USING AGENT PLATFORMS FOR SERVICE COMPOSITION

Agostino Poggi, Michele Tomaiuolo, Paola Turci
Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Parma
Viale delle Scienze 181A, 43100 Parma, Italy

Keywords: Agent platforms, Service composition

Abstract: Autonomous software agents and semantic web technologies promise to pave the way for really dynamic and adaptive distributed applications. The Agenticities project is the first effort to create a large and open society, where collaborating and competing peers are able to interact effectively. This paper describes the services deployed on the Agentcities network, both at the infrastructure and application level, and shows how they can be successfully used for the intelligent and dynamic composition of simple services provided by different providers into more complex ones.

1 INTRODUCTION

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. 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 societies where agents act as buyers and sellers negotiating their goods and services.

A result of the project is in facts the demonstration of a working application that, relying on the marketplace infrastructure deployed on the Agentcities network, is able to dynamically discover and negotiate the services offered by different service providers. The collected services are then composed into a new compound service to be offered back on the marketplace.

Specifically, the rest of the paper is organized in the following way. Section 2 introduces some basic concepts, which are common for existing service composition platforms. Then, in section 3, the Agentcities network infrastructure is taken into consideration, and some undergoing developments are described. Each step of the service composition process is analyzed in section 4, and finally section 5 gives a description of all deployed services and applications.

2 RELATED WORKS

In (Chakraborty, 2001) a number of existing service composition frameworks are analyzed, showing they often rely on a common set of basic components and features provided by an underlying infrastructure level, including:

Service discovery – to find out all instances of a particular service. A useful feature is the ability to discover the services on the basis of their particular functionality, and not the way they are invoked.

Service Coordination and Management – to coordinate the services involved in the composition. This component usually provides (to different extents) fault tolerance, availability, efficiency, optimization of used resources, as to reduce the cost of the composite service. Scalability issues require this component not to be completely centralized.

Information Exchange Infrastructure – to enable communication among the different entities involved in the composition. It should allow the integration of services following different type of information exchange, as message passing and remote method invocation.

Poggi A., Tomaiuolo M. and Turci P. (2004).
USING AGENT PLATFORMS FOR SERVICE COMPOSITION.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 98-105
DOI: 10.5220/0002611600980105
Copyright © SciTePress

One of most advanced platforms to achieve service composition is *eFlow*, developed by HP (Casati, 2000). The system is made of a service discovery system, a composition engine and elementary services. The composition engine controls the state of all composition processes, which are modelled as graphs consisting of service nodes, event nodes and decision nodes. Moreover, *transaction regions* can be defined to identify sections of the graph to be executed as atomic operations.

Another platform for service composition is *SWORD*, described in (Ponnekanti, 2002). It is founded on a rule-based expert system that is used to find a plan for realizing the desired composite service. Each existing service is modelled as a rule, and its inputs and outputs are specified as preconditions and postconditions. The sequence of rules activated by the rule-engine represents a plan for the composition.

A different approach is used instead in the *Ninja* service composition platform developed at UC, Berkeley (Ninja, 1999). Its main aim is enabling clients with different characteristics and network connection abilities to invoke a set of existing network services. One of the founding concepts is that of a service path, matching the *pipe/filter model* of software architecture. It consists of a set of operators, modelling the invoked services, and connectors, network connections to transport data from one operator to the following one. A logical path is created using a shortest path strategy.

The same ideas are at the basis of *Paths* (Kiciman, 2001) and other similar projects (Kiciman, 2000). All these systems will probably take great benefit from the standardization of a formalism to represent service interfaces (OWL-S, 2003). In (Raman, 2003) the concept of path is used for addressing load balancing and stability issues. An analogy is supposed to exist between quality of service issues in routing algorithms and service composition, and in both cases the cost of using a particular link is supposed to be inversely proportional to its available capacity. To help selecting a sequence of services for a path, i.e. a composite service, the *least-inverse-available-capacity* metric is introduced, modelled after the least-distance metric in network literature.

3 AGENTCITIES NETWORK

The Agentcities network (Agentcities.net, 2003) 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, which can connect their own agent-based services, making the network really open and continuously evolving.

