Jingshan Huang Ryszard Kowalczyk
Zakaria Maamar David Martin
Ingo Müller Suzette Stoutenburg
Katia P. Sycara (Eds.)

LNCS 4504

Service-Oriented Computing: Agents, Semantics, and Engineering

AAMAS 2007 International Workshop, SOCASE 2007
Honolulu, HI, USA, May 2007
Proceedings

Springer

Jingshan Huang Ryszard Kowalczyk
Zakaria Maamar David Martin
Ingo Müller Suzette Stoutenburg
Katia P. Sycara (Eds.)

Service-Oriented Computing: Agents, Semantics, and Engineering

AAMAS 2007 International Workshop, SOCASE 2007
Honolulu, HI, USA, May 14, 2007
Proceedings

Springer

Volume Editors

Jingshan Huang
University of South Carolina, Columbia, SC 29208, USA
E-mail: huang27@sc.edu

Ryszard Kowalczyk
Swinburne University of Technology, Hawthorn, VIC 3122, Australia
E-mail: rkowalczyk@ict.swin.edu.au

Zakaria Maamar
Zayed University, PO Box 19282, Dubai, United Arab Emirates
E-mail: zakaria.maamar@zu.ac.ae

David Martin
SRI International, Menlo Park, CA 94025-3493, USA
E-mail: martin@ai.sri.com

Ingo Müller
Swinburne University of Technology, Hawthorn, VIC 3122, Victoria, Australia
E-mail: imueller@ict.swin.edu.au

Suzette Stoutenburg
The MITRE Corporation, Colorado Springs, Colorado 80910, USA
E-mail: suzette@mitre.org

Katia P. Sycara
Carnegie Mellon University, Pittsburgh, PA. 15213, USA
E-mail: katia@cs.cmu.edu

Library of Congress Control Number: 2007927059

CR Subject Classification (1998): H.3.5, H.3.3, H.3-4, I.2, C.2.4

LNCS Sublibrary: SL 3 – Information Systems and Application,
incl. Internet/Web and HCI

ISSN 0302-9743
ISBN-10 3-540-72618-7 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-72618-0 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12066605 06/3180 5 4 3 2 1 0

Preface

The global trend towards more flexible and dynamic business process integration and automation has led to a convergence of interests between service-oriented computing, semantic technology, and intelligent multiagent systems. In particular the areas of service-oriented computing and semantic technology offer much interest to the multiagent system community, including similarities in system architectures and provision processes, powerful tools, and the focus on issues such as quality of service, security, and reliability. Similarly, techniques developed in the multiagent systems and semantic technology promise to have a strong impact on the fast-growing service-oriented computing technology.

Service-oriented computing has emerged as an established paradigm for distributed computing and e-business processing. It utilizes services as fundamental building blocks to enable the development of agile networks of collaborating business applications distributed within and across organizational boundaries. Services are self-contained, platform-independent software components that can be described, published, discovered, orchestrated, and deployed for the purpose of developing distributed applications across large heterogeneous networks such as the Internet.

Multiagent systems are also aimed at the development of distributed applications, however, from a different but complementary perspective. Service-oriented paradigms are mainly focused on syntactical and declarative definitions of software components, their interfaces, communication channels, and capabilities with the aim of creating interoperable and reliable infrastructures. In contrast, multiagent systems center on the development of reasoning and planning capabilities of autonomous problem solvers that apply behavioral concepts such as interaction, collaboration, or negotiation in order to create flexible and fault-tolerant distributed systems for dynamic and uncertain environments.

Semantic technology offers a semantic foundation for interactions among agents and services, forming the basis upon which machine-understandable service descriptions can be obtained, and as a result, autonomic coordination among agents is made possible. On the other hand, ontology-related technologies, ontology matching, learning, and automatic generation, etc., not only gain in potential power when used by agents, but also are meaningful only when adopted in real applications in areas such as service-oriented computing.

This volume consists of the proceedings of the Service-Oriented Computing: Agents, Semantics, and Engineering (SOCASE 2007) workshop held at the International Joint Conferences on Autonomous Agents and Multiagent Systems (AAMAS 2007). It also includes the four best papers selected from the Service-Oriented Computing and Agent-Based Engineering (SOCABE 2006) workshop held at AAMAS 2006. The papers in this volume cover a range of topics at the intersection of service-oriented computing, semantic technology, and intelligent multiagent systems, such as: service description and discovery; planning, composition and negotiation; semantic processes and service agents; and applications.

