Chapter XVI
Multi-Agent Systems for Semantic Web Services Composition

Agostino Poggi
Università degli Studi di Parma, Italy

Paola Turci
Università degli Studi di Parma, Italy

ABSTRACT

The vision which is making its way in information technology is to encapsulate organizations’ functionalities within appropriate interfaces and advertise them as one or more Web services, which could be integrated, when brought into play, in workflows. This innovative idea brings with it new outstanding opportunities but also new great issues, related mainly to the ability to automatically discover and compose Web services. Several researchers belonging to the agent community are convinced that this technical area is a natural environment in which the agent technology features can be leveraged to obtain significant advantages. This chapter is aimed at briefly recalling the major results achieved by agent community and showing how their exploitation in the area of service-orientation systems could be very promising.

INTRODUCTION

Industry has been and still, more than ever, is interested in executing business functions that span multiple applications. This demands high-levels of interoperability and a more flexible and adaptive business process management. Most of the technology and market research companies, which provide their clients with advice about technology impact on business and consumers, agree on the fact that the adoption of a SOA paradigm is strategic and should be part of the most
forward-looking software projects. Nevertheless the paradigm shift is still quite challenging.

Many researchers belonging to the agent community are convinced that this technical area appears to be a natural environment in which the agent technology features can be leveraged to obtain significant advantages. Multi-agent systems, in fact, can play an important role in a service-oriented scenario, by efficiently supporting distributed computing and the dynamic composition of Web services. It is plain, in fact, that service-oriented technologies cannot provide by themselves the autonomy and social and proactive capabilities of agents. Agents, taking advantage of their social ability, exhibit a flexible coordination that makes them able to both cooperate in the achievement of a global goal and compete in the distribution of resources and tasks. However, what comes out is that the agent technology has to be appropriately engineered and integrated with other key technologies in order to provide a real powerful approach, combining ubiquity, context-awareness and intelligence. Driven by such motivations, a number of research works have been undertaken with the aim of tackling the problem of integrating service-oriented technologies with multi-agent systems.

This chapter has the goal of reporting a synopsis of these works and providing evidence of why multi-agent systems may be considered one of the most promising technologies for semantic Web services composition.

**BACKGROUND**

The first subsection includes a brief survey of the literature in the areas of standard Web services, workflow and semantic Web services technologies with the objective of showing the scenario in which the agent possibly contribution should be set and at the same time to give a short preamble acting as a motivation and rationale of the research work done by the agent community.

There are plenty of papers on the subject of agent and multi-agent system definition. The purpose of the second subsection is not to be comprehensive but simply to establish some basic concepts.

**Service-Oriented Applications: State of the Art**

The new vision of a Web constituted by dynamically interoperating nodes and the ever increasing demand for high-levels of interoperability by organizations that want applications to have broader reach, have stimulated the rapid growth of novel standards, technologies and paradigms with the aim of giving an answer to such problems. The most appropriate response to this need seems to be a service-oriented architecture (SOA), i.e. a system assembled from a loosely coupled collection of services and in particular of Web services—the integration technology preferred by organizations implementing SOA.

The basic specifications of Web services provide the infrastructure that supports the fundamental operations of a SOA. They are based on XML and define layers of abstraction including the message format description and communication protocol (SOAP), the operations performed by Web services and the related messages exchanged (WSDL), publishing and discovery capabilities (UDDI). This allows accomplishing an explicit agreement on the way Web services interact, providing a basic service-oriented middleware which can be exploited by possible higher layers enabling the realization of an effective SOA.

While Web services may be used in an isolated way to accomplish a specific business task, the need to aggregate multiple services in a new single meaningful composite service or to integrate them as part of workflow processes is more and more felt. During the last years a lot of research works have been undertaken, ranging from theoretical foundations, standardization efforts to concrete tools, technologies and real-world application case
Multi-Agent Systems for Semantic Web Services Composition

Multi-Agent Systems for Semantic Web Services Composition studies (Alonso et al. 2005; Dietrich et al., 2007; Dustdar & Schreiner, 2005; WS-Coordination, 2007). However, a still open problem is that the information and the research activities in this area are quite fragmented (Papazoglou et al., 2006).

