Service Composition in Open Agent Societies

Agostino Poggi, Paola Turci, Michele Tomaiuolo

***Abstract* — Agentcities is a network of FIPA compliant agent platforms that constitute a distributed environment to demonstrate the potential of autonomous agents. One of the aims of the project is the development of a network architecture to allow the integration of platforms based on different technologies and models. It provides basic white pages and yellow pages services to allow the dynamic discovery of the hosted agents and the services they offer. An important outcome is the exploitation of the capability of agent-based applications to adapt to rapidly evolving environments. This is particularly appropriate to dynamic societies where agents act as buyers and sellers negotiating their goods and services, and composing simple services offered by different providers into new compound services.**

***Index Terms* — Autonomous agent, agent-mediated electronic commerce, agent security, ontology, web services, service composition, negotiation, goal delegation, service advertisement, service discovery.**

I. Introduction

One of the main reasons to use autonomous software agents is their ability to interact to show useful social behaviours rapidly adapting to changing environmental conditions. But the most interesting applications require that large and open societies of agents are in place, where collaborating and competing peers are able to interact effectively. In a context where a number of possible partners or competitors can appear and disappear, agents can highlight their ability to adapt to evolving social conditions, building and maintaining their networks of trust relations within a global environment.

The greatest effort to create such a large and open society of autonomous software agents is Agentcities [1]. The project, that started on the 1st of July as a research project founded by the European Commission, developed a network of FIPA-compliant agent platforms spanning over the whole globe. A number of complex agent-based applications are currently deployed on the network and, thanks to their compliance to the standards defined by FIPA (Foundation for Intelligent Physical Agents), they are able to communicate in an effective way to reciprocally exploit the service they provided to the community.

In fact, to allow the integration of different applications and technologies in open environments, high level communication technologies are needed. The project largely relies on semantic languages, ontologies and protocols in compliance with the FIPA standards [7].

One of the main achievement has been the demonstration of a working application that, relying on the marketplace infrastructure deployed on the Agentcities network, was able to dynamically discover and negotiate the services offered by different service providers. The collected services were then composed into a new compound service to be offered back on the marketplace, eventually allowing other agent based applications to dynamically discover it and, if useful to achieve their own goals, add it to their plans of action.

II. Network

The Agentcities network [13] grows around a backbone of 14 agent platforms, mostly hosted in Europe. These platforms are deployed as a 24/7 testbed, hosting the services and the prototype applications developed during the lifetime of the project. The backbone is an important resource for other organizations, even external to the project, that can connect their own agent-based services, making the network really open and continuously evolving.

Fig. 1. The Agentcities backbone

Currently, the Agentcities network counts 160 registered platforms, among which 80 have shown activities in the last few weeks. The platforms are based on more than a dozen of

Manuscript received July 10, 2003. This work is partially supported by the European Commission through the contracts IST-2000-28385 Agentcities.RTD.

A. Poggi is with the University of Parma, Information Engineering Department, Parco Area delle Scienze 181A, Parma, IT (phone: +39 0521 905728; fax: +39 0521 905723; e-mail: poggi@ce.unipr.it).

P. Turci is with the University of Parma, Information Engineering Department, Parco Area delle Scienze 181A, Parma, IT (phone: +39 0521 905728; fax: +39 0521 905723; e-mail: turci@ce.unipr.it).

M. Tomaiuolo is with the University of Parma, Information Engineering Department, Parco Area delle Scienze 181A, Parma IT (phone: +39 0521 905708; fax: +39 0521 905723; e-mail: tomamic@ce.unipr.it).

heterogeneous technologies, including Zeus, FIPA-OS, Comtec, AAP, Agentworks, Opal. More than 2/3 of them are based on JADE [9] and its derived technologies, as LEAP and BlueJADE.