The workshop organizers would like to thank all members of the Program Committee for their excellent work, effort, and support in ensuring the high-quality program and successful outcome of the SOCASE 2007 workshop. We would also like to thank Springer for their cooperation and help in putting this volume together.

May 2007

Jingshan Huang
Ryszard Kowalczyk
Zakaria Maamar
David Martin
Ingo Müller
Suzette Stoutenburg
Katia Sycara

Organization

SOCASE 2007 was held in conjunction with The Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007) on May 14, 2007 at the Hawaii Convention Center in Honolulu, Hawaii.

Organizing Committee

Jingshan Huang, University of South Carolina, USA
Ryszard Kowalczyk, Swinburne University of Technology, Australia
Zakaria Maamar, Zayed University Dubai, United Arab Emirates
David Martin, SRI International, USA
Ingo Müller, Swinburne University of Technology, Australia
Suzette Stoutenburg, The MITRE Corporation, USA
Katia Sycara, Carnegie Mellon University, USA

Program Committee

Esma Aimeur, University of Montreal, Canada
Stanislaw Ambroszkiewicz, Polish Academy of Sciences, Poland
Yacine Atif, United Arab Emirates University, United Arab Emirates
Youcef Baghdadi, Sultan Qaboos University, Oman
Djamal Benslimane, Université Claude Bernard Lyon 1, France
Jamal Bentahar, Concordia University Montreal, Canada
M. Brian Blake, Georgetown University, USA
Peter Braun, the agent factory GmbH, Germany
Paul A. Buhler, College of Charleston, USA
Bernard Burg, Panasonic Research, USA
Jiangbo Dang, Siemens Corporate Research, USA
Ian Dickinson, HP Laboratories Bristol, UK
Chirine Ghedira, Université Claude Bernard Lyon 1, France
Karthik Gomadam, University of Georgia, USA
Slimane Hammoudi, ESEO, France
Jingshan Huang, University of South Carolina, USA
Patrick Hung, University of Ontario, Canada
Nafaâ Jabeur, University of Windsor, Canada
Jugal Kalita, University of Colorado at Colorado Springs, USA
Mikko Laukkanen, TeliaSonera, Finland
Sandy Liu, NRC Institute for Information Technology, USA
Peter Mork, The MITRE Corporation, USA
Nanjangud C. Narendra, IBM India Research Lab, India
Manuel Núñez García, Universidad Complutense de Madrid, Spain

Author Index

Albayrak, Sahin 49, 92

Botelho, Luís Miguel 1
Braun, Nicolas 49

Cáceres, César 132
Cissée, Richard 49

Dikenelli, Oguz 118

Ekinci, Erdem Eser 118
Endert, Holger 92

Fernández, Alberto 78, 132

Groza, Adrian 160
Gümüs, Özgür 118
Gürcan, Önder 118

Hirsch, Benjamin 92

Inaba, Masumi 107

Jennings, Nicholas R. 16

Kardas, Geylani 118
Kawamura, Takahiro 107
Kowalczyk, Ryszard 147
Küster, Tobias 92

Letia, Ioan Alfred 160
Li, Hong Xia 63
Lopes, António Luís 1

Ma, Yu-Fei 63
Marginean, Anca 160
Mizoguchi, Yumiko 107

Nagano, Shinichi 107
Nguyen, Xuan Thang 147

Ossowski, Sascha 78, 132

Payne, Terry R. 16

Stein, Sebastian 16
Sun, Pei 63

van Riemsdijk, M. Birna 31
Vasirani, Matteo 132

Wirsing, Martin 31

Table of Contents

Executing Semantic Web Services with a Context-Aware Service Execution Agent 1
António Luís Lopes and Luís Miguel Botelho

An Effective Strategy for the Flexible Provisioning of Service Workflows 16
Sebastian Stein, Terry R. Payne, and Nicholas R. Jennings

Using Goals for Flexible Service Orchestration 31
M. Birna van Riemsdijk and Martin Wirsing

An Agent-Based Approach to User-Initiated Semantic Service Interconnection 49
Nicolas Braun, Richard Cissée, and Sahin Albayrak