Restricting the ambit of investigation and focusing mainly on the standardization efforts related to service composition based on workflow patterns for the business process management, a prominent standard that provides for explicit process modeling and execution, based on Web services, is particularly interesting, i.e. the business process execution language for Web services (WS-BPEL).

The WS-BPEL (WS-BPEL, 2007) specification defines an XML-based language for the formal description of a business process based on Web services orchestration. It is an open standard approved as an OASIS standard. A WS-BPEL workflow is a structured XML document composed of three main parts: (i) the definition of the process attributes; (ii) the definition of the execution context, and (iii) the activities to be executed. Due to industry’s increased interest on business process management and the wide acceptance of WS-BPEL as the language to use in the workflow definition, several vendors are producing software tools for workflow design, specification and enactment.

WS-BPEL is particular significant if used in combination with another W3C candidate recommendation: i.e. the Web service choreography description language (WS-CDL, 2005). As a matter of fact, the two specifications are complementary: WS-BPEL is about orchestration, it is based on a central coordinator that is responsible for invoking and combining the single sub-activities, while WS-CDL is about the modelling and execution of cross-organizational business processes, specifying the common observable behaviour of all participants engaged in business collaboration.

The main drawback of these standards, specifications and also of other work based on XML is that they only support static service composition. The composition takes place during design-time when the architecture and the design of the software system are planned, in other words they do not provide support for dynamically discovering new Web services or exploiting new of them in case of unpredictable events. This may be suitable on condition that business partners and service components do not or only rarely change, but in a dynamic environment characterized by continuous evolution where companies need a business process that is capable of ongoing modifications, this often represents an undesirable limit.

Relying, indeed, exclusively on XML one can reach only a syntactic interoperability. Expressing message content in XML does not make possible semantic “understanding” of the message contents and as a consequence programmers have to reach explicit agreement on both the format of the messages exchanged and the way Web services interact.

On the one hand the syntactic description is essential since it provides information about the structure of input and output messages of a service and how to invoke the service itself, but on the other hand semantics is needed to describe what a service really does.

On the research front, in response to these limitations several proposals have been submitted to W3C in order to make Web services semantically described and thus make the description of Web service interfaces unambiguous, paving the way for semi-automatic or even automatic discovery and composition of services. From an analysis of such submissions, in our opinion the most prominent are: OWL-S (OWL-S, 2004) and SAWSDL (SAWSDL, 2007). Both approaches are based on the reference to one or more domain ontologies, in which the concepts, referred in the semantic part of the service description, are defined.

OWL-S, a proposal submitted to the W3C by a community of researchers, is characterized by a quite comprehensive approach to the semantic orientation of the Web service description. It defines an OWL-based service ontology.
Multi-Agent Systems for Semantic Web Services Composition

(domain-independent), to which one has to refer for a semantic description of a service. Such a description keeps the semantic information into a separate file, but has a clearly defined grounding to the WSDL document of the real service. To facilitate the adoption of OWL-S specifications the OWL-S researchers developed an OWL-S integrated development environment, called OWL-S IDE (Srinivasan et al., 2006). OWL-S IDE is a framework with the aim of supporting the complete lifecycle of semantic web services: that is the development of OWL-S descriptions, the advertisement, discovery and the execution of OWL-S web services. The most significant and central feature of that work is represented by the support to semantic Web service discovery, achieved by extending the standard UDDI registry specification incorporating OWL-S semantics.