A. Network Architecture

The structure of the Agentcities network architecture version 1.0 consists of a single domain ("the network") with a collection of FIPA 2000 Agent Platforms deployed in it. All these agent platforms are visible to one another and can exchange messages using HTTP or IIOP communication services. There is a collection of FIPA 2000 agents, each of which is running on one of the agent platforms in the network, and a collection of services, each of which is offered by one or more agents in the network. The central node, hosting the network services for the whole network, represents the critical point of the entire architecture. As a matter of fact the components and network services supporting the global network are configured in the form of a star topology with core directory and management facilities hosted at a single root node. This leads to problems with robustness (single point of failure) and scalability (potential overloads at the central point).

The network services, offered by the central node, are: Agent Platform Directory, Agent Directory and Service Directory. Each of these services is accessible over the Internet using both standard FIPA interfaces (access for agents) and web interfaces (access for humans). All services are publicly available and can be used by both project partners and third parties (non Agentcities.RTD partners) alike. Figure 2 shows the generic architecture for Agentcities network directory services (platforms, agents and services). For each type of directory a polling agent uses a data source of registered entities (platforms, agents or services) and regularly polls the instances found. The resulting information is accessible through agent and web interfaces.

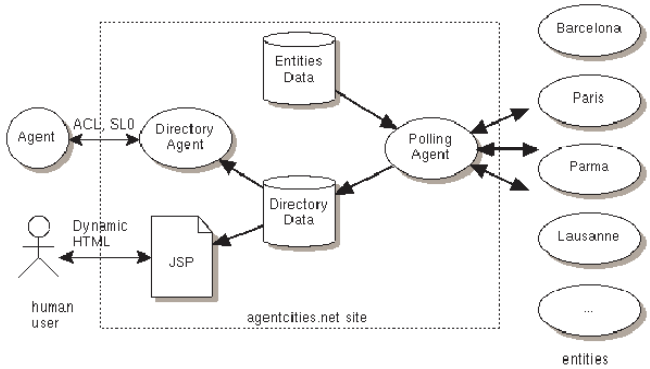

Fig. 2. Agentcities Network generic architecture

In the following each of these services are described in terms of its functions, interfaces and other possible useful criteria.

1) Agent Platform Directory

This service provides mechanisms for finding which platforms exist in the network and which are currently active. It is based on a database of registered platforms. The content of the directory is a single entry per platform that includes all the data registered by users for the platform. The directory contains an active component that regularly contacts registered platforms to check if they are running or not. Agents are able to access the list of platforms currently registered and the list of platforms currently running by sending queries to a dedicated Global Agent Platform Directory agent.

2) Agent Directory

This service provides mechanisms for finding which agents are currently registered and active in the network. More precisely it lists which agents are registered on known platforms, platforms which are currently registered in the database of the Agent Platform Directory service. The Agent Directory is therefore directly dependent upon the Agent Platform Directory and uses it as a data source. Even though the objective is to have information only about agents currently registered and active in the system, in practice the information is not always up to date. The freshness of this information depends on the time interval between two queries to the local AMS services on known platforms.

3) Service Directory

This is the more interesting service from the point of view of the service composition.

The Service Directory provides mechanisms for finding services that are currently being provided in the network; it works in the same way as the Agent Directory.

Frame Ontology	`service-description FIPA-Agent-Management`			
Parameter	**Description**	**Presence**	**Type**	**Reserved Values**
`name`	The name of the service.	Optional	`String`	
`type`	The type of the service.	Optional	`String`	`fipa-df fipa-ams`
`protocol`	A list of interaction protocols supported by the service.	Optional	Set of `String`	
`ontology`	A list of ontologies supported by the service.	Optional	Set of `String`	`FIPA-Agent-Management`
`language`	A list of content languages supported by the service.	Optional	Set of `String`	
`ownership`	The owner of the service	Optional	`String`	
`properties`	A list of properties that discriminate the service.	Optional	Set of `property`	

Tab 1. FIPA definition of a service description. Replicated from [FIPA00023]

The directory contains entries of the type defined in the FIPA Agent Management specification [7] for entries in the FIPA-DF yellow pages service. The service description definition is shown in Table 1.

