Agent-Based Interoperability for e-Government

Fábio Marques[1], Gonçalo Paiva Dias[2], and André Zúquete[3]

[1] ESTGA-UA/IEETA, Campus Univ. de Santiago 3810 – 193 Aveiro, Portugal
fabio@ua.pt
[2] ESTGA-UA/GOVCOPP, Campus Univ. de Santiago 3810 – 193 Aveiro, Portugal
gpd@ua.pt
[3] DETI-UA/IEETA, Campus Univ. de Santiago 3810 – 193 Aveiro, Portugal
andre.zuquete@ua.pt

Abstract. The provision of valuable e-government services depends upon the capacity to integrate the disperse provision of services by the public administration and thus upon the availability of interoperability platforms. These platforms are commonly built according to the principles of service oriented architectures, which raise the question of how to dynamically orchestrate services while preserving information security. Recently, it was presented an e-government interoperability model that preserves privacy during the dynamic orchestration of services. In this paper we present a prototype that implements that model using software agents. The model and the prototype are briefly described; an illustrative use case is presented; and the advantages of using software agents to implement the model are discussed.

1 Introduction

E-Government is defined as the use of ICT by government agencies to transform the relations with their clients (citizens, businesses and other government agencies) [14]. The need to deliver services and in a more effective way was a catalyst towards presenting solutions for transversally integrating business processes and, consequently, to the interoperability problem [11, 12]. This process often triggers the re-engineering of existing business processes in the Public Administration (PA), which adds extra complexity. Finally, several other aspects (legal, social and technical) that must be as well taken into consideration (see, for instance, [2, 8]).

Security is one of the main issues regarding interoperability. Trust, authentication and access control are issues that must be dealt carefully, mainly to keep sensitive information private. In [4, 5] it has been proposed a model that addresses the dynamic orchestration of services (ensuring interoperability) and security, in particular trust, authentication and privacy, to provide government services. According to the model, the PA consists of a set of heterogeneous entities with heterogeneous systems, which may deliver overlapping services. Each entity is autonomous, reactive, proactive and able to communicate with other entities. These are the same characteristics that are identified by Luck et al. in [3] for a software agent (SA). SAs also support distributed computing efficiently and allow the dynamic composition of services [10], two key characteristics of the model. Based on these reasons, and

S. Omatu et al. (Eds.): *Distrib. Computing & Artificial Intelligence,* AISC 217, pp. 561–568.
DOI: 10.1007/978-3-319-00551-5_67

assuming that an SA represents an entity, we believe that the SA is the appropriate technology for implementing a prototype for validating the referred e-government interoperability model.

However, besides verifying the accordance between the characteristics of the technology and of the entities comprising the PA, a set of questions remain: are SAs a solution to implement the proposed model? What are the problems that are raised by using this technology? With this paper it is our purpose to present an agent-based prototype that implements the referred interoperability model and to evaluate the advantages and pitfalls of using SAs to accomplish that task.

Our paper is organized as follows. The related work is presented in Sect. 2. In Sect. 3 we present the basic principles of the model and an illustrative use case. The agent based prototype is presented in Sect. 4. In Sect. 5 we discuss the utility of agents and we conclude the paper in Sect. 6.

2 Related Work

There are a few examples of the use of software agents for improving some e-government functionalities. In this section we will describe them briefly as a base for comparison with our proposal.

In [9] it is addressed the use of mobile software agents as a facilitator for people migration. In this platform, the software agent represents the user interacting with the PA. The remaining components allow interoperability and a way to communicate with the service repositories and with legacy systems of the PA.

In [7] it is proposed an Intelligent Agent Technology to support decision making on government agencies. This system consists on 3 types of agents: User agents; Service Manager agents; and PA agents. The User agents support the access to services. The Service Manager agents allow searching for the best suited service for the user. PA agents give support for the decision making of a manager.

In [13] it is presented a web-based intermediary for the management of workflows that integrate services. The approach used by the authors combine workflow engines with agents' technologies. In this model the workflow controls the logic of the service delivery and selects the appropriate agent to continue with the service, achieving, this way, the interoperability.

In [6] it is briefly described an agent-based architecture, implemented with Web Services, which has four components: Citizen agent; Portal agent; Ministries and agencies agents; and the Service Agents. The Citizen agent represents the citizen and communicates with the Portal agent. The Portal agent collaborates with the Ministry and agencies agents to search the appropriate service information. The Service agents participate in the delivery of the service.

In [16] it is proposed a structure based on multi-agent technology. Agents are grouped according to their duty and forming a hierarchical structure. There are three types of agents: Organization agents are responsible for decision making and for intermediary management; the Function agents can set or invoke tasks; and Resource agents manage the information resources and provide the services.