SAWSDL, which became a recommendation of the W3C last year, follows a lightweight, incremental approach. It proposes a small set of extensions to WSDL in order to associate semantic annotation, called model reference, to the WSDL document itself. These annotations relate elements belonging to the WSDL service descriptions with domain-specific semantic elements, e.g. concepts belonging to domain ontologies, but, as opposed to OWL-S, being agnostic to the semantic representation language. Besides semantic annotations, SAWSDL also introduces schema mapping annotations, to be added to XML Schema element declarations and type definitions, with the aim of specifying mappings between semantic data and XML. Such mappings could be used during the invocation phase, particularly when a software agent or more in general a software application playing the role of mediator is used. In fact, model references could be used to determine if a service match client requirements, during the discovery phase, but can also be used during the invocation phase. In a likely scenario, an agent invokes a Web service referring to a high level description of the service itself, based on ontological concepts. In view of the fact that there may be mismatches between the semantic description of the service and its real invocation, the lowering schema mapping is used to obtain the concrete data required to invoke the service; vice versa, in the case of the lifting schema mapping. At the moment, both mappings rely solely on XSLT transformations between RDF/XML, which is a tedious and often an error-prone task. Since it is quite a novel W3C recommendation, the open source tools (e.g. API for handling SAWSDL documents and tools for annotating WSDL services to produce SAWSDL documents) realized by the research community are still in a preliminary phase.

To improve the service discovery phase, Web services registries represent an important component of the semantic Web service infrastructure. As a matter of fact, the need for a dynamic composition of services requires the identification of services based on their capabilities in order to recognize those services that can be combined together. As stated before, the UDDI standard for Web services registries provides publishing and inquiry functionalities, but since it does not make use of semantic information, it fails to address such issue. OWLS-UDDI matchmaker (Srinivasan et al., 2005) and the hybrid matchmaker, called OWLS-MX (Klusch et al., 2006), are at present the most visible of a few semantic Web service matchmakers which have been developed in the last years. Both are based on a matching algorithm which gives as a result a list of suitable services associated with a degree of match, giving the information of how much an advertisement and a query are semantically compatible. They perform service I/O based profile matching, exploiting ontological concepts as values of service input and output parameters. The peculiarity of OWLS-MX is that it is based on a hybrid matching algorithm, which utilizes both logic based reasoning and information retrieval techniques for semantic service discovery, outperforming the previous approaches based only on logic reasoning. The main drawback of both approaches is that they do not consider in their matching algorithm advance
reasoning on logically defined preconditions and effects and additional filtering on non functional requirements.

Agents and Multi-Agent Systems

Agent, software agent and multi-agent system are terms that have found their way into a number of technologies and have been largely used, for example, in artificial intelligence, databases, operating systems and computer networks literature. Although there is no single definition of an agent (Genesereth & Ketchpel, 1994; Wooldridge & Jennings, 1995; Russell & Norvig, 2003), all definitions agree that an agent is essentially a special software component that has autonomy that provides an interoperable interface to an arbitrary system and/or behaves like a human agent, working for some clients in pursuit of its own agenda. Even if an agent system can be based on a solitary agent working within an environment and if necessary interacting with its users, usually they are based on multiple agents and so they are called multi-agent systems. These multi-agent systems are suitable for modelling complex systems, introducing the possibility of agents having common or conflicting goals. Agents may interact with each other both indirectly (by acting on the environment) or directly (with communication and negotiation among them). Moreover, agents may decide to cooperate for mutual benefit, or may compete to serve their own interests.

There is no universally accepted definition of the term agent, however, the different definitions allow distinguishing between the features that all the agents should own and the features that some special kinds of agents should provide. In particular, an agent should be autonomous, because it should operate without the direct intervention of humans or others and should have control over its actions and internal state, should be social, because it should cooperate with humans or other agents in order to achieve its tasks, should be reactive, because it should perceive its environment and respond in a timely fashion to changes that occur in the environment, should be pro-active, because it should not simply act in response to its environment, but should be able to exhibit goal-directed behaviour by taking the initiative. Moreover, if necessary an agent can be mobile, showing the ability to travel between different nodes in a computer network, it can be truthful, providing the certainty that it will not deliberately communicate false information, it can be benevolent, always trying to perform what is asked to it, it can be rational, always acting in order to achieve its goals, and never to prevent its goals being achieved, and it can learn, adapting itself to fit its environment and to the desires of its users.

From the technological point of view, a lot of work has been done in the last decade for spreading the use of agents for the realization of software applications. Several software systems and technological specifications are the results of such work. Among them the main results are the definition of FIPA specifications (FIPA, 2000), a set of specifications oriented to support the interoperability between heterogeneous agent systems, and an agent development framework, called JADE (Bellifemine et al., 2008, JADE, 2008), that implements such specifications and supports the interoperability between agents and the most common technologies currently used for realizing software applications.