4) Drawbacks

The deployed network services, based on the network architecture 1.0, proved to be useful tools and provided an important live update on the state of vital features of the network. Despite this, a number of issues needed to be addressed in order to improve the network functionality. A first issue concerns the centralization and the strictly related problems of robustness and scalability. A second issue deals with the service independence. In fact the Agent and Service Directory services depend upon the Platform Directory service, meaning that a failure in the later would cause a failure in the both of the former. A further issue regards the Agent and Service Directory service agents, which need only register on their local platform to appear in the global directories. This fact is not known to the AMS and DF services on the local platforms, since they have no way of knowing that their local information is being replicated elsewhere. Furthermore agents and services cannot be explicitly registered in global directories, they can only be registered locally. Finally registering a platform in the system requires a human to access a web site. These main issues and other drove the partners of the projects to design and implement a new version of the network architecture.

5) Network Architecture 2.0

While the FIPA 2000 standard will continue to be the backbone of the Agentcities network the new version of the architecture aims at generalise the network model to make it possible to employ or link to other technologies such as web services, GRID services or peer-to-peer networks. This is achieved by adopting an abstract model based on the notion of actors and services and mapping this to a number of different implementation technologies. Furthermore the changes carried out make possible to flexibly specify and describe detailed structural information about deployed networks of actors and services. This is achieved by adopting a model based on domain structures that can be used to describe and implement arbitrary organisational structures. Domains are instantiated as directories represented by collections of directory services that implement domain policies. A prototype of the network architecture version 2.0 was implemented and deployed. The actual adoption in the Agentcities project of a preliminary version of the network provided some early feedback on its usefulness and functionality. However, more work needs to be done to gather more comments within the Agentcities community.

III. Service Composition

The main rationale for using agents is their ability to adapt to rapidly evolving environments and yet being able to achieve their goals. In many cases, this can only be accomplished by collaborating with other agents and leveraging on the services provided by cooperating agents. This is particularly true when the desired goal is the creation of a new service to be provided to the community, as this scenario often calls for the composition of a number of simple services that are required to create the desired compound service.

In the following pages we will show how the different components hosted on the Agentcities network can be used to orchestrate the composition of simple services to build a new compound service. Figure 3 depicts the reference scenario of a demonstration held at the end of the project, when a number of individual applications deployed by the different partners were shown to work in close integration to achieve this goal.

A. Event Organizer

The Event Organizer is an agent-based prototype application showing the results that can be achieved using the services provided by the Agentcities project. It allows a conference chair to organize an event, booking all needed venues and arranging all needed services, and then sell the tickets for the new event.

Using the web interface of the Event Organizer, its user can list a set of needed services, fixing desired constraints on each individual service and among different services. The global goal is then split into sub-goals, assigned to skilled solver agents [2].

The Event Organizer uses the marketplace infrastructure deployed on the Agentcities network to search for suitable venues. These are matched against cross-service constraints and, if found, a proper solution is proposed to the user as a list of services that allow the arrangement of the event. As a matter of facts, the task of selecting services that validate all the constraints is distributed between the Trade House, which checks the constraints regarding individual services, and the Event Organizer, which instead checks the constraints that link the features of two different services.

The selected services are then negotiated on the marketplace with their providers and a list of contracts is proposed to the user for final acceptance. The negotiation process involves the servant agents, representing the buyer and the seller, hosted on the Trade House. If successful, the negotiation ends suggesting a point of balance between the contrasting interests of the buyer and the seller. It can take into account multiple attributes of the traded good, for example the price and the period of booking.

Finally, after the new event has been successfully organized and all contacts have been accepted and signed, the tickets for it can be sold, once again using the marketplace infrastructure.