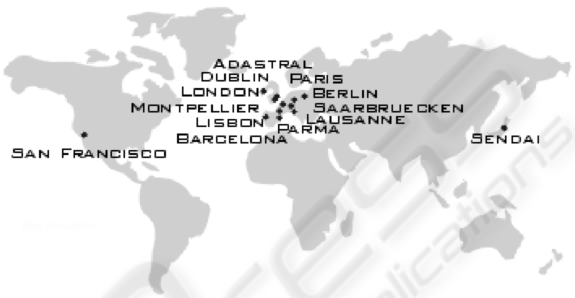

Figure 1: The Agentcities backbone

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 heterogeneous technologies, including Zeus, FIPA-OS, Comtec, AAP, Agentworks, Opal. More than 2/3 of them are based on JADE (Jade, 2004) and its derived technologies, as LEAP and BlueJADE.

3.1 Network Architecture

The structure of the Agentcities network architecture version 1.0 consists of a single domain with a collection of FIPA 2000 Agent Platforms deployed in it (FIPA, 2003). All these agent platforms are visible to one another and can exchange messages using HTTP 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 network services, hosted on a 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 alike.

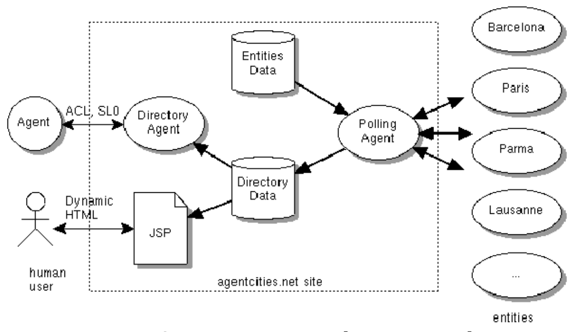

Figure 2: Agentcities Network generic architecture

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.

3.2 Network Architecture 2.0

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.

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.

3.3 P2P Agent Directory Service

Another effort we are deploying, to make the Agentcities network sounder, is the development of a DF founded on a JXTA peer to peer network (JXTA, 2003).

JXTA technology is a set of open protocols allowing any connected device on the network to communicate and collaborate in a peer to peer fashion. JXTA peers can form peer groups, which are virtual networks where any peer can seamlessly interact with other peers and resources, be them connected directly or through intermediate proxies.

In (FIPA, 2003) a set of new components and protocols are described, to allow the implementation of a DF-like service on a JXTA network. These include:

- *Generic Discovery Service* – a local directory facilitator, taking part in the peer-to-peer network and implementing the *Agent Discovery Service* specifications.
- *Agent Peer Group* – a child of the JXTA *Net Peer Group* that must be joined by each distributed discovery service.
- *Generic Discovery Advertisements* – to handle agent or service descriptions, for example FIPA *df-agent-descriptions*.
- *Generic Discovery Protocol* – to enable the interaction of discovery services on different agent platforms. It's a request/response protocol to discover advertisements, based on two simple messages, one for queries and one for responses.

4 DYNAMIC 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.

4.1 Service advertisement

The services provided on the Agentcities network can be advertised on its distributed marketplace infrastructure, which is domain and ontology independent.

Advertisements can represent arbitrary objects given they are encoded in *SL*. Advertised objects must be associated with their own ontology, which can later be used to match the offered objects with demanded ones. No assumption is made on the ontology itself, used to describe the concepts of the application domain. The only constraint is that both sellers and buyers agree on a common ontology. This means that the products searched for and negotiated by the buyers, must be expressed in the same ontology used by the sellers, otherwise no matches will be found.

A number of different constraints can be associated with each field of the advertised objects, allowing them to assume only values in a given continuous range, or in a discrete set.

Moreover, the service providers can express their interests, as a preference for lowest or highest values in the allowed range. These will be taken into account during the negotiation process, when the marketplace will propose a mediation between the interests of the sellers and the ones of the buyers.

4.2 Constraint-based service discovery

The buyers operating on the marketplace can search for advertised products and services. This allows to discover dynamically the presence of new products and service providers.

The search is a process where the products advertised on the marketplace are matched against a product pattern provided by the buyer agent.

When searching for new products, buyers can express a list of constraints they want to be satisfied. Only products having a not empty intersection between the constraints of the buyer and the seller will be listed.