These models stress the suitability of agent technology to implement service orchestration and interoperability in e-government. The majority of the models use agents as a representation of clients (people) and of PA agencies. Our approach uses agents only to model the agencies that provide services, which simplifies service orchestration. In fact, agents, besides providing services, may also request services from other agents, since they have an active role in the orchestration process.

3 e-Government Orchestration and Interoperability

To understand the prototype it is important to describe the basic principles of its orchestration and interoperability model. The basic concept is fairly simple: the PA is composed by entities that deliver simple services and that work together to build complex services suitable to be requested by citizens. The services that each entity is able to deliver are registered in service repositories. The following principles apply:

- All services (simple or complex) are provided by entities (public or private). Entities that have Local Information Systems (LIS) provide simple services;
- Interoperability is achieved through the orchestration services. The complex services are delivered by Intermediary entities (which may or may not be real entities) that compose services, which are consumed by other government agencies.
- The management of the delivery of a complex service is not centralized. Entities may delegate the management of the workflow (the sequence of steps for delivering a service) or part of it to other entities. So, workflows are established dynamically and there is no prior knowledge of what entities participate in the service being provided. Thus, results of service requests might be produced without knowing to whom they will be delivered, causing confidentiality or privacy issues;
- Consequently, a result is kept by the entity that produces it until requested by another entity, to be consumed. Thus, the result is provided only when its addressee is known and confidentiality or privacy measures can be enforced. The knowledge about the existence of the result is transmitted through all entities that participate in the service delivery, allowing the result to be requested when needed. This principle solves the confidentiality and privacy issues raised before.

3.1 Illustrative Use Case – Trip to a Foreign Country

This use case is very simple: a citizen needs to travel to another country and he uses a travel agency to book the tickets and the accommodation and to deal with the visa request (see Fig. 1). Only the visa acquiring and usage is addressed in the use case. It was chosen to illustrate various characteristics of the model and, therefore of the prototype. Although this use case is based on a real scenario, the sequence of services defined is the result of re-engineering the actual service to take advantage of the model and thus does not directly corresponds to how it is delivered today.

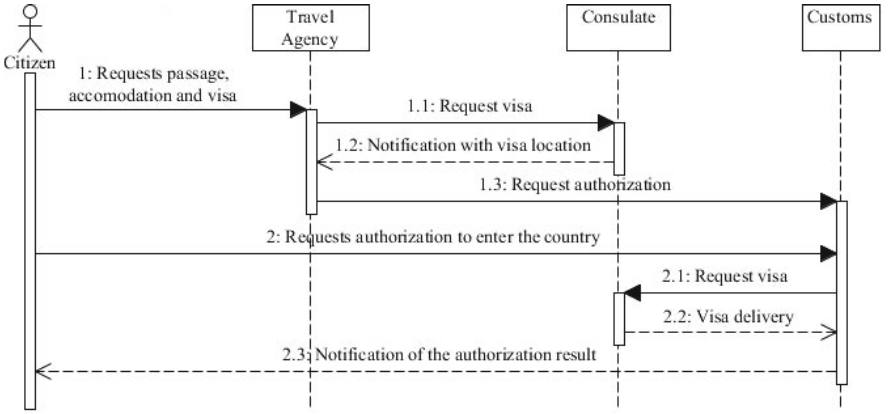

Fig. 1 Representation of the acquisition and presentation of a visa

After receiving the request from its client (1), the travel agency sends a request for a visa to the consulate of the country to which the client plans to travel (1.1). The consulate produces the visa and informs the travel agency that it is available in a given location (1.2). Then, the travel agency sends the information to the Customs of the country of destination as a request to admit its client in the country (1.3). Finally, when the client arrives at the country of destination (2), the Customs already has the information that enables it to acquire the visa from the consulate, which it does (2.1; 2.2). After receiving the visa, the Customs has the information it needs to assess if the person is allowed to enter the country or not (2.3).

In this use case there are private entities interacting with public entities which are from a foreign country. The process is asynchronous: the production of a visa is not instantaneous and the trip is booked in advance. The use case shows the utility of the basic characteristics of software agents: entities communicate amongst them (communication between agents); each entity does isolated work autonomously (autonomy); each entity reacts when a request arrives, producing some result (reactivity); and entities autonomously anticipate actions (pro-activity), an example being the sending of information from the travel agency to the Customs.

