An Abstract Architecture for Service Coordination in IP2P Environments[1]

Cesar Caceres, Alberto Fernandez, Sascha Ossowski, Matteo Vasirani

Artificial Intelligence Group, University Rey Juan Carlos, Madrid, Spain

Abstract. Intelligent agent-based peer-to-peer (IP2P) environments provide a means for pervasively providing and flexibly co-ordinating ubiquitous business application services to the mobile users and workers in the dynamically changing contexts of open, large-scale, and pervasive settings. In this paper, we present an abstract architecture for service delivery and coordination in IP2P environments that has been developed within the CASCOM project. Furthermore, we outline the potential benefits of a role-based interaction modelling approach for a concrete application of this abstract architecture based on a real-world scenario for emergency assistance in the healthcare domain.

1 Introduction

The ever-growing number of services on the WWW provides enormous business opportunities. In particular, there is a huge potential for creating added value through service coordination. For this to happen, technology must be developed capable of pervasively providing and flexibly co-ordinating ubiquitous business application services to mobile users and workers in the dynamically changing contexts of open, large-scale and pervasive application domains.

One step toward the realization of this vision is the development of an intelligent agent-based peer-to-peer (IP2P) environment. IP2P environments are extensions to conventional P2P architectures with components for mobile and ad hoc computing, wireless communications, and a broad range of pervasive devices. Basic IP2P facilities come as web services, while their reliable, task-oriented, resource-bounded, and adaptive coordination-on-the-fly characteristics call for agent-based software technology. A major challenge in IP2P environments is to guarantee a secure spread of (personal) service requests across multiple transmission infrastructures and ensure the trustworthiness of services that may involve a broad variety of providers.

In this paper, we present an abstract architecture for service delivery and coordination in IP2P environments that has been developed within the CASCOM project [4], [8] and discuss the benefits of role based interaction modelling for its

[1] This work has been supported in part by the European Commission under the project grant FP6-IST-511632 (CASCOM), and by the Spanish Ministry of Education and Science, project TIC2003-08763-C02-02. The authors would like to thanks all project partners for their contributions.

Caceres C., Fernandez A., Ossowski S. and Vasirani M. (2006).
An Abstract Architecture for Service Coordination in IP2P Environments.
In *Proceedings of the 3rd International Workshop on Computer Supported Activity Coordination*, pages 13-22
DOI: 10.5220/0002501100130022

instantiation to a particular application domain. This architecture aims at providing easy and seamless access to semantic Web services to *end users* and an innovative platform for various mobile business application services to *service providers*.

The paper is organised as follows. In Section 2 we discuss a real-world use case scenario to illustrate the kind of domains that the CASCOM abstract architecture aims at, and the types of services that it is to provide. Section 3 describes the structure of the abstract CASCOM architecture and elements in further detail. Section 4 presents a role-based interaction modelling approach for the CASCOM architecture and illustrates it in the aforementioned use case scenario. Finally, Section 5 outlines the learned lesson from this enterprise.

2 Application Scenario

The business application scenario that we will use throughout this paper is taken from the healthcare business domain, although our architecture does not contain any features that are specific to this field. The healthcare domain is to be one of the most viable and growing application field for intelligent mobile service coordination, not just for the technical requirements that it imposes, but also for its huge economic and social relevance. The following emergency assistance healthcare scenario, illustrates the kind of application that we aim at:

Valtteri, visiting Portugal, is seriously suffering from some disease unknown to him. His personal CASCOM agent installed on his PDA quickly finds out the contact information of a local emergency centre, so that Valtteri gives the noticeable symptoms of his sickness, his location and some general information. The local representative orders him to go to the nearest hospital immediately.

Immediately after Valtteri's call, his personal agent contacts the Emergency Medical Assistance (EMA) CASCOM agent at Finland and requests to send Valtteri's medical history to the Portuguese hospital. During the first examinations by the physician, there are some doubts about the diagnosis and he wants to obtain a second opinion, so he searches for a second opinion service using his CASCOM agent. After having found a specialized cardiologist, the agent provides Valtteri's symptoms and medical records. The cardiologist asks the physician to provide more precise information about a particular symptom (e.g. if the body area affected by the pain has been operated in the past). Finally the cardiologist provides her opinion, but it is not sufficiently clear to the physician in a certain part, so he asks for a further explanation.