The match process takes into account only products expressed into a given ontology, chosen by the buyer. The marketplace does not provide any mechanism to fit requests and offers for similar products, but described in different ontologies. A previous agreement on a shared ontology is supposed to exist between the buyer and the seller.

The search can be limited to a single marketplace, or it can be forwarded to other federated marketplaces, too, in a peer-to-peer fashion.

4.3 Service selection

Service composition in the Agentcities network is achieved as the selection and negotiation of different simple services, allowing the arrangement of the desired composite service.

The composite service is described by the user as a set of individual services and a list of required features, for example a room in a three star hotel and a table in a French restaurant. Then, a list of cross constraints among the different services can be set, for example fixing a maximum distance between the two venues.

The simple services, as well as the composite one, are advertised and discovered on the marketplace infrastructure. But the marketplace allows searching only for one item at a time, meaning that it’s not possible to fix cross-constraints among the different needed services.

This duty has to be carried on at the application level, where the constraints relating fields of different services must be validated. For example, after having received a list of hotels and restaurants as result of two searches, then checking they’re close enough to not annoy the customer is a duty of the application.

While constraints regarding each individual service can be validated by the marketplace infrastructure, the constraints linking fields of different services must be validated by the application arranging the composite service.

4.4 Service negotiation

If the buyer agent finds a suitable set of services, then it can start a negotiation with the corresponding providers.

The negotiation can be carried on directly by both parties, or they can delegate a servant agent, hosted on the marketplace platform, to achieve an agreement satisfying both the interests of the buyer and the seller.

If a servant agent is used, the delegating party can express his own interests as a range or a discrete set of allowed values, and a preference for certain values. For example, while the buyer will probably have some budget constraints and a preference for lowest prices, the seller instead will fix a minimum price and a preference for highest prices.

The servant agents can be instructed about the negotiation strategy to adopt. A number of different, ready-to-use strategies are provided by the marketplace infrastructure.

The negotiation process is not limited to the price attribute, but can regard a number of different attributes at the same time. For example both the price and the duration of a service can be negotiated at the same time.

4.5 Secure payment

If an agreement is reached for each constrained property of the negotiated service, then the negotiation process is considered successful. In this case a contract is proposed to both parties, which finally have to pay the price they agreed on. Probably, before final signature, the contract will be submitted to the end user for his definitive authorization. At this point, a banking service must be contacted to transfer the money from the buyer’s account of the seller’s one. The Agentcities e-banking service provides means to open, verify and close on-line accounts, and for the electronic payment of traded good and services.

The e-banking service is associated with a security infrastructure, based on a certification authority. Each message related to the e-banking service is encrypted and signed, using the cryptographic keys that are linked to each user of the service. This link is secured by means of identity certificates signed by the trusted certification authority. Linking each user with his own banking accounts and verifying the consistency of the requested operations are supposed to be duties of the e-banking service.

4.6 Semantic issues

A great part of the depicted process, from the advertisement of provided services to the final payment, is made possible only thanks to the underlying use of ontologies and semantic-web concepts. Buyers and sellers certainly must agree on a common understanding of what a particular field is intended for. Otherwise they would never be able to negotiate and agree on a price, for example.

However, one of the most important limitations of the project is that currently, while easily customizable at construction time, most of deployed agents are not able to acquire new semantic notions of the world at run-time.

For this purpose, we are developing an add-on for the JADE framework, to facilitate the dynamic, at run-time, acquisition of new ontologies, specified in the standard OWL language (OWL, 2003) and published on the Agentcities Ontology Server. In facts, we think the whole project will greatly benefit from the possibility to easily discover and load ontologies about new kinds of traded goods. This will give the deployed agents greater flexibility, and more ability to adapt to changing environmental conditions, as they will be able to negotiate new services as soon as they are made available.

5 DEPLOYED SERVICES

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.

5.1 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 helps the organization of an event, booking all needed venues, arranging all needed services, and finally selling the tickets.