A Lightweight Agent Fabric for Service Autonomy 63
Yu-Fei Ma, Hong Xia Li, and Pei Sun

Semantic Service Composition in Service-Oriented Multiagent Systems: A Filtering Approach 78
Alberto Fernández and Sascha Ossowski

Towards a Mapping from BPMN to Agents 92
Holger Endert, Benjamin Hirsch, Tobias Küster, and Sahin Albayrak

Associated Topic Extraction for Consumer Generated Media Analysis 107
Shinichi Nagano, Masumi Inaba, Yumiko Mizoguchi, and Takahiro Kawamura

An MAS Infrastructure for Implementing SWSA Based Semantic Services 118
Önder Gürcan, Geylani Kardas, Özgür Gümüs, Erdem Eser Ekinci, and Oguz Dikenelli

A Role-Based Support Mechanism for Service Description and Discovery 132
Alberto Fernández, Matteo Vasirani, César Cáceres, and Sascha Ossowski

WS2JADE: Integrating Web Service with Jade Agents 147
Xuan Thang Nguyen and Ryszard Kowalczyk

Z-Based Agents for Service Oriented Computing 160
Ioan Alfred Letia, Anca Marginean, and Adrian Groza

Author Index 175

Executing Semantic Web Services with a Context-Aware Service Execution Agent

António Luís Lopes and Luís Miguel Botelho

We, the Body, and the Mind Research Lab of ADETTI-ISCTE,
Avenida das Forças Armadas, Edifício ISCTE, 1600-082 Lisboa, Portugal
{antonio.lopes,luis.botelho}@we-b-mind.org

Abstract. The need to add semantic information to web-accessible services has created a growing research activity in this area. Standard initiatives such as OWL-S and WSDL enable the automation of discovery, composition and execution of semantic web services, i.e. they create a Semantic Web, such that computer programs or agents can implement an open, reliable, large-scale dynamic network of Web Services. This paper presents the research on agent technology development for context-aware execution of semantic web services, more specifically, the development of the Service Execution Agent (SEA). SEA uses context information to adapt the semantic web services execution process to a specific situation, thus improving its effectiveness and providing a faster and better service to its clients. Preliminary results show that context-awareness (e.g., the introduction of context information) in a service execution environment can speed up the execution process, in spite of the overhead that it is introduced by the agents' communication and processing of context information.

Keywords: Context-awareness, Semantic Web, Service Execution, Agents.

1 Introduction

Semantic Web Services are the driving technology of today's Internet as they can provide valuable information on-the-fly to users everywhere. Information-providing services, such as cinemas, hotels and restaurants information and a variety of e-commerce and business-to-business applications are implemented by web-accessible programs through databases, software sensors and even intelligent agents.

Research on Semantic Web standards, such as OWL-S [16] [19] and WSDL [3], opens the way for the creation of automatic processes for dealing with the discovery, composition and execution of web-based services. We have focused our research on the development of agent technology that allows the context-aware execution of semantic web services. We have decided to adopt the agent paradigm, creating SEA to facilitate the integration of this work in open agent societies [12], enabling these not only to execute semantic web services but also to seamlessly act as service providers in a large network of interoperating agents and web services.

In [21] the same approach was used in the Web Services infrastructure because of its capability to perform a range of coordination activities and mediation between requesters and providers. However, the broker-agent approach for the discovery and mediation of semantic web services in a multi-agent environment described in [21] does not take into account the use of context information. Thus we have decided to introduce context-awareness into the service execution process as a way of improving the services provided in dynamic multi-agent environments, such as the ones operating on pure peer-to-peer networks. Furthermore, the use of context information helps improve the execution process by adding valuable situation-aware information that will contribute to its effectiveness.

Being able to engage in complex interactions and to perform difficult tasks, agents are often seen as a vehicle to provide value-added services in open large-scale environments. However, the integration of agents as semantic web services providers is not easy due to the complex nature of agents' interactions. In order to overcome this limitation, we have decided to extend the OWL-S Grounding specification to enable the representation of services provided by intelligent agents. This extension is called the *AgentGrounding* and it is further detailed in [15].

We have also introduced the use of *Prolog* [4] for the formal representation of logical expressions in OWL-S control constructs. As far as we know, the only support for the formal representation of logical expressions in OWL-S (necessary for conditions, pre-conditions and effects) is done through the use of SWRL [13] and PDDL. Performance tests show that our *Prolog* approach improves the execution time of logical expressions in OWL-S services.