The local physician, the patient, and EMA jointly take the decision that Valtteri should be transferred back to Finland as soon as possible. Valtteri's personal agent automatically finds out possible flight arrangements and informs all people that are involved during the travel (e.g., possible doctors and escorts). The agent also makes all the arrangements with a Finnish hospital. Back in Finland, Valtteri is treated at a hospital in Helsinki.

Similar scenarios, as well as a detailed discussion and the architecture requirements, can be found in CASCOM's deliverable D3.1 ([4]).

3 CASCOM Abstract Architecture

CASCOM abstract architecture is based on a layered approach that combines innovative network aspects, such as peer-to-peer and mobile networks, with modern software technologies, like semantic web services and intelligent agents (see Fig. 1). The four main components of this architecture act as a mediator between the applications and the underlying networks. They will be described in more detail below.

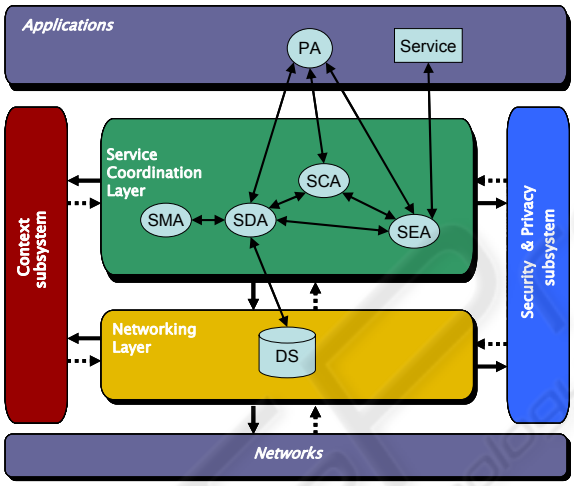

Fig. 1. CASCOM Abstract Architecture.

3.1 Networking Layer

The Networking Layer is located just over the networks, as a middleware to cope with the variability of the connections below, providing a generic, secure, and open Intelligent P2P network infrastructure with the following functionalities:

- Efficient, secure, and reliable agent message transport communication over wireless (and wireline) communication paths independently of the access technology. Bit efficient ACL encoding has been chosen due to the requirement for efficient communication over slow communication networks and resource limited devices. The simplicity and ease of implementation on small devices has been a key issue to choose FIPA-HTTP (HTTP 1.1 persistent connections) as the message transport protocol.
- Provide network-related context information to the context subsystem (see Section 3.3), which will then acquire, store, and update network/communication environment information (available networks, QoS of data communication…).
- Agent execution environment for resource-constrained mobile devices. Among different agent platforms, JADE/LEAP ([1]) is chosen as the most appropriate for

CASCOM: it follows the FIPA standard, allows agents to be efficiently executed on small devices, and is an active open source project.

- Low-level service discovery in IP2P environment. The networking layer should provide some support, mainly the "low-level" IP2P service look-up, to higher layers where the semantic service discovery takes place.

Requesters search dynamically for services published by different providers in a directory service (DS), which can be centralized or distributed. In the latter case, a federation of DSs could be built, where each DS registers itself in other DSs as a service. Thus, a DS can be found by querying entries in a DS. Such an approach was used in the Agentcities project [5], where the directories were federated accordingly to "application domains". Among the multiple combinations for interacting with DSs, the CASCOM abstract architecture favours a transparent access to federated DSs for reasons of transparency and simplicity, so that the requester interacts with a unique DS that is federated with other DSs.

Services are represented as structured objects within the directory using OWL-S [14] or WSMO [21]. However, directory entries are described in FIPA-SL0 language [7] because it is independent from the descriptions of the web services, is general enough, has a strong expressiveness, and keeps the architecture homogeneity. Translators will be developed to transform service descriptions into directory entries.

3.2 Service Coordination Layer

The Service Coordination Layer is located between the Networking Layer and the Application Layer, and uses the services offered by both the Context and the Security & Privacy Subsystems. This layer has two main functionalities: (1) Semantic service discovery (service discovery + semantic matchmaking), and (2) Service coordination (service composition + execution monitoring and replanning).