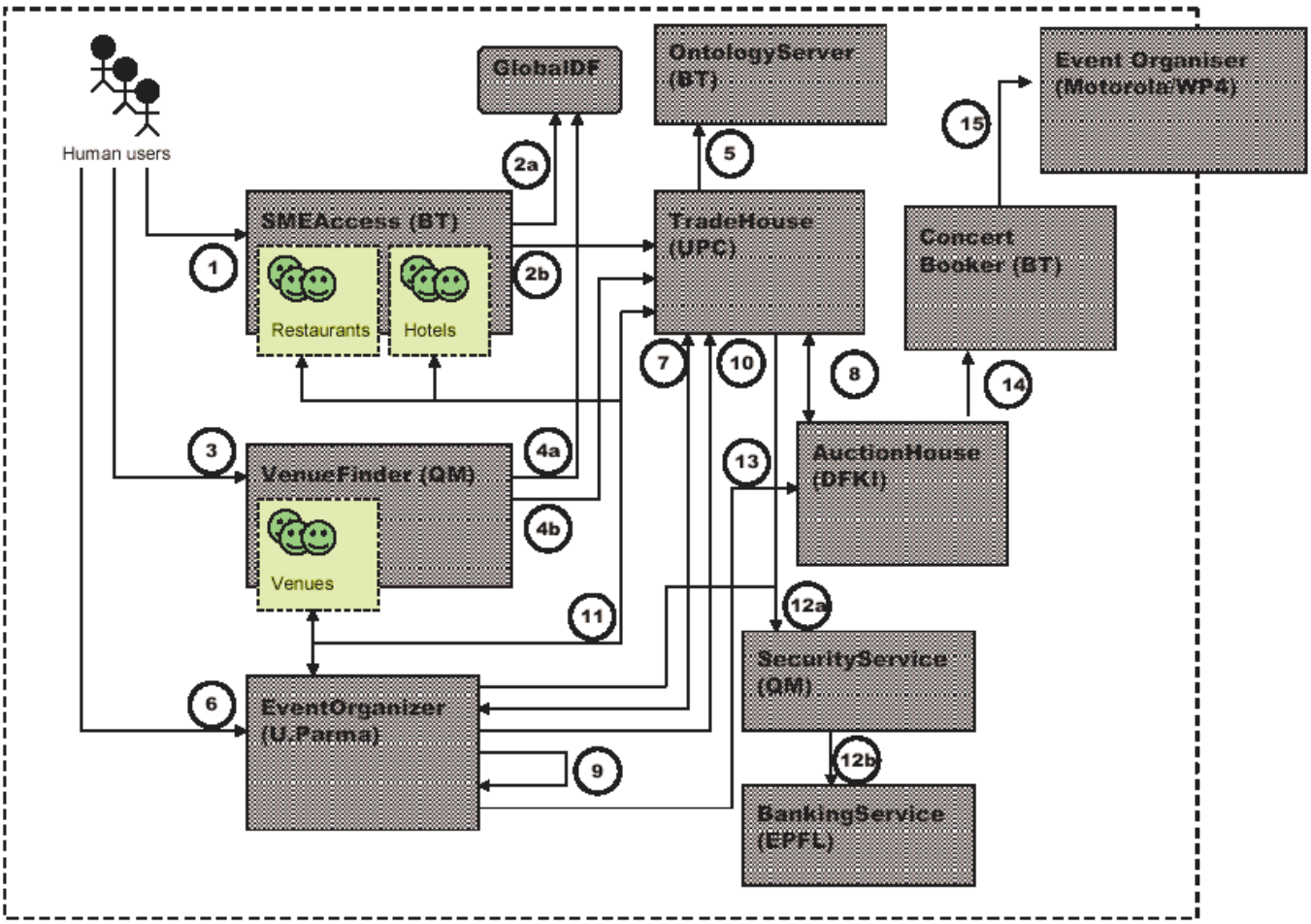

Fig. 3. Event organizer scenario

The whole process requires the cooperation of a number of partners. Each one of them can exploit the directory services of the Agentcities network to dynamically discover the location of the others.