The remaining of this paper is organized as follows: section 2 gives a brief overview of related work; section 3 describes the use of context information and the introduction of context-aware capabilities in SEA; section 4 describes a motivating example which depicts a scenario where SEA is used; section 5 fully describes SEA by presenting its internal architecture, external interface, execution process and the implementation details; section 6 presents the performance tests and the overall evaluation of our research; finally, in section 7 we conclude the paper.

2 Related Work

The need to add semantic information to web-accessible services has created a growing research activity over the last years in this area. Regarding semantic web services, two major initiatives rise above the other, mainly because of their wide acceptance by the research community: WSMO [23] and OWL-S. A comparison between the two service description languages [14] concludes that the use of WSMO is more suitable for specific domains related to e-commerce and e-business, rather than for generic use, i.e., for different and more complex domains. OWL-S was designed to be generic and to cover a wide range of different domains but it lacked the formal specification to deal with logic expressions in the service description. We decided to use OWL-S as the Service Description Language due to its power in representing several different and complex domains. Other semantic web service execution approaches, such as WSMX [8] [9] are available but all rely on WSMO. However, since OWL-S was the chosen service description language to be used in this research, it is important to analyze the existing developed technology related to this standard in particular.

Two main software tools are referred in the OWL-S community as the most promising ones, regarding OWL-S services interpretation and execution: OWL-S VM [20] and OWL-S API [24]. However, at the time this research work has started, the OWL-S VM did not have a public release. OWL-S API is a developed Java API that supports the majority of the OWL-S specifications. For being the only OWL-S tool publicly available at the time, we have chosen to use and extend the OWL-S API.

In the interest of making the created technology interoperable with other systems that were already developed, we decided to ground its design and implementation on FIPA specifications, which are widely accepted agent standards. There are several existing FIPA-compliant systems that can be used: AAP [11], JADE [2], ZEUS [18], AgentAcademy [17] are just a few to be mentioned. We decided to use the JADE multi-agent platform because of its large community of users that constantly contribute to the improvement of the technology.

3 Service Execution and Context-Awareness

Context-aware computing is a computing paradigm in which applications can discover and take advantage of contextual information. As described in [7] "context is any information that can be used to characterize the situation of an entity, being an entity a person, place or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves". We can enhance this definition of context by stating that the context of a certain entity is any information (provided by external sensors or other entities) that can be used to characterize the situation of that entity individually or when interacting with other entities. The same concept can be transferred to application-to-application interaction environments.

Context-aware computing can be summarized as being a mechanism that collects physical or emotional state information on a entity; analyses that information, either by treating it as an independent variable or by combining it with other information collected in the past or present; performs some action based on the analysis; and repeats from the first step, with some adaptation based on previous iterations [1].

SEA uses a similar approach as the one described in [1] to enhance its activity, by adapting it to the specific situation that the agent and its client are involved in, at the time of the execution process.

SEA interacts with a generic context system [5] in order to obtain context information, subscribe desired context events and to provide relevant context information. Other agents, web services and sensors (both software and hardware) in the environment will interact with the context system as well, by providing relevant context information related to their own activities, which may be useful to other entities in the environment.

Throughout the execution process, SEA provides and acquires context information from and to this context system. For example, SEA provides relevant context information about itself, such as its queue of service execution requests and the average time of service execution. This will allow other entities in the environment to determine the service execution agent with the smallest work load, and hence that can provide a faster execution service.

During the execution of a compound service, SEA invokes atomic services from specific service providers (both web services, and service provider agents). SEA also provides valuable information regarding these service providers' availability and average execution time to the context system. Other entities can use this information (by contacting the context system) to rate service providers or to simply determine the best service provider to use in a specific situation.

Furthermore, SEA uses its own context information (as well as information from other sources and entities in the environment) to adapt the execution process to a specific situation. For instance, when selecting among several providers of some service, SEA will choose the one with better availability (with less history of being offline) and lower average execution time.

In situations such as the one where service providers are unavailable, it is faster to obtain the context information from the context system (as long as service providers can also provide context information about their own availability) than by simply trying to use the services and finding out that they are unavailable (because of the time lost waiting for connection time-outs to occur). After obtaining this relevant information, SEA can then contact other service-oriented agents (such as service discovery and composition agents) for requesting the re-discovering of service providers and/or the re-planning of composed services. This situation-aware approach using context information on-the-fly helps SEA providing a value-added execution service.