In the CASCOM abstract architecture, the semantic service discovery functionality is realised by two different types of agents: *Service Discovery Agents* (SDA) and *Service Matchmaking Agents* (SMA). This was done for reasons of efficiency and flexibility, as in some application domains the matchmaking functionality may not be necessary. In much the same way, the service coordination functionality is realised by *Service Composition Agent*s (SCA) and *Service Execution Agents* (SEA).

SDAs manage the discovery of required services, handling both abstract service descriptions and concrete service groundings. Usually, SDAs receive service specifications from the users' *Personal Agents* (PA) and acquire relevant contextual information from the context subsystem (see Section 3.3). With that information, they use the service discovery functionality of the networking layer and the semantic matching functionality of SMAs, to determine services that fulfil the received service discovery request. SDAs return then a set of descriptions/providers and their correspondent service process model and/or grounding.

SMAs provide the means to compare service specifications in a context dependent fashion. Again, they may focus on abstract service descriptions, on concrete service groundings, or both. Several semantic matchmaking approaches have been proposed [15], [19]. In CASCOM, we will use an OWL-S service matchmaker called OWLS-

MX [10]. The OWLS-MX matchmaker takes any OWL-S service as a query, and returns an ordered set of relevant services that match the query each of which annotated with its individual degree of matching, and syntactic similarity value. The user may extend the query by specifying the desired degree, and syntactic similarity threshold.

SCAs are capable of creating value-added composite services that match specific service specifications. Once SCAs receive service specifications, they contact SDAs to discover existing services in a given domain (high level descriptions or concrete service groundings), constrained to the current context (that can be acquired either by receiving information from other agents or by accessing the context subsystem), and plan a composite value-added service matching the received service specification. The SCAs make use of the OWLS-Xplan [11]. Xplan is a heuristic hybrid search planner based on the FF-planner [9]. The generated composite service will orchestrate one or more simpler services. The generated valued-added composite service may or may not contain service grounding information. In a typical interaction, when no single service is found matching a given service specification, the Service Composition functionality is used to create a value-added composite service matching the service specification, the output of which is a service description. These service descriptions may be stored or cached in some directory for later use (by agents looking for similar services' specifications).

SEAs manage the execution of composite value-added service descriptions generated by SCAs. Since the received compound service description relies on simpler services, the execution will also coordinate the execution of these simpler services. Whenever necessary, SEAs will use SDAs to discover appropriate available service providers for each of the simpler services invoked from the compound service description. The execution, namely the discovery of necessary service providers, will be dependent on the current context. The execution model is based on principles of the OSIRIS process management system [16].

Fig. 1 summarises the structure of the CASCOM abstract architecture and its relation to the agent types and their interactions described previously. Solid arrows indicate the main direction of control flow between components, while dotted lines refer to indicate the main direction of data flow.

3.3 Context Subsystem

The Context Subsystem [13] is accessed by both the Network Layer and the Service Coordination Layer, working as a gateway of context information between the layers. Its main functionality is to discover, acquire and store useful context information (e.g. the geographical position or the user preferences). This kind of information can be accessed by both layers either by explicitly querying the context subsystem ("pulling") or by subscribing a listener that is in charge of notifying changes and events occurring in the environment ("pushing"). The "pulling" solution permits to save resources, because the context is accessed only when needed, but on the other hand it requires more time to discover and acquire information, in the case that the needed information is not stored in a repository of historical data. The "pushing" solution offers less latency, because the context information is regularly sent to the subscriber so that it's always available, but on the other hand it uses more resources.

3.4 Security and Privacy Subsystem

The Security and Privacy Subsystem is also orthogonal to the first two layers. It is responsible for ensuring security and privacy of information throughout the different components of the CASCOM infrastructure.

Every node of the network should keep its data confidentiality, integrity, and availability (CIA) [17].But not only data is a security concern, also the software that deals with it, especially for network-centric systems, where the misuse, theft, and unauthorized usage of computing resources is well studied.

The security and privacy requirements identified are: identification, authentication, authorisation, single sign-on, and local and network security. We also need to guarantee the integrity of transmitted data, non-repudiability, traceability, privacy, delegation, and nationalization.

In order to guarantee the correct treatment of data, in the CASCOM project we rely on two novel architectural abstractions [2]. Such abstractions, namely *Validation-Oriented Ontologies and Guarantors*, are somehow known concepts in the realm of security and trust management, but we reformulate them here to make them first class architectural elements.