- The Event Organizer directly interacts with a Trade House to search for venues and negotiate selected services.
- Other agent-based applications, as the Venue Finder and the SME Access, are responsible to offer goods on the Trade House and to negotiate them on behalf of their users.
- A Banking Service takes care of managing the banking accounts of the involved partners, securing all requests against tampering and eavesdropping.
- An Auction House is used to create auctions and sell the tickets of the new event.

The interesting part of the game is that these tickets are available for other agent-based applications. In the integrated demonstration staged at the end of the project an Evening Organizer, helping its user to arrange an evening out for example booking a restaurant and buying the tickets for a concert, was able to discover the new event and bid for some tickets on the Auction House.

B. Trade House

The Trade House is an agent-based application that allows the advertisement and negotiation of products. It is not limited to a fixed set of known products, but can manage any kind of tradeable goods described through a custom ontology. In this sense, we can say that the Trade House is product-indipendent, since it can load the ontologies that describe new products at run time. In fact, whenever an external user agent advertise a product defined through a new ontology, this is dynamically added to the system. The trading engine supports the negotiation of different attributes at the same time, trying to mediate the interest defined explicitly by buyers and sellers.

The main features of the Trade House are:

- *Ontology independence* – The Trade House supports dynamically-loaded, user-defined ontologies, both in the advertising and in the trading. Thus, agents are not required to map explicitly their ontology into a different one, already known by the system, but they can use

their own. Doing so, the Trade House is able to offer its services for any kind of product, or being more general, for any kind of concept defined by means of an ontology. The user-defined ontologies are indeed published in the Agentcities Ontology Server.

- *Constraint-based search* – Search into the product repository is done through constraints, which can be defined for each product's attribute. Thus, the search process might be considered as a matchmaking process, since the obtained results are bounded by user-defined constrains.
- *Federation* – Each instance of the Trade House can be federated either with other instances of the Trade House or with instances of the Auction House. Through federation, houses can share their product repositories to provide the users access to a wider market, which is a distributed one. This federated search works in a peer-to-peer fashion: each house forwards search requests to their neighbours and collect the obtained results, from which the user agent could decide to join another house.
- *Multi-attribute negotiation* – The Trade House contains a trading engine which can adjust the properties of a product to maximize the satisfaction of both the seller and the buyer. Such setting is evaluated into the intersection of the interest ranges defined by both parties, weighted with their preferences. A range and a preference are required for each attribute agents want to negotiate for.
- *Servant agents* – In order to ease the interaction with the Trade House, user agents can customize a servant agent by defining their own negotiation strategies and their own interests. Then, most of the required interactions between user agents and the system are delegated to their servant agent, reducing the complexity of usage in customer's side.

C. Auction House

The Auction House is an agent-based service for businesses that is designed to allow for a broad range of typical auction-style trading activities.

The main features of an Auction House are the following:

- *Concurrent auctions* – Users of the Auction House may concurrently create, participate in, and bid on multiple auctions of different type including English, Dutch, and Vikrey auctions.
- *Bidding strategies* – The Auction House provides different parameterized bidding strategies, appropriate for each type of auctions.
- *History functions* – Users, both auctioneers and bidders, are allowed to inspect their past or current activities on auctions, making the auction house even more convenient to use.
- *Constraint-based search* – Any user that is registered at an Auction House may search for items that are currently traded in the Auction House as commodities. Users may search for trading opportunities for a given type of commodity, either locally at the Auction House they are actually registered at, or globally in any Auction or Trade House that is physically connected to the local one via the Internet.
- *Integrated payment* – The auction house provides an integrated payment service to its users, exploiting the Agentcities virtual Banking Service. Payment calculations include the price of the traded item, auction house registration fee, and warranty.
- *Ontology independence* – Users may upload their individual ontology for describing their items to be auctioned in the auction house. These ontologies are published and available for use at the Ontology Server.