4 Example: Search Books' Prices

In this section we present an example scenario in the domain of books' prices searching, in order to better prove the need for the existence of a broker agent for the execution of semantic web services.

Imagine a normal web user that wants to find a specific book (of which he doesn't recall the exact title) of a certain author, at the best price available. Probably, he would start by using a domain-specific search engine to find the intended item. After finding the exact book, he would then try to find websites that sell it. After doing an extensive search, he would finally find the web site that sells the book at the best price, but it only features the price in US dollars. The user is Portuguese and he would like to know the book's price in Euros, which leaves him with a new search for a currency converter service that would help him with this task. As we can see, this user would have to handle a lot of different web sites and specific searches to reach its objective: find the best price of a certain book.

The composition of atomic semantic web services into more complex value-added services would present an easy solution to this problem. The idea is to provide a unique compound service with the same features as the example described above, but the use of which would be a lot simpler, since it would be through the interaction with a service execution broker agent.

Fig. 1 shows the overall scenario description, by presenting all the participants and the interactions between them. We will assume the existence of a compound service called "book best-price search". The dashed gray lines represent the interactions that are not subject of this paper.

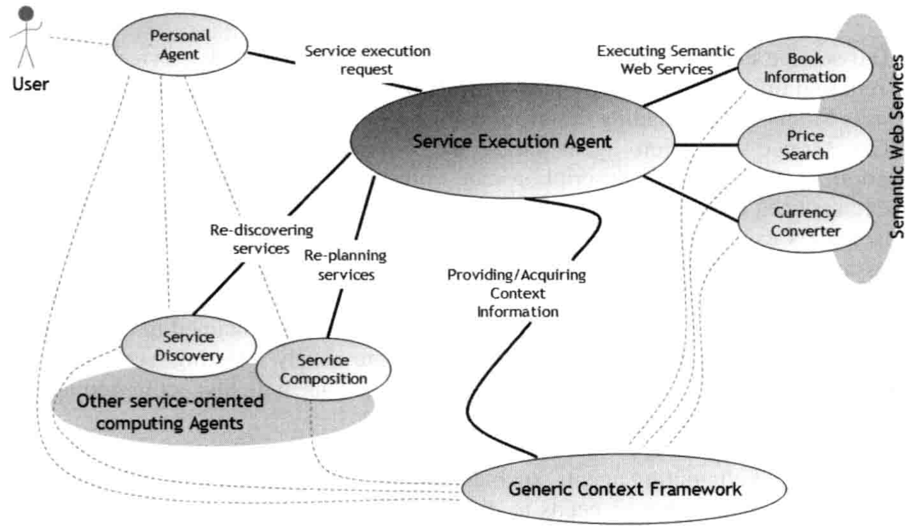

Fig. 1. Overall Scenario Description

The user of this service is represented in the figure as the client user and by his personal agent. The client user will provide the necessary information (author's name, book's title and expected currency) to his personal agent so that this can start the interaction with the remaining participants in order to obtain the desired result of the "book best-price search" service.

In order to request the execution of the "book best-price search" service, the personal agent needs to have the service's OWL-S description. This could have been previously obtained from a composition planner agent or service discovery agent. This interaction and the composition of the compound service are not covered by this paper.

After sending the "book best-price search" service's OWL-S description and the instantiation information (input parameters) provided by its user to the service execution agent, the personal agent will wait for the result of the service to then inform its user.

Semantic Web Services can be any web-based computer application, such as web services or intelligent agents, as long as they are described using a formal description language, such as OWL-S. They are represented in the figure on the opposite end to the client user. For this example, we'll consider the existence of the following semantic web services:

- *Book Information Agent* – this web-accessible information agent provides information about books and its authors
- *Price Search Web Services* – these web-accessible services provide an item's best-price search service (such as Amazon and Barnes & Noble)
- *Currency Converter Web Service* – this web service provides simple conversion of a monetary value from one currency to another.

The service execution agent interacts with these semantic web services to obtain the required information, according to instructions in the compound service's OWL-S Process Model and Grounding descriptions.