3.5 Instantiations of the CASCOM Abstract Architecture

The CASCOM abstract architecture can be instantiated in different ways, on the basis of the possible network infrastructures. We have considered four possible solutions: (1) a centralized, (2) a super-peer, (3) a structured pure and (4) an unstructured pure peer-to-peer solution.

In the first instantiation (centralized P2P) a single directory system is used by the peers as a lookup mechanism to find services. Peers search for service descriptions by querying directly and only to the directory.

In the second option (super-peer P2P) multiple directories exist in a federation and they construct the look-up mechanism for the peers to find adequate services. In addition to this super-peer structure, a pure P2P layer (typically a Distributed Hash Table) is used by the peers to find a bootstrap directory in the directory federation.

The third application (structured pure P2P) of the abstract architecture avoids bottlenecks and asymmetries by a regular distribution of the index information over the peers. Therefore the peers of a structured P2P-system have parts of the overall index stored locally and routing tables.

Finally, in the fourth application (unstructured pure P2P) of the CASCOM architecture there is no information about the distributed indexing and routing tables.

Due to the fact that the super-peer solution enhances robustness and decentralization, keeping the time a query is satisfied reasonably low, as the network is highly structured, this was the option chosen in the CASCOM project. The structured (hierarchical) composition of the architecture, consisting of a federation of directories, suites well to the nature of the emergency assistance scenario described in section 2. In case of emergency, fastest service discovery is required with as less as possible search and routing delay. Therefore, a service discovery request can be successfully answered by the bootstrap or adjacent directory. It is evident that in case of an emergency, only services in close proximity of a patient are helpful. This means

that directories should be organised in a way that each of them groups information of more or less confined spaces rather than functionality categories of service descriptions.

4 Role-based Interaction Modelling

In order to improve both the efficiency and the usability of the CASCOM architecture, we claim that it is necessary to account for the *types* of interaction that certain semantic services can be used in. In this section we outline how we will apply and extend our role-based and interaction-centric modelling framework [18] within the CASCOM project.

The abstract architecture conceives services to be delivered essentially by agents. In such an approach the agents usually act as mere wrappers for web services. The difference between a web service and a service provided by an agent boils down to a matter of interface: an agent can provide an implemented web service by a process of wrapping the service within an ACL interface in such a way that any agent can invoke its execution by sending the adequate (e.g. request) message.

However, agents are not only able to execute the service but also they can engage in different communicative interactions around that service. For example, an agent that provides the *second opinion* service, in the healthcare domain, should not only be able to return its diagnostic but also it might be required to explain it, give more details, recommend a treatment, etc., as shown in the scenario description of section 2. That means the provider must be able to engage in several different interactions during the provision of a service. Thus, if a physician requires one or more agents to provide a second opinion, it is actually looking for agents that include those communication capabilities around the "basic" *second opinion* service. In some sense, this is similar to the abstraction that an object makes by providing a set of methods to manipulate the data it encapsulates. In this case, the agent provides a set of interaction capabilities to provide the service.

To better explain our role-based approach, let's analyze the second opinion use case. In this scenario, the patient (or the physician of the hospital) can ask a health professional for a diagnosis on the basis of the symptoms and his medical records, like exams and past diseases. The "conversation" between the patient and the health professional can be modelled with a sequence of speech acts between the two agents involved, as depicted in Fig. 2. The patient *asks* the health professional for an opinion, providing the symptoms and the medical records. If there is no sufficient information, the health professional *requests* (possibly more times) additional information and finally gives his *advisement*. If the provided diagnosis is doubtful or not clear, the physician can also solicit an *explanation*.