AGENTS CAPABILITIES FOR WEB SERVICES COMPOSITION

The purpose of this section is to provide a summary of the achievements by the agent community, without aiming at being exhaustive, bearing in mind the great amount of work done, but purely highlighting what we consider relevant because their particular advantages when applied to the specific field of service-oriented applications.
Agents are designed to operate in dynamic and uncertain environments, making decisions at run-time. Moreover, agents take advantage of their social ability to exhibit a flexible coordination that makes them able to both cooperate in the achievement of a global goal and compete in the distribution of resources and tasks. Therefore, agent technology may be considered an interesting means for the enhancement of the current Web services composition solutions and even more for the realization of more effective and reliable Web services composition solutions.

In fact, the agents ability of operating in dynamic and uncertain environments allows coping with the usual problems of failures or unavailability of services and the consequent need of finding substitute services and/or back tracking the system in a state where execute an alternative workflow. Moreover, the capabilities of some kinds of agent of learning from their experience make them able to improve their performance over the time avoiding untrusted and unreliable providers and reusing successful solutions.

Even more important it is the agents’ ability in coordinating themselves, because it can be the main ingredient for the development of flexible, intelligent and automatic Web services composition solutions. Coordination among agents can be handled with a variety of approaches including, negotiation, contracting, organizational structuring and multi-agent planning, and all these approaches can be used to cope with some problems of Web services composition.

Negotiation is the communication process of a group of agents in order to reach a mutually accepted agreement on some matter (Jennings, 2001). Negotiation can be competitive or cooperative depending on the behaviour of the agents involved. Competitive negotiation is used in situations where agents have independent goals that interact with each other. They are not a priori cooperative, share information or willing to back down for the greater good, namely they are competitive. Cooperative negotiation is used in situations where agents have a common goal to achieve or a single task to execute. Negotiation can be used with success for automating some operations both in the design and the execution phases of the provision of a composite service. In fact, while in the design phase, negotiation techniques help customers in selecting the most appropriate component services and in reaching an agreement about all the issues charactering the provision of such services, in the execution phase, negotiation can be useful for the redistribution of the component services among both the different servers of a provider and even among different providers.

Among the negotiation techniques, contracting is probably the best way for searching the most appropriate services that satisfy a specific contract. Contracting is a negotiation technique based on a decentralized market structure where agents can take on two roles, a manager and contractor and where managers tries to assign tasks to the most appropriate contractors (Smith & Davis, 1980). The basic premise of this form of coordination is that, if an agent cannot solve an assigned problem using local resources/expertise, it will decompose the problem into sub-problems and try to find other willing agents with the necessary resources/expertise to solve these sub-problems. The problem of assigning the sub-problems is solved by a contracting mechanism consisting of: 1) contract announcement by the manager agent, 2) submission of bids by contracting agents in response to the announcement, and 3) the evaluation of the submitted bids by the contractor, which leads to awarding a sub-problem contract to the contractor(s) with the most appropriate bids.

Contracting solves the problems of searching and selecting the services that can be used in the realization of a composite service, but does not give any help in identifying the contracts that the different component services must satisfy and in defining the way in which they are executed to obtain the composite service. To cope with these two issues, agent can take advantage of multi-agent
planning techniques. Multi-agent techniques enable agents to allow the realization of plans that move agents towards their common/individual goal preventing any possible interference among the actions of the different agents (Tonino et al., 2002). In order to avoid inconsistent or conflicting actions and interactions, agents build a multi-agent plan that details all the future actions and interactions required to achieve their goals, and interleave execution with more planning and re-planning. Multi-agent planning can be either centralized or distributed (Rosenschein, 1982; Durfee, 1999). In centralized multi-agent planning, there is usually a coordinating agent which, on receipt of all partial or local plans from individual agents, analyses them in order to identify potential inconsistencies and conflicting interactions (e.g., conflicts between agents over limited resources). The coordinating agent then attempts to modify these partial plans and combines them into a multi-agent plan where conflicting interactions are eliminated. In distributed