

***a*-Buildings: Creating structured Agent Platforms to host agent-mediated services that share resources**

Javier Vázquez-Salceda¹, Carlos Mérida-Campos¹ and Josep M. Pujol¹

¹Departament de Llenguatges i Sistemes Informàtics.

Universitat Politècnica de Catalunya.

c/ Jordi Girona 1-3. E08034 Barcelona, Spain.

{jvazquez, dmerida, jmpujol}@lsi.upc.es

Phone: 34 934017989 Fax: 34 934017014

Abstract. After a period where the Internet was exclusively filled with content, the present efforts are moving towards services, which handle the raw information to create value from it. Therefore labors to create a wide collection of agent-based services are being performed in several projects, such as Agentcities does. In this work we present an architecture for agent platforms named *a*-Buildings. The aim of the proposed architecture is to ease the creation, installation, search and management of agent-mediated services and the share of resources among services. To do so the *a*-Buildings architecture creates a new level of abstraction on top of the standard FIPA [10] agent platform specification. Basically, an *a*-Building is a service-oriented platform which offers a set of *low level* services to the agents it hosts. We define *low level* services as those required services that are necessary to create more complex (*high level*) composed services.

1 Introduction and motivation

The potential of agent based applications in commercial and research areas is being increasingly exploited. In the last years Agent technologies have raised to make up a new paradigm for the research community, extending to many different areas (economics, sociology, artificial intelligence, computer science, etc.). It is a fact that the more the research and development of this 'entities' evolves, the more examples appear supporting the use of agents to emulate human behaviours. As the Multi-Agent Systems grow in complexity, more typically-human ways of organisation and interaction are adopted. These human ways have been extensively studied and fine-tuned along years in the real world. *Agent-mediated Institutions*[14, 5], *Electronic Market Places*[11, 4], *Virtual Communities*[12] are some of the examples of this 'agentification' process where humans are represented by agents keeping their working interactions and their procedures in a virtual manner.

A key issue for the agent technologies expansion is being the service-oriented use of them. This use [18, 9] offers the possibility of having more compact and best organized systems. In fact this is one of the objectives of the *Agentcities Project* [1]: "To facilitate a world-wide, open, heterogeneous and interoperable environment in which semantic, autonomous services can be defined, deployed and utilised in dynamic, composable and value-added ways". Currently this world-wide project is at its first stages,

focusing their efforts to implement interoperating agent platforms and the services provided by such platforms.

With this idea in mind, the Agentcities project aims to use agent technologies not only to create intelligent service search systems but also to create adaptive services, that are able to negotiate, compete and/or cooperate with other services in order to increase their success, and even creating added-value services composed by several interacting services (eg. a airline booking service plus a hotel booking service). An *Agentcity* is a virtual city that is composed of agent-mediated services, and users that interact with them *via* personal assistant agents. From the technological point of view, an *Agentcity* is a FIPA-compliant [10] agent platform (or a set of them) where user agents and service agents can interact. The platform gives support to such interaction at a low-level (message routing, agent directories, platform-to-platform connection).

Right now there are two ways to create a service and include it in the service network: a) To create a FIPA-compliant agent platform and link it to the Agentcities network. It requires to have machines (usually dedicated ones) with broad bandwidth connections and an amount of software and hardware to be installed and configured (databases, firewalls, etc.). b) To seek an existent platform where the service can be hosted. In the actual stage of the project, the research groups involved in Agentcities are following the first approach. But we think that there is work to be done to ease the hosting of services in the future, when regular companies will want to create their services and include them in the service network.

In order to ease the creation, installation, administration and maintenance of services in hosting agent platforms we propose to structure the agents and the resources in a platform in a way that standardizes the interaction among services and the resources they need, easing the installation process. To do so we propose to create a new level of abstraction, where platforms are organized in services. Using this abstraction, a service is seen as a *block* of agents that are related in order to provide a service. Each service specifies the resources it needs from the platform, and the platform offers such resources in the form of services. This agentification of the resources (such as Databases, heavy-weight reasoning engines) allows to do an adaptive management of such resources¹ and also to share such resources, avoiding to duplicate them for each service.