The service execution agent bases the execution on the service's OWL-S description. This OWL-S description can, sometimes, be incomplete, i.e., missing atomic services information regarding Grounding information. This can compromise the service's execution simply because the service execution agent doesn't have the necessary information to continue. On the other hand, if the service's OWL-S description is complete but the service execution agent is operating on very dynamic environments (such as pure P2P networks), the information contained in the service's OWL-S description can be easily out-dated (previously existing semantic web services are no longer available in the network). This will also compromise the agent's service execution activity.

To solve this problem, the service execution agent can interact with other service-oriented computing agents, such as service discovery and service composition agents, for example, when it needs to discover new services or when it needs to do some re-planning of compound services that somehow could not be executed, for whatever reasons explained above.

5 Service Execution Agent

The Service Execution Agent (SEA) is a broker agent that provides context-aware execution of semantic web services. The agent was designed and developed considering the interactions described in sections 3 and 4 and the internal architecture was clearly designed to enable the agent to receive requests from client agents, acquire/provide relevant context information, interacting with other service coordination agents when relevant and execute remote web services.

This section of the paper is divided into four sub-sections. Sub-sections 5.1 and 5.2 describe the internal architecture of the agent, explaining in detail the internal components and their interactions both internal and external, using the agent FIPA-ACL interface. Sub-section 5.3 describes the execution process that the agent carries out upon request from a client agent. Sub-section 5.4 provides some details on the implementation of the agent.

5.1 Internal Architecture

The developed agent is composed of three components: the Agent Interaction Component (AIC), the Engine Component (EC) and the Service Execution Component (SEC). Fig. 2 illustrates the internal architecture of the agent and the interactions that occur between the components and with external entities.

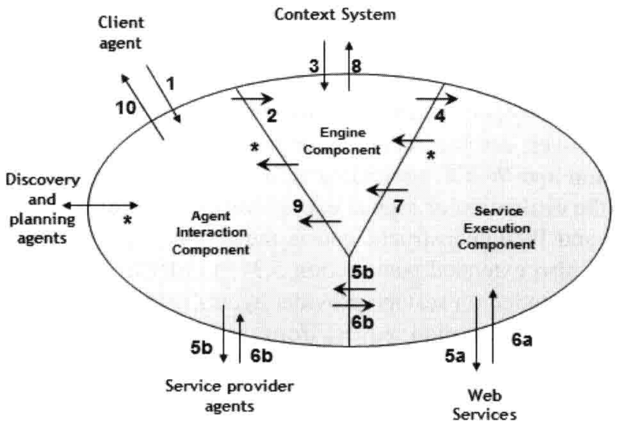

Fig. 2. SEA Internal Architecture and Interactions

The AIC was developed as an extension of the JADE platform and its goal is to provide an interaction framework to FIPA-compliant agents, such as SEA's clients (requesting the execution of specified services – Fig. 2, step 1) and service discovery and composition agents (when SEA is requesting the re-discovering and re-planning of specific services – Fig. 2, steps *). This component extends the JADE platform to provide extra features regarding language processing, behaviour execution, database information retrieval and components' communication. Among other things, the AIC is responsible for receiving messages, parsing them and processing them into a suitable format for the EC to use it (Fig. 2, step 2). The reverse process is also the responsibility of the AIC – receiving data from the EC and processing it into the agents' suitable format to be sent as messages Fig. 2, step 9).

The EC is the main component of SEA as it controls the agent's overall activity. It is responsible for pre-processing service execution requests, interacting with the context system and deciding when to interact with other agents (such as service discovery and composition agents). When the EC receives an OWL-S service execution request (Fig. 2, step 2), it acquires suitable context information (regarding potential service providers and other relevant information, such as client location – Fig. 2, step 3) and schedules the execution process. If the service providers of a certain atomic service (invoked in the received composed service) are not available, SEA interacts with a service discovery agent (through the AIC – Fig. 2, steps *) to discover available providers for the atomic services that are part of the OWL-S compound service. If the service discovery agent cannot find adequate service providers, the EC can interact with a service composition agent (again through the AIC – Fig. 2, steps *) asking it to create an OWL-S compound service that produces the same effects as the original service. After having a service ready for execution, with suitable context information, the EC sends it to the SEC (Fig. 2, step 4), for execution. Throughout the execution process, the EC is also responsible for providing context information to the context system, whether it is its own information (such as