Using a web interface, the user can list a set of needed services, fixing desired constraints on each individual service and among different services. The

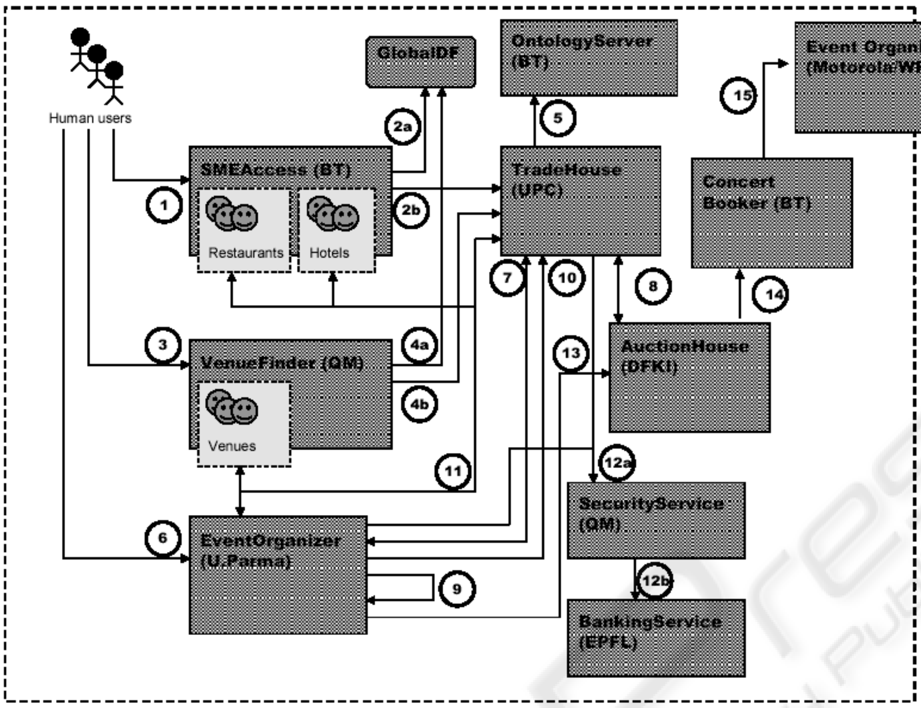

Figure 3: Deployed services

global goal is then split into sub-goals, assigned to skilled solver agents (Somacher, 2002).

The rest of the process requires the cooperation of a number of partners, which can exploit the directory services of the Agentcities network to dynamically discover the location of the others.

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, namely the *Venue Finder* and the *SME Access*, and a list of contracts is proposed to the user for final acceptance. The *Banking Service* takes care of managing the banking accounts of the involved partners, securing all requests against tampering and eavesdropping. 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. In this case the *Auction House* is contacted to create an auction and sell tickets for the new event.

The interesting part of the game is that the tickets are available for other agent-based applications. In the integrated demonstration staged at the end of the project, an *Evening Organizer*, helping 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.

5.2 Markeplaces

The marketplace infrastructure of the Agentcities network comprises two different kinds of services: the *Trade House* and the *Auction House*.

The main characteristic of the Trade House is the *multi-attribute negotiation*. The Trade House contains a trading engine which can adjust multiple 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.

Instead the main function of the Auction House is to host *different 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. The Auction House provides different parameterized *bidding strategies*, appropriate for each type of auctions. 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.

Both the Trade House and the Auction House services show an interesting set of features, including:

- *Ontology independence* – They support dynamically-loaded, user-defined ontologies, both in the advertising and in the trading. Doing so, houses are able to offer their 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 two services can be federated with other instances. 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.
- *Servant agents* – In order to ease the interaction with the houses, user agents can customize a servant agent by defining their own negotiation and auction 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.

5.3 Ontology Service

The purpose of building an *Ontology Server* is to provide a common knowledge base in an agent community. A web ontology language is used to formalise domains by defining classes and properties of those classes, defining individuals and assert properties between them. It supports reasoning about these classes and individuals, to the degree permitted by the formal semantics of the language. The ontology language used in the current version is DAML+OIL (DAML, 2003). Jena library (HP) is used as a base to maintain ontology repository. Ontologies are centrally maintained and are linked to each other through ontology imports statements, so that a new ontology can be plugged in easily and inherent the needed general knowledge base instead of building it totally from scratch. An agent is associated with Ontology server to communicate with other agents through FIPA ACL Language.