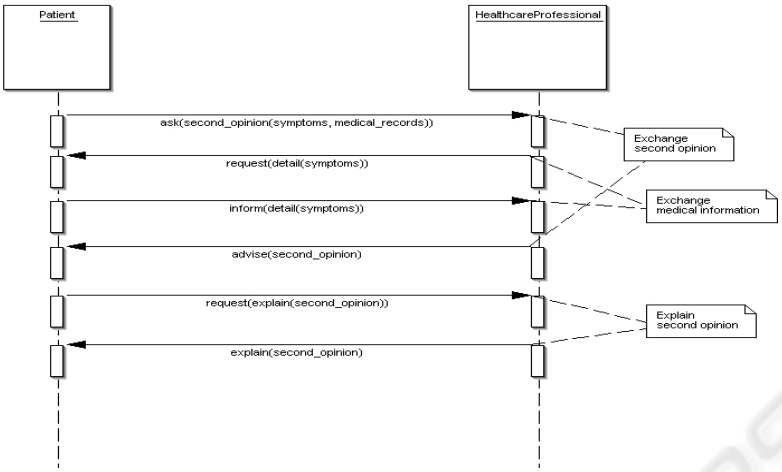

Fig. 2. Second Opinion conversation.

Starting from this conversation we can isolate three different interactions: (1) the second opinion exchange, which can comprise (2) a detailed information exchange about the health status. When the second opinion exchange finishes, an explanation (3) could occur. Having individuated the basic interactions, now we can model the second opinion use case in a more precise way. like a set of interactions and furthermore define the roles involved in. The product of this analysis should be a basic ontology of roles (Fig. 3), which is possible to refine and make more general. For example, the role *SecondOpinionRequestee* could be generalized in an *MedicalAdvisor* role, and in turn in a *Advisor* role, as well the *SecondOpinion* interaction could be generalized in a *MedicalAdvisement* interaction, and then in an *Advisement* one, in which the Advisor informs the Advisee about his beliefs with the aim of persuading the Advisee of the goodness of these beliefs.

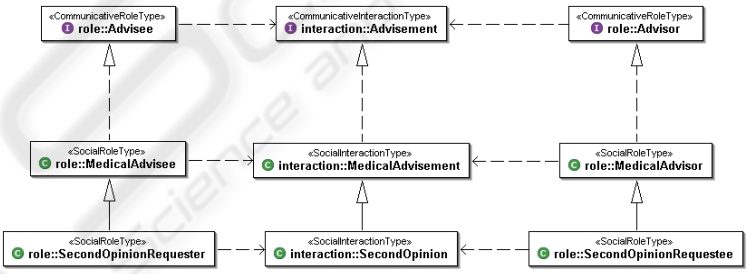

Fig. 3. Roles and interactions taxonomy.

Role and organizational concepts are abstraction mechanisms that have been used for many years in object oriented methodologies and are usually present in most agent-oriented methodologies like GAIA [20], MASE [6], etc. However, to the best of our knowledge there are no approaches that use them to describe what the services do.

Most current OWL-S matchmakers [15], [12] only consider service inputs and outputs in their matching algorithm. We propose considering roles and interactions as

well. This information may help the matchmaker in discriminating the service provider agents that better match a query.

We are exploring means of taking into account the roles that agents can play and the interactions in which they can engage in the publication of services in the service directory in the CASCOM project. Roles and interactions will be defined in an ontology that is being constructed. The efficiency of the matchmaking process can be improved by previously filtering those services that are compatible in the terms of roles and interactions they can participate in.

5 Conclusions

In this paper, we have described the CASCOM abstract architecture that smoothly integrates intelligent agent technology, semantic Web services, peer-to-peer, and mobile computing, for intelligent peer-to-peer mobile service environments. The different components of this architecture, as well as their interactions, have been described in detail. Four possible instantiations of the architecture (centralized, super-peer, structured pure and unstructured pure IP2P), depending on domain requirements, were explained and compared. The potential benefits of a role-based interaction modelling approach have been illustrated based on a real-world use case scenario for emergency assistance in the healthcare domain.

Besides the emergency assistance scenario discussed in this paper, the adequacy of the above CASCOM architecture has been analysed for two other scenarios in the patient telemonitoring & e-inclusion, as well as in the shopping mall assistance domains. The main conclusion drawn is that a hybrid super-peer option is the most general and applicable to all three domains. Still, the decision of how to organise services among directories (super-peers) undoubtedly depends on the particular requirements of a specific application. For example, spatial locations of services are more important in the emergency assistance scenario than logical (categorised) distribution (preferred for the other use case scenarios). Even a hybrid approach combining, for instance, pure P2P and super-peer approaches may sometimes be desirable.