D. Ontology Service

The propose of building an ontology sever is to provide a common knowledge base in an agent community whereby a better understanding can be achieved amongst agents with minimum human involvement. Ideally, agents can communicate with each other without human intervention that is currently necessary to interpret the semantics of messages between them. The task can be divided into three significant parts. First, using a web ontology language to formalise domains by defining classes and properties of those classes, defining individuals and assert properties between them. Second, to support reasoning about these classes and individuals to the degree permitted by the formal semantics of the language. Last but not least, to build a management system that conducts both ontologies and users. The management system plays an important role in building knowledge bases that are shared and well-maintained by users. The idea is to set up an open collaborative environment that shows the following features:

- *Ontology language* – The ontology language used in the current version is DAML + OIL [4]. Jena library (HP Labs) is used as a base to maintain ontology repository. Some reasoning functions, such as querying relationship between entities amongst ontologies, consistency checking before asserting or retracting a relationship, are also

implemented.

- *Import statements* – Ontologies are centrally maintained and are linked to each other through ontology imports statements. With the imports mechanism, a distributed knowledge base hierarchy is built so that a new ontology can be plugged in easily and inherent the needed general knowledge base instead of building it totally from scratch. The imported ontology can also be treated as intersection with the other ontology that imports the same ontology. In order to maintain the consistency of the existing knowledge base, an assertion or a retraction that violent the consistency is disallowed by the system.
- *Role-based access control* – User access control is introduced to build a collaborative environment. Each ontology is associated with an access group managed by its owner. An ontology owner can configure the access rule for different kind of user and can assign users to the ontology's group. Every user is rated against some criteria, such as the number of successful assertions that has been introduced by the user. The rating of a user can be used automatically by program to judge whether the user can be added to a group. An ontology therefore can be set up as "open", which means a standard for joining the ontology' group is specified by the owner. Any user who satisfies the standard gains the access of the open ontology. Of course, any contribution to the ontology will still be checked for consistency with existing knowledge base to secure the system.
- *Interface* – An agent is associated with Ontology server to communicate with other agents through FIPA ACL Language. A set of APIs is provided for calling ontology server's functionality. An interactive web access written in JSP is also built to demonstrate those functionalities.

E. SME Access

SME Access is an agent generation and hosting service for businesses, specifically designed to allow them to deploy a presence on the Agentcities network. The system is primarily split into two elements: the web interface and the agent server.

- *Web interface* – The web interface is a collection of Java Server Pages, backed by a Struts architecture of form beans and servlets. The JSP pages provide user registration and login, as well as agent configuration, and agent control through the user's page. The information required to generate the agent is stored within the scope of the web application, and when a request is made to generate the agent, that data is passed via RMI to the agent server element.
- *Agent server* – The agent server uses the Zeus toolkit as the basis of most of its operation. It's primary purpose is the management of the agents, allowing them to be generated, run, stopped and deleted, all via RMI invocations. When the agents are run, they are linked to pre-existing classes that provide the behaviour associated with that service type. All agent management functions are accomplished through the web interface, but the agents, once run, will operate on the server without any interaction from the user. In the future, it is hoped that the agents will be able to notify the user of events that happen, possibly through linking to a database the user already operates.

The service permits registered users to configure a collection of agents to represent aspects of their business, then run them, whereupon they will register with a selected agent platform in order to advertise their presence to agents there.