With all this ideas in mind we propose a particular platform architecture called *a-Buildings*. An *a-Building* is a special agent platform with an internal structure that eases the creation of agent-mediated services. It defines a service-centered framework where:

- services can be easily connected and/or composed,
- services that are commonly needed for certain applications are already defined².

The concept of *a-Buildings* stems from a widely accepted human way of organization that is the counterpart in the real world, the office-buildings. These office-buildings

¹ The grounds of this idea of using agents to monitor and manage resources come from the multiple research in agent-mediated resource allocation, such as bandwidth control [6].

² We think in having specific *a-Buildings* adapted for special areas. An example would be an *a-Building* to host e-commerce services, which will include as part of the pre-defined services the one or ones to ensure secure payments.

offer several integrated services that are necessary for any company, independently from its business area, avoiding the companies the need of performing those tasks. Services such as cleaning, heating, security, maintenance, etc. are independent from the company and can be managed by the office-buildings instead of by the hosted companies. Following the analogy, the hosted services in an *a*-Building could delegate some tasks to built-in services in order to reduce expenses and win efficiency (for instance, the *a*-Buildings might offer a powerful data storing system with a sound backup and restoring service to its guests which could be prohibitive or hard to manage by most of them). Thus, a service that *hires* a place in an *a*-Building has the chance of taking profit of the pool of shared *low level* services whether to improve the performance or to reduce costs. Once a set of *low level* services (which are likely to be used by everybody) are offered, the service could focalize exclusively on its own goals, easing the development of the service.

2 System Architecture

The architecture (depicted in figure 1) is composed by 5 layers:

- **a-building management level:** the outermost level, enclosing all the components of the architecture.
- **services level:** the level where all the systems that have contracted a hosting in the platform are placed.
- **wrappers level:** in this level there are several facilitator agents acting mainly as wrappers among agents from higher levels to the resources available underneath. There can be also agents that create composed services.
- **resource interface level:** this level does the adaptation of the resources to be connected and shared among agents in the previous level.
- **resource level:** the level where all the resources are placed.

All agent interactions are performed in levels A to C (the agent platform organizes the agents in several containers in order to ensure a separation of those levels). Levels D and E provide the connection among the agent platform and the resources (including the legacy software) that are available.

2.1 Level A: The *a*-Building management level

The main task of this level consists on the management of the services hosted in the platform and the access of agents to such services. The *a*-Building platform has a set of special agents to monitor not only the agents of the hosted services or the external agents, but also to be introspective about its own state in order to optimize the *a*-Building performance by means of tasks such as a) monitor the state of the services, b) monitor the use of resources in order to ensure a proper load-balancing³. All the agents that

³ The load-balancing is reached by a floating value of resources when services negotiate with the *a*-Building their use. Scarce, highly demanded services will be more expensive than those poorly demanded.

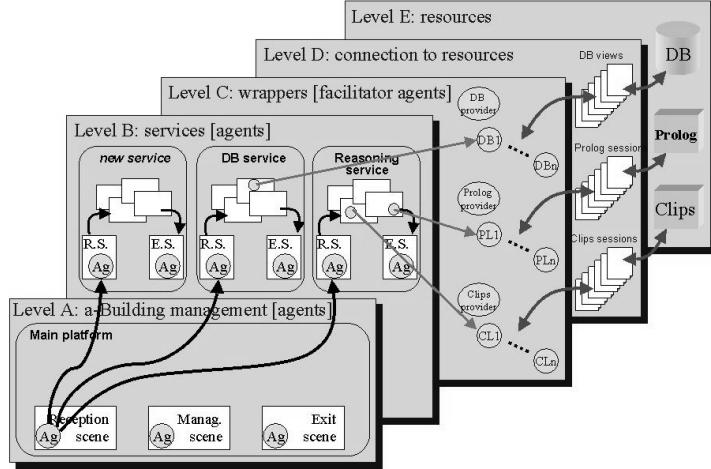


Fig. 1. The 5-layer architecture.

perform such tasks are grouped in an area called the *Management Scene*⁴, as depicted in figure 1.