We are currently implementing the presented architecture. In particular, the interest of the authors will focus on the instrumentation of role-based mechanisms for service discovery and coordination. At the end of the project, we will come up with a fully fledge demonstrator for a concrete real-world business application scenario.

References

1. Bergenti, F., and Poggi, A.: LEAP: A FIPA Platform for Handheld and Mobile Devices, Intelligent Agents VIII, Springer, 2002, pages 436–446.
2. Bianchi, R., Fontana, A., and Bergenti, F.: A Real-World Approach to Secure and Trusted Negotiation in MASs. Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems (AAMAS), 2005.
3. Cáceres, C., Fernández, A., Ossowski, S.: CASCOM – Context-aware Health-Care Service Coordination in Mobile Computing Environments. *ERCIM News 60*, pp. 77-78

4. CASCOM, http://www.ist-cascom.org
5. Constantinescu, I., Willmott, S. and Dale, J.: Deliverable 2.3: Agentcities Network Architecture, 2003.
6. DeLoach, S.A., Wood, M. F., and Sparkman, C. H.: Multiagent systems engineering. International Journal of Software Engineering and Knowledge Engineering, 2001, 11(3):231-258
7. FIPA08, Foundation for Intelligent Physical Agents: FIPA SL Content Language Specification. Geneva, Switzerland. Specification number SC00008I, 2002
8. Heikki, H., Klusch, M., Lopes, A., Fernandez, A., Schumacher, M., Schuldt, H., Bergenti, F., and Kinnunen, A.: Context-aware Business Application Service Co-ordination in Mobile Computing Environments, in AAMAS05 workshop on Ambient Intelligence - Agents for Ubiquitous Computing, 2005.
9. Hoffmann, J., and Nebel, B.: The FF Planning System: Fast Plan Generation Through Heuristic Search. Journal of Artificial Intelligence Research (JAIR), 2001, (14):253–302.
10. Klusch, M., Fries, B., Khalid, M., and Sycara, K.: OWLS-MX: Hybrid Semantic Web Service Retrieval. Proceedings 1st International AAAI Fall Symposium on Agents and the Semantic Web, Arlington VA, USA, 2005.
11. Klusch, M., Gerber, A., and Schmidt, M.: Semantic Web Service Composition Planning with OWLS-Xplan. Proceedings 1st International AAAI Fall Symposium on Agents and the Semantic Web, Arlington VA, USA, 2005.
12. Li, L., and Horrock, I.: A software framework for matchmaking based on semantic web technology. In Proc. 12th Int World Wide Web Conference Workshop on E-Services and the Semantic Web (ESSW), 2003.
13. Lopes, A., and Botelho, L.: SEA: a Semantic Web Services Contextaware Execution Agent. Proceedings 1st International AAAI Fall Symposium on Agents and the Semantic Web, Arlington VA, USA, 2005.
14. OWL-S Home Page. http://www.daml.org/services/owl-s/
15. Paolucci, M., Kawamura, T., Payne, T., and Sycara, K.: Semantic matching of web services capabilities. In Proceedings of the First International Semantic Web Conference on The Semantic Web, 2002, pages 333-347. Springer-Verlag
16. Schuler, C., Weber, R., Schuldt, H., and Schek, H.-J.: Scalable Peer-to-Peer Process Management – The OSIRIS Approach. In Proceedings of the 2nd International Conference on Web Services (ICWS), 2004, pages 26–34, San Diego, CA, USA, IEEE Computer Society
17. SecPedia. http://www.infosecpedia.org/pedia/index.php/Main_Page
18. Serrano, J.M.; Ossowski, S.; Fernández, A.: The Pragmatics of Software Agents – Analysis and Design of Agent Communication Languages, Intelligent Information Agents – The European AgentLink (Klusch et al. ed.), Springer, 2002, pp. 234274
19. Sycara, K., Klusch, M., Widoff, S., and Lu, J.: Larks: Dynamic matchmaking among heterogeneous software agents in cyberspace. Journal of Autonomous Agents and Multi-Agent Systems, 5(2). Kluwer Academic Press, 2002.
20. Wooldridge, M., Jennings, N. R., and Kinny, D.: The Gaia Methodology for Agent-Oriented Analysis and Design. Journal of Autonomous Agents and Multi-Agent Systems, 2000, 3(3) 285-312.
21. WSMO working group http://www.wsmo.org/.