavior as shown in Fig. 5. Each state of the FSM is implemented as a subclass of jade.core.behaviors. OneShotBehaviour.

Initially, the state is CARD_WAITING. After the card is inserted and a successful customer PIN entry is made, customer's personal knowledge is transferred from smart card and the RESERVATION_INIT becomes the new state. In that state, the customer agent searches for the descriptions of the agents providing hotel reservation services and updates its contact list based on the response from

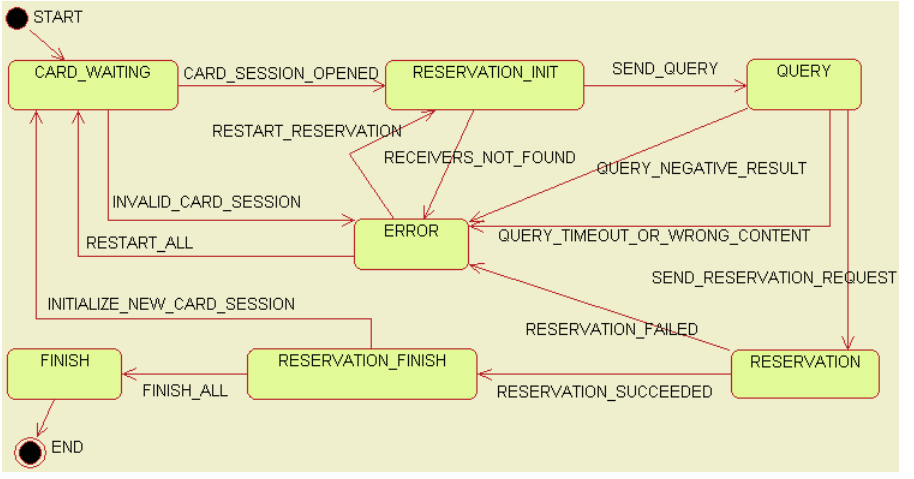

Fig. 5. Finite state machine of the customer agents behavior

the directory service so that the hotel agents that are going to be queried are determined. Now, the state becomes QUERY. In this state, the customer agent first constructs an ontology model for the personal ontology instance transferred from the card. Second, based on the concepts in the domain dependent part of users personal knowledge, the customer agent prepares requests, places them in agent communication language messages and sends them to each hotel agent determined before. Then, it waits for the replies from hotel agents. When a successful query result is returned from one of the hotel agents, the customer agent begins to behave in RESERVATION state and makes a second contact with the suitable hotel agent for reservation. Customer agent writes the reservation information to the customers smart card and finishes customers card session in RESERVATION_FINISH state. At this point, it can again wait for a new card session or completely terminates by calling doDelete() method in FINISH state. The errors during the execution of the behavior are managed in ERROR state of the behavior.

4 Conclusion

In this paper, it has been mainly discussed how to enable personalized access to semantic web enabled multi-agent systems using smart cards that store users' personal knowledge as instances of a specific personal ontology. For this purpose, a generic agent plan, which is responsible for communicating with the smart card terminal via a suitable middleware and for transferring and semantically manipulating the personal knowledge, is given. This plan has been tested with agents implemented on JADE platform. The reason for choosing JADE is that it is currently one of the well-known and widely used agent development frameworks/platforms for Java environments.

Another aspect that is discussed in the paper is the various alternatives for storing ontological knowledge in smart cards. As it is stated in the local conclusion part in section 3, we have preferred the approach of storing the personal knowledge completely in the card, because we wanted to take the advantage of smart cards as a means of secure, portable and cheap personal knowledge storage media for accessing semantic web enabled multi-agent systems.

References

1. Bellifemine, F., Poggi, A., and Rimassa, G.: Developing multi-agent systems with a FIPA-compliant agent framework. Software Practice and Experience, 31 (2001) 103-128.
2. Berners-Lee, T., Hendler, J. and Lassila, O.: The Semantic Web, Scientific American, 284(5), (2001), pp:34-43.
3. Bobineau, C., Bouganim, L., Pucheral, P.,Valduriez, P., PicoDMBS: Scaling Down Database Techniques for the Smartcard, VLDB (2000): 11-20
4. Chan, A.T.S.: Web-enabled smart card for ubiquitous access of patient's medical record. WWW '99: Proceeding of the eighth international conference on World Wide Web, 31 (1999) 1591-1598.
5. Chen, Z.: Java CardTM technology for smart cards architecture and programmers guide. Addison-Wesley, Massachusetts USA, (2000)
6. Dickinson, I.: The Semantic Web and Software Agents: Partners, or Just Neighbours?, AgentLink News 15, September, 3-6 (2004). http://www.agentlink.org
7. Dogac, A., Kabak, Y., Laleci, G., Sinir, S., Yildiz, A., Kirbas, S., Gurcan, Y.: Semantically enriched web services for the travel industry. SIGMOD Record 33(3): 21-27 (2004).
8. Hansmann, U., Nicklous, M.S., Schack, T. and Seliger, F.: Smart card application development using Java, Springer-Verlag, Berlin Germany, (2000)
9. JENA, A Semantic Web Framework for Java, http://jena.sourceforge.net/
10. OpenCard Consortium, OpenCard Framework 1.2 Programmer's Guide, IBM Deutschland Entwicklung GmbH, Boeblingen Germany, (1999)
11. Paolucci, M. et al., A Planning Component for RETSINA Agents, Intelligent Agents VI, LNAI 1757, N.R.Jennings and Y. Lesperance, eds., Springer Verlag, (2000).
12. Web Ontology Language (OWL), http://www.w3.org/2001/sw/WebOnt/