Other of the tasks of this layer is access management (dealing with new agents willing to enter into the platform). To do so, there are defined two scenes: the *Reception Scene* and the *Exit Scene*, which are the entry and exit points of the platform. The *Reception Scene* is a key piece of the *a*-Building. In this scene there is a *Receptionist Agent* which receives the external agents and gives them access to the hosted services, extending the functionalities of the platform's DF by means of a service matchmaking process [13]. A complete description of the Reception Scene is provided in section 3.2.

Security issues Although there is still no coherent, completed picture for agent security within FIPA specifications, we recommend its trust and security model as an underlying specification for the security issues. Security is domain and platform (implementation) specific, and therefore, there is no general agent security architecture which is suitable for all the applications and implementations [16]. However, using trust and security FIPA models, we will be closer to the familiar triple of confidentiality, integrity and non repudiation within the platform [15].

FIPA specifies the use of a facilitator agent called *Directory Facilitator* (DF). The FIPA DF agent reveals service provider details to anyone that asks, moreover any agent registered in the platform with the *Agent Management System* (AMS, another FIPA facilitator agent) has the authority to register their services with the DF. This is a security

⁴ The term *scene* comes from a dialogical view of agent interaction (see §3).

aspect that an a-building must control in order to force the use of services through the control of the *Receptionist Agent*, and also to prevent the registration of services that have not been formally registered through a hosting hiring process. By the time this paper is being written, FIPA doesn't specifies support for keeping parts of directories private nor to control the access of them. In any case, an *a*-Building specifies a restricted use of DF in order to be consulted and to be modified just by the *Receptionist Agent*. In our architecture, the AMS must be directly accessible by the *Receptionist Agent* in order to ask it to perform a terminate instruction of a certain external agent considered malicious.

Communication between external agents and services hosted in a-buildings will be secured using symmetric and asymmetric cryptographic techniques. We propose the combination of asymmetric confidentiality and authentication techniques to send to the external agent a session key. After that the communication will be secured using symmetric cryptographic techniques (which are faster in the coding and decoding operations), making use of the session key provided.

In addition of requiring a secure communication, some services hosted in an a-building may also require an Electronic Certificate signed by a publicly recognized *Certification Authority* to attest the identity of an entity (*Authentication*). The *a*-Building specification also counts with this possibility in the control protocols defined in §3.

2.2 Level B: The services level

This level hosts all the services of the platform, either the *low level* services or the hosted services. Each service to be hosted specifies the resources it needs from the platform, and the platform offers such resources in the form of agent-mediated services. This agentification of the resources (such as data storage, heavyweight reasoning engines) allows to do an adaptive management of such resources, to compose resources as a composition of services⁵ and also to share those resources, avoiding to duplicate them for each service.

Services should be described by means of some language of service description, such as DAML [7], which is used in the Agentcities initiative.

2.3 Levels C to E: agentification of resources

In Level C there are several facilitator agents acting mainly as wrappers among agents from higher levels to the resources available underneath. There can be also agents that create composed services.

Level D does the adaptation of the resources to be connected and shared among agents in the previous level. This level is more logical than physical: for instance, depending on the resource, this adaptation can be done directly with tools provided by the resource in level E (like sessions, views, etc), or should be simulated/created by some kind of adapting software code (that can be done by means of the facilitator agents in level C).

⁵ We are using here the idea of service composition that is the grounds of the Agentcities Project[1], as presented in §1.

In Level E we sitate all the resources, including the legacy software the services will use (such as DB engines, language interpreters or reasoning engines) and a connection to other tools provided by the operating system (such as access to hard drives or any other operating system commands).

3 System protocols

To model interactions among external agents (agents entering into the platform), service agents (agents belonging to one of the services hosted in the platform) and platform agents (agents that manage the platform and interact with the other agents) we have chosen a *dialogical view*, where all interactions are seen as *messages* among agents. These interactions (called *illocutions* [14]) are structured through agent group meetings called *scenes* that follow well-defined protocols. A key element of the ISLANDER formalism⁶ used in our work, are agent *roles*. Each agent can be associated to one or more roles, and these roles define the scenes the agent can enter and the protocols it should follow (the *scene protocols* are defined as multi-role conversational patterns).

In the case of an a-building, there are three kinds of roles: the *external roles* (roles for incoming agents looking for services), the *platform roles* (roles for agents that carry out the management of the a-building) and *service roles* (roles for agents that belong to one of the sertvices hosted in the platform).