4 Prototype Implementation

The implementation of the prototype follows the model presented in [4]. The model is composed by: support services, which allow Certification and Service publication; support data structures, which define the data structures that are used to store the information for service provisioning; and by the messages that are used to communicate between entities. For implementing the architecture we used the Java Agent Development Framework (JADE).

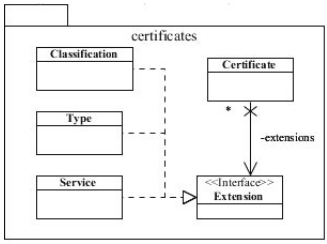

Fig. 2 Class diagram of the Certificate data structure

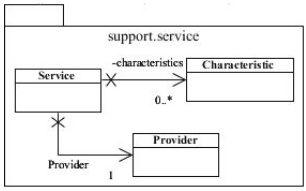

Fig. 3 Class diagram of the Service data structure

Support Services

There are three support services in the model: Time-stamping services; Certification services; and Directory services. The time-stamping services allow defining the date and time of message creation and sending, which allow the chronological assortment of messages exchanged between entities. The directory services allow the search, publication and removal of services. Finally, the certification services allow establishing, managing and issuing the certificates that certify what services an entity is able to provide and the services an entity is authorized to request.

Although these services are essential to support interoperability in the model, specially its dynamism and its cryptography functions, they were not deployed since they are fully validated technologies and are not essential for the proof of concept we desire.

Support Structures

There are two support structures in the model, both being based on open standards. The Certificate data structure (see Fig. 2), represents a certificate that is issued to an entity, is based on version 3 of the X.509 standard [1], and supports the identification, accreditation (by creating the extensions Classification, Type and Service) and authorization (by implementing the extensions Classification and Type) of entities with respect to services.

The Service Description data structure (see Fig. 3) is also based in an open standard: the Web Services Description Language (WSDL) [15]. This data structure allows the description of a service that is active and that is provided by an entity.

Both these structures were implemented and are used by the prototype. So, although the services repositories are not available in the prototype, the description of the services, described by the instantiation of these structures, are available to the agents that need it.

Messages

The model defines four types of messages (see Fig. 4). Each type has its own function: Request Service; Notify; Request Result; and Deliver Result. All the types were implemented in the prototype allowing the communication between agents and ensuring the correct use of the guiding principles of the model.

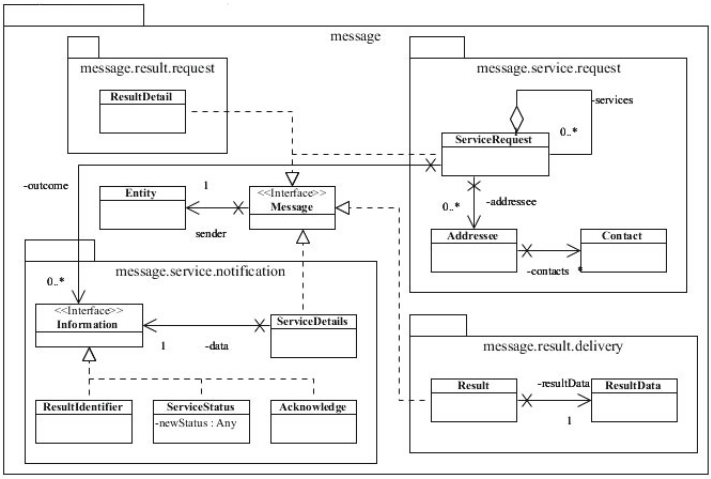

Fig. 4 Class diagram of the Message data structure

The messages also make use of the data structures defined in Sect. 4. The Message structure defines the language which is used by the agents to communicate. This structure is the common point between all messages types: the requested entity is identified and the timestamp is included. The only requirement is that agents must recognize and interpret messages with this structure.

Agent

One of the main components of the prototype is the agent. In the implementation we used the WADE extension of the JADE platform. The agent implementation was divided over four packages: the **workflow** package - includes the classes that define the tasks that an agent must execute and knows how to execute, it is used during the internal execution of the agent while the service is being provided; the **com** package - includes the classes that are responsible to build and to prepare the messages to send, likewise it includes all the necessary features to receive and validate the messages upon reception; the **security** package - contains the classes with the cryptographic functions and the assessment of accreditation and authorization; the **servicedeliverers** package - the class within this package represents a generic agent, each entity that is represented in the prototype is a subclass of this class.

Implementation of the Use Case

The use case presented (see Sect. 3.1) was deployed using three agents. Each agent (despite of the entity it represents) is an instantiation of a class that inherits the behavior of the class *BaseAgentGov*, which belongs to the *servicedeliverers* package. The differences between agents are materialized in the workflow, in the set of certificates and in the description of the services it offers.