5.4 Service Providers

SME Access is an agent generation and hosting service for businesses, specifically designed to allow them to deploy a presence on the Agentcities network. A web interface provides user registration and login, as well as agent configuration, and agent control through the user's page. The underlying agent server uses the Zeus toolkit as the basis of most of its operation. The application allows businesses to create agents according to templates within the application. Currently, there are compatible templates for restaurants, hotels and cinemas, allowing to advertise on the Trade House.

The *Venue Finder* service allows venue based service domains to create instances of individual venues they would like to publish and sell. It allows venue procuring services to access the system for querying, negotiating and booking of 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.

5.5 Secure Banking Service

The expression *Banking Service* refers to the set of processes and mechanisms that a virtual agent-based banking institution offers to the agents accessing the Agentcities marketplace. The banking service design consists therefore of two main sub-sets, both relying upon the same unique ontology for banking services:

- *Electronic payment service* – for enabling agents to make and receive side payments. It allows the transfer of money between distinct accounts either under the control of the same bank or under the control of different banks.

- *Account management service* – for creating, maintaining, verifying and closing bank accounts

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. A Certificate Authority (CA) service which provides credential management services is published on an agent server hosted on the Agentcities network. A plug-in for agent security management is installed on clients and the Banking Service to provide the necessary security support.

6 CONCLUSIONS

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.

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. Even if the scope of the prototype application was not to advocate any particular technology or approach but was simply demonstrative of the network potentiality, it can be interesting to draw some remarks about the traits of the system deployed.

What mainly characterizes it is its decentralized architecture based on whatever number of marketplace instances. Considering the ability of the agents to assemble a compound service on customers' request, the system represents a good support for dynamic adaptive composition, by taking maximum advantage of the currently available services. The distributed nature of the system, on the other side, makes any control on the composition process difficult, for example to add load balancing, fault tolerance, and transactional features.

ACKNOWLEDGMENTS

Thanks to all Agentcities partners who contributed to the development of the project. This work is partially supported by the European Commission through the contract ALA/2002/049-055 @lis TechNet, and by the "Ministero dell'Istruzione, dell'Università e della Ricerca" through the contract ANEMONE.

REFERENCES

Agentcities Network, 2003. http://www.agentcities.net/

Agentcities.RTD, 2003. Reference IST-2000-28385. http://www.agentcities.org/EURTD/

Casati F., Ilnicki S., Jin L.J., Krishnamoorthy V., Shan M.C. Adaptive and Dynamic Service Composition in eFlow. HPL-2000-39. March, 2000.

Castelfranchi C., Falcone R. Socio-Cognitive Theory of Trust. http://alfebiite.ee.ic.ac.uk/docs/papers/D1/ab-d1-cas+fal-soccog.pdf

Chakraborty D., Joshi A. Dynamic Service Composition: State of the Art and Research Directions. December, 2001.

DAML, 2003. http://www.daml.org/

Durfee E. H. Coordination of Distributed Problem Solvers. Kluwer Academic Publishers, Boston, 1988.

Finin T., Labrou Y. KQML as an agent communication language. In *J.M. Bradshaw (ed.), Software Agents, MIT Press, (Cambridge, MA, 1997)*, 291-316.

FIPA, 2003. http://www.fipa.org/

JADE, 2003. http://jade.tilab.com/

Jena, 2003. http://jena.sourceforge.net/

JXTA, 2003. http://jxta.org/

Kiciman E., Fox A. Separation of Concerns in Networked Service Composition. May, 2001.

Kiciman E., Fox A. Using Dynamic Mediation to Integrate Cots Entities in a Ubiquitous Computing Environment. 2000.

Ninja. UC Berkeley Computer Science Division. http://ninja.cs.berkeley.edu/overview. 1999

OWL, 2003. http://www.w3.org/TR/owl-features/

OWL-S, 2003. http://www.daml.org/services/owl-s/1.0/

Ponnekanti S.R., Fox A. Sword: A Developer Toolkit for Building Composite Web Services. *WWW 2002*.

Raman B., Katz R. H. Load Balancing and Stability Issues in *Algorithms for Service Composition, IEEE Infocom* 2003.

Somacher M., Tomaiuolo M., Turci P. Goal Delegation in Multiagent Systems. 2002.

Wong H. C., Sycara K., A taxonomy of middle-agents for the internet. *Agents-1999 Conference on Autonomous Agents*, 1999.

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.