3.1 Hosting new services

As explained in section 1, external agents will hire the hosting of the service they represent, in an *a-Building*. Each *a-Building* will have its own policy to alocate its resources among the demand that may exist. An *a-building* can either fix the prices of its services and sell them or create a negotiation market of services depending on service load and demand, following some of the approaches that are being developed to put several issues of negotiation up for automated auction [17, 18, 2]. The kind of issues that *a-building* will offer are neither desks nor electric power, but computational power, hard drive space, and also centralised services as database services, backup services, etc. *a-building* will also offer a 'reception' service (explained in §3.2).

3.2 Managing the external agents

External agents will enter into the platform through the *Reception Scene* and will ask for services oftered by the hosts of the *a-building*. The protocol proposed for this request can be seen in figure 3 (where *prs* stands for the *Platform Reception Scene Manager* role, performed by the *Receptionist Agent*, and *ea* stands for an *External Agent* role). It consists on the following steps:

An external agent (client) makes a request for a service (message 1) in the *a-building*

⁶ The ISLANDER formalism [8] views an agent-based electronic institution as a type of *dialogical system* where all the interactions inside the institution are a composition of multiple dialogic activities (message exchanges).

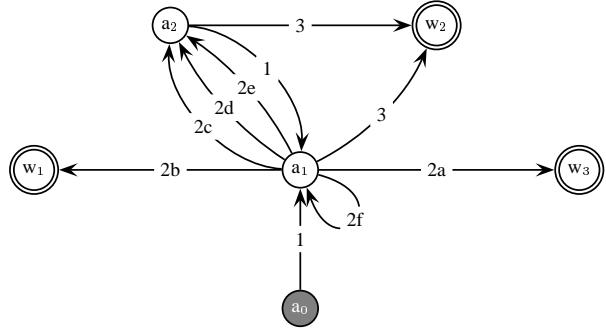


Fig. 2. The conversation graph for the Platform’s Reception Scene

Msg#	Illocution
1	(request (?x ea) (?y prs) (admission ?id_agent ?service_id {?certificate null}))
2a	(accept (!y prs) (!x ea) (accepted !id_agent !service_id session_cryptographic_key))
2b	(deny (!y prs) (!x ea) (admission_rejected ?deny_reason))
2c	(deny (!y prs) (!x ea) (service_not_found !service_id))
2d	(deny (!y prs) (!x ea) (service_not_available !service_id))
2e	(deny (!y prs) (!x ea) (certificate_needed !service_id))
2f	(inform (!y prs) (!x ea) (hold_on !service_id ?estimated_delay))
3	(inform (?x ea) (?y prs) (exit ?exit_reason))

Fig. 3. The illocutions for the Platform’s Reception Scene

The receptionist knows which services are hosted in the *a-building*, so it will look up for the requested service, and if it does not exist in, the client will be notified of this fact (message 2c) and brought to error state *a2*. If it exists, more validations will be needed in order to be allowed to access to the service. A request can be refused by several reasons:

- The service can be not available for an undetermined amount of time, therefore the client will be informed (message 2d) and brought to the error state *a2*.
- In case that for acceding the service an electronic certificate is needed, and the client had not provided it in its request, it will be informed (message 2e).
- if the external agent is in error state *a2*, it has the option to re-sending a new request fixing the notified problem fixed (such as the missing certificate) or to leave the *a-Building*.
- The service provider may be busy or temporaly unavailable, if so the external agent will be informed of this by the receptionist (message 2f) and the agent will have the option of holding on in the same state (*a1*) until the receptionist sends him a new message, or leaving the *a-building* (message 3).

- In the case a certificate was needed to use the service but the certificate the client issued is not accepted by the service provider the client will be informed by the receptionist (message 2b). Automatically the client will be brought to the exit state.

If the client is finally accepted, it will be endowed with a symmetric cryptographic key that will permit the agent to establish a ciphered communication with the service during this session.