5 Discussion

All the basic characteristics of software agents (see Sect. 4) are used in the prototype and proved to be important for the model implementation. The illustrative use case described in 3.1 take advantage of all these characteristics.

The characteristic that relates to the ability to communicate with other software agents is critical not only for the model itself, but even in a more basic way: to achieve interoperability. Only with this characteristic it is possible to delegate the management of the workflow and to require and provide services, which are crucial for service orchestration and for the presented interoperability model.

The reactive characteristic is an important feature with respect to the service provisioning since after the reception of the service request the agent knows how to deal with it and perform the necessary tasks to complete its execution.

The remaining two characteristics, autonomy and pro-activity, are essential to implement two of the main features of the model – its dynamism and concurrency. The concurrency is a feature that can only be achieved if there are two or more entities able to provide the same service. The dynamism of the model is only achieved if it is possible to know that a new entity is available or that an old entity was removed. The autonomy and the pro-activity allow the agent to be aware of its surroundings, being able to identify new entities and to identify services that are provided by different entities but with similar outcomes.

6 Conclusions

In this paper we presented an agent-based prototype that implements a model for e-government interoperability, an illustrative use case that demonstrate its deployment; and the advantages of using agents to implement the referred model.

We conclude that agent technology proves to be an appropriated choice to implement the tested e-government interoperability model, namely because of its key characteristics: interoperability, service orchestration, dynamism and concurrency. No major pitfalls were identified.

References

1. Cooper, D., Santesson, S., Farrell, S., Boyeen, S., Housley, R., Polk, W.: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280. Internet Society (2008), http://tools.ietf.org/html/rfc5280
2. Gugliotta, A., Cabral, L., Domingue, J.: Knowledge Modelling for Integrating e-Government Applications and Semantic Web Services. In: AAAI Spring Symposium: Semantic Web Meets eGovernment, pp. 21–32 (2006)
3. Luck, M., Ashri, R., D'Inverno, M.: Agent-Based Software Development. Artech House, Norwood (2004)
4. Marques, F., Dias, G.P., Zòquete, A.: A General Interoperability Architecture for e-Government based on Agents and Web Services. In: 6th Iberian Conf. of Inf. Sys. and Tech., pp. 338–343 (2011)

5. Marques, F., Dias, G.P., Zòquete, A.: Modelo de Segurança para a Composição Dinâmica de Workflows em Arquiteturas de e-Government. Iberian J. of Inf. Sys. and Tech. 9, 15–26 (2012)
6. Mellouli, S., Bouslama, F.: Issues in Mult-agent Systems for e-Government Applications. In: Current Trends in Information Technology (CTIT), pp. 53–56 (2011)
7. De Meo, P., Quattrone, G., Terracina, G., Ursino, D.: Utilization of intelligent agents for supporting citizens in their access to e-government services. Web Intelligence and Agent Systems: An International Journal 5, 273–310 (2007)
8. Muthaiyah, S., Kerschberg, L.: Achieving Interoperability in e-Government Services with two Modes of Semantic Bridging: SRS and SWRL. Journal of Theoretical and Applied Electronic Commerce Research 3 (2008)
9. Piotrowski, Z.: Perspectives for using Software Agents in e-Government Applications. Annales UMCS, Informatica 8, 203–212 (2008)
10. Rishi, O.P., Sharma, A., Bhatnagar, A., Gupta, A.: Service Oriented Architecture for Business Dynamics: an Agent-based Approach. Emerging Technologies in E-Government, 19–28 (2008)
11. Sabucedo, L.M.A., Rifon, L.E.A., Corradini, F., Polzonetti, A., Re, B.: Knowledge-based Platform for eGovernment Agents: A Web-based Solution using Semantic Technologies. Expert Systems with Applications 37, 3647–3656 (2010)
12. The European Communities: Linking up Europe: The Importance of Interoperability for eGovernment Services. Commission staff working paper (2003)
13. Verginadis, Y., Mentzas, G.: Agents and Workflow Engines for Inter-Organizational Workflows in e-government cases. Business Process Management Journal 14, 188–203 (2008)
14. World Bank, http://go.worldbank.org/JKO5DVDMQ0
15. World Wide Web Consortium (W3C): Web Services Description Language (WSDL), version 1.1, W3C Recommendation (2001), http://www.w3.org/TR/wsdl
16. Zhang, P., Wang, Y., Wang, X.: Research on the Integration in E-government Based on Multi-agent. In: Web Intelligence and Intelligent Agent Technology, vol. 3 (2006)