- *Agent creation* – The application allows businesses to create agents according to templates within the application. These templates a compliant with instances of the same agent type on the Agentcities network. Currently, there are compatible templates for restaurants, hotels and cinemas. Therefore, any agents that can query those service types will be able to interact with the SME Access instances of them. All the user needs to provide is the information the agent will respond with, and will base its decisions on. This information is collected through the web interface to the application, which is an easy to use collection of forms.
- *Agent deployment* – Once the agents are created, they can then be generated and deployed to the hosting platform, where they will run and make themselves available for communication with other agents. Their status can be queried from the web interface, and their details changed
- *Advertisement* – In order to facilitate service discovery, the agents can register with any of the directory facilitators on the Agentcities network, and will write out DAML-S service descriptions [4] to an accessible location, creating both service instance descriptions and service profile descriptions. A process model has been written for each service type, and is placed in the same location. Therefore, remote agents can discover the location of the instances, and determine what

service they provide. Additionally, some of the templates provide compatibility with the Trade House system, and can register their goods with the system, providing another avenue for custom to arrive at the business.

F. Venue Finder

The Venue Finder Service provides B2B services to Market Service domains and personal agents. The service is modelled into two levels, in the first it allows venue based service domains to create instances of individual venues they would like to publish and sell. Whilst the second level, allows venue procuring services to access the system for querying, negotiating and booking of venues. The general motivation for this work is to provide venue-finder services to support the expanding use of thirdparty venues for corporate (business) entertainment of staff and clients, and to support private functions such as weddings and birthday celebrations.

- *Heterogenous venues* – The Venue Finder is able to semantically convert heterogeneous venues into a common communicable interactive system and additionally offer its services to the Trade House. The venues are generated by converting information regarding venues into a general ontology.
- *Advertisement on the Trade House* – When the venues load into the venue finder domain, it also attempts to register itself with the Trade House. The venues would then publish and overload its ontology over the market place ontology to enable negotiation between offers or contracts placed on the market place, and simultaneously provide semantic information of the actual venues through a generic ontology of the venue finder.
- *Integrated payment* – Relatively, the venue finder also provides the Trade House with payment information that enables the Payment Service to complete a sale and purchase agreement.
- *Match-making routines* – Apart from relying on the Trade House infrastructure to advertise and negotiate its venues, the finder can even provide information of venues to support service selection and aggregation through match-making routines. In fact the Venue Finder is able to directly match the constraints of venues against the requirements of the Event Organizer.

G. Banking Service

The expression “banking services” refers to the set of processes and mechanisms that a virtual agent-based banking institution offers to the agents accessing the Agentcities marketplace. This mainly includes:

- Electronic payment service for enabling agents to make and receive side payments.
- Account management service for creating, maintaining and closing bank accounts.

The banking service design consists therefore of two main sub-sets that are described in the following as two distinct frameworks, even if they both rely upon the same unique ontology for banking services.

The payment service includes the set of operations and mechanisms allowing the transfer of money between distinct accounts either under the control of the same bank or under the control of different banks. With the term “bank” we refer here to a virtual banking institution.

The account management service groups the set of actions and operations needed to manage bank accounts. It is possible to:

- Open a bank account.
- Dynamically verify the information about an account.
- Close an existing bank account.

All these operations require the agent requiring the service to be performed to be an “authorized” entity. Authorization and authentication of agents accessing the banking services has been implemented integrating the service with the security library and the infrastructure provided on the Agentcities network.

H. Security Service for e-Banking

Services being developed by the Agentcities project require, and would benefit from having security. In particular, security mechanisms have been developed to protect the access to the Banking Service, supporting core security requirements such as message confidentiality, message integrity and agent authentication. Additionally, an authorisation model is closely linked to the authentication model using simple policies or profiles. In order for an e-banking institution to support payment mechanisms for the purchase and sale of goods, an appropriate level of security is needed to secure e-commerce transactions within the Agentcities network.

Specifically, both the electronic payment and the account management services of the Banking Service are supported by the design and implementation of a distributed security service.