Identification of incomers is a key issue, as it should provide: a) Identification of the incoming agent by a unique key. b) Authentification of the incoming agent c) the identification of a third entity that can take the commitments and obligations unattended by this agent. When the incoming agents are not mobile (so their execution is out of the control of the platform), the last capacity is really important: an agent can stop its execution without fulfilling the commitments (for instance, an incoming agent that crashes before paying the fees it committed to pay). One way to get all these capacities is to use Electronic Certificates issued by a *Certification Authority*, where the entity certifying (signing) the electronic certificate can be asked to fulfill commitments the agent holding that certificate hasn't fulfilled, such as giving some requested information, pay a bill, do some computation, etc.

3.3 Communication with hosted services

The services that have hired the hosting in the *a-building* delegate some responsibilities to the platform. The reception service will free the hosted service from some access control and filtering tasks but, in any case, some communication is needed between the *Receptionist Agent* and the service in order to accept or reject the incoming clients.

There are a set of predefined protocols that allow the interaction among the platform agents and the services hosted. Following these protocols is the only condition that is required from the services to be hosted, as they define an standard way of interacting.

One example of such protocols is the *Reception Scene* conversation graph depicted in figure 5 (where *prs* stands for the *Platform Reception Scene Manager* role and *srs* stands for the *Service Reception Scene Manager* role). The first thing that the receptionist will check is the availability of the requested service (message 4) for a certain client. The service provider can answer:

- The service is unavailable for an undetermined amount of time (message 5b)
- The service is currently busy but will be available in short (message 5c)
- The service is available and ready to receive new clients (message 5a)

If the service is available, and the service provider is not asking the clients for an electronic certificate, it may confirm the acceptance of the client to the receptionist straightforward (message 6a). If de lo contrario the service provider needs an electronic certificate from the client, it will enquire it to the receptionist (message 6b), and the receptionist will reply with a new message (message 7) enclosing in it the certificate from the client. Then the service provider will decide whether to accept the client or not (message 6a and 8 respectively). If the external agent was accepted, then the platform receptionist will give to the service the cryptographic key to be used between the client and the service (message 9).

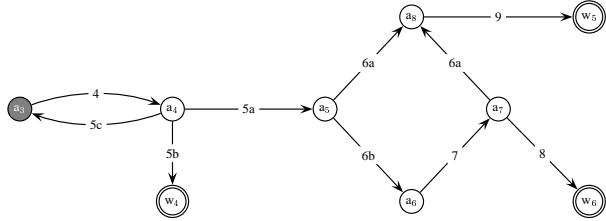


Fig. 4. The conversation graph among the *Platform Reception Scene Manager* and a *Service Reception Scene Manager*

Msg#	Illocution
4	(query-if (?x prs) (?y srs) (service_availability !id_agent))
5a	(inform (!y srs) (!x prs) (service_available))
5b	(inform (!y srs) (!x prs) (service_not_available ?reason))
5c	(inform (!y srs) (!x prs) (hold_on ?estimated_delay))
6a	(inform (?x srs) (?y prs) (accept_admission !id_agent))
6b	(query (?x srs) (?y prs) (user_certificate !id_agent))
7	(inform (!y prs) (!x srs) (user_certificate !id_agent ?certificate))
8	(inform (?x srs) (?y prs) (certificate_not_accepted))
9	(inform (!y prs) (?x srs) (session_key !id_agent ?session_cryptographic_key))

Fig. 5. The illocutions among the *Platform Reception Scene Manager* and a *Service Reception Scene Manager*

4 Conclusions and future work

In this work we present some of our ideas in a architecture for agent platforms, the *a*-Buildings, where all services are encapsulated and structured in a way that eases their developing and their hosting in an agent platform. We have used along the document the methafor of office buildings to illustrate the way that agents would interact within the *a*-Building. We believe that *a*-Building is a valid and intuitive model to exolte and share resources, for wide agent communities such as Agentcities [1], and also an intuitive and reliable way to manage the resources of platforms.

We are currently working in several aspects of the architecture, such as the Reception Scene, and the coupling to be made among the *Receptionist Agent*, the AMS and the DF. We are also working on mechanisms for resource pricing. A prototype of the *a*-Building is being developed using JADE agent platforms [3].

References

1. Agentcities Project. <http://www.agentcities.org/>.

2. M. Barbuceanu and Wai-Kau Lo. "a multi-attribute utility theoretic negotiation architecture for electronic commerce". In Frank Dignum and Ulises Cortés, editors, "Agent-Mediated Electronic Commerce III, Current Issues in Agent-Based Electronic Commerce Systems (includes revised papers from AMEC 2000 Workshop)", volume 2003 of *Lecture Notes in Computer Science*, pages 15–30. Springer, 2001.
3. F. Bellfamine, A. Poggi, and G. Rimassa. JADE : A FIPA compliant agent frame-work. In *Proc. PAAM'1999*, pages 97–108, 1999.
4. A. Chavez and P. Maes. Kasbah: An agent marketplace for buying and selling goods. In *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, 1996.
5. U. Cortés, A. López-Navidad, J. Vázquez-Salceda, A. Vázquez, D. Busquets, M. Nicolás, S. Lopes, F. Vázquez, and F. Caballero. Carrel: An agent mediated institution for the exchange of human tissues among hospitals for transplantation. In *3^{er} Congrés Català d'Intel.ligència Artificial*, pages 15–22. ACIA, 2000.
6. C. Courcoubetis, M. Dramitinos, and G.D. Stamoulis. An auction mechanism for bandwidth allocation over paths. In *17th International Teletraffic Congress (ITC)*, 2001.
7. The DARPA Agent Markup Language Homepage. <http://www.daml.org/>.
8. Marc Esteva, Julian Padget, and Carles Sierra. Formalizing a language for institutions and norms. In Milinde Tambe and Jean-Jules Meyer, editors, *Intelligent Agents VIII*, Lecture Notes in Artificial Intelligence. Springer Verlag, 2001. to appear.
9. P. Faratin, C. Sierra, and N. R. Jennings. Negotiation decision functions for autonomous agents. *Int. Journal of Robotics and Autonomous Systems*, 24(3-4):159–182, 1998.
10. The Foundation for Intelligent Phisical Agents, <http://www.fipa.org/repository/fipa2000.html>. *FIPA Specifications*, 2000.
11. The FishMarket Project. <http://www.iiia.csic.es/Projects/fishmarket>.
12. Henry Kautz, Bart Selman, and Mehul Shah. Referral Web: Combining social networks and collaborative filtering. *Communications of the ACM*, 40(3):63–65, 1997.
13. Matthias Klusch and Katia Sycara. Brokering and matchmaking for coordination of agent societies: A survey. In Andrea Omicini, Franco Zambonelli, Matthias Klusch, and Robert Tolksdorf, editors, *Coordination of Internet Agents: Models, Technologies, and Applications*, chapter 8, pages 197–224. Springer-Verlag, March 2001.
14. P. Noriega. *Agent-Mediated Auctions: The Fishmarket Metaphor*. Number 8 in IIIA Monograph Series. Institut d'Investigació en Intel.ligència Artificial (IIIA), 1997. PhD Thesis.
15. Agostino Poggi, Giovanni Rimassa, and Michele Tomaiuolo. Multi-user and security support for multi-agent systems. In Andrea Omicini and Mirko Viroli, editors, *WOA 2001 – Dagli oggetti agli agenti: tendenze evolutive dei sistemi software*, Modena, Italy, 4–5 September 2001. Pitagora Editrice Bologna.
16. S. Poslad and M. Calisti. "towards improved trust and security in fipa agent platforms". In Rino Falcone, Munindar P. Singh, and Yao-Hua Tan, editors, "Trust in Cyber-societies, Integrating the Human and Artificial Perspectives [based on a workshop on Deception, Fraud, and Trust in Agent Societies held during the Autonomous Agents Conference in Barcelona, Spain in June 2000]", volume 2246 of *Lecture Notes in Computer Science*. Springer, 2001.
17. C. Preist and C. Mérida-Campos. Automated negotiation in many-to-many markets for imperfectly substitutable goods. In *To appear in Proceedings of the AAMAS 2002 Agent-Mediated Electronic Commerce Workshop, Bologna, Italy*, 2002.
18. C. Sierra, P. Faratin, and N. Jennings. A service-oriented negotiation model between autonomous agents. In *Proc. 8th European Workshop on Modeling Autonomous Agents in a Multi-Agent World (MAAMAW-97)*, pages 17–35, Ronneby, Sweden, 1997.