The services are developed using a modular approach where security processes and behaviours are separated to support ease of integration between various agent-based systems. The system provides two key functionalities:

- A Certificate Authority (CA) service which provides credential management services. It is published on an agent server hosted on the Agentcities network.
- A plug-in for agent security management. It is installed on clients and the Banking Service to provide the necessary security support. The security service acts as a plugin for agent systems to communicate securely by offering the following features: end-to-end confidentiality, mutual authentication, data integrity, and session and fingerprint management.

IV. Conclusion

Agentcities is certainly the greatest effort to create a global network of FIPA-compliant agent platforms. It is giving a great impulse toward the openness and interoperability of different agent platforms and agent-based applications, paving the way for a large and distributed society of agents.

In this context, where new cooperating and competing peers can rapidly appear, agents can show their ability to adapt to evolving environmental conditions, creating and maintaining their network of social relations.

This paper focused on a particular application that, relying on the marketplace infrastructure deployed on the Agentcities network, is able to dynamically discover and negotiate some services offered by different providers.

In particular it can search for and book all the venues required to organize an event, for example a concert or a conference, in a service-composition process. Once the event is organized, it can be again advertised as a compound service on the marketplace and eventually used by other agents to plan their actions and achieve their own goals.

Acknowledgment

Thanks to all Agentcities partners who contributed to the development of the project. This work is partially supported by the European Commission through the contracts IST-2000-28385 Agentcities.RTD.

References

[1] *Agentcities.RTD*, reference IST-2000-28385, http://www.agentcities.org/EURTD/
[2] Bergenti F., Botelho L. M., Rimassa G., Somacher M. *A FIPA compliant Goal Delegation Protocol*. In Proc. Workshop on Agent Communication Languages and Conversation Policies, AAMAS, Bologna, Italy, 2002.
[3] Castelfranchi C., Falcone R., *Socio-Cognitive Theory of Trust*. http://alfebiite.ee.ic.ac.uk/docs/papers/D1/ab-d1-cas+fal-soccog.pdf
[4] *DAML* http://www.daml.org/
[5] Durfee E. H. *Coordination of Distributed Problem Solvers*. Kluwer Academic Publishers, Boston, 1988.
[6] Finin T., Labrou Y. *KQML as an agent communication language*. In J.M. Bradshaw (ed.), Software Agents, MIT Press, (Cambridge, MA, 1997), 291-316.
[7] FIPA spec. XC00023H. *FIPA Agent Management Specification*. http://www.fipa.org/specs/fipa00023/
[8] FIPA spec. XC00037H. *FIPA Communicative Act Library Specification*. http://www.fipa.org/specs/fipa00037/
[9] *JADE Home Page*, 1999. Available at http://jade.cselt.it
[10] Jennings N. R., P. Faratin P., Lomuscio A. R., Parsons S., Sierra C. and Wooldridge M., *Automated Negotiation: Prospects, Mehods and Challenges*. In International Journal of Group Decision and Negotiation. 10(2), pages 199-215, 2001.
[11] Maes, P., *Agents that reduce work and information overload*. Communications of the ACM vol. 37 no. 7, pp 30 – 40, July 1994
[12] Pitt J., Kamara L., Artikis A.. *Interaction Patterns and Observable Commitments in a Multi-Agent Trading Scenario*. http://alfebiite.ee.ic.ac.uk/docs/papers/D1/abd1-pitkamart-ipoc.pdf.
[13] *The Agentcities Network* . http://www.agentcities.net/
[14] Wong H. C., Sycara K., *A taxonomy of middle-agents for the internet*. Agents-1999 Conference on Autonomous Agents 1999.
[15] Yuan, Yufei, Liang, T.P. & Zhang, Jason J. *"Using Agent Technology to Support Supply Chain Management: Potentials and Challenges"*, Michael G. DeGroote School of Business Working Paper No. 453, October 2001.
[16] Zlotkin G. and Rosenschein J. S. *"Mechanisms for Automated Negotiation in State Oriented Domains"*. In Journal of Artificial Intelligence Research 5, pages 163-238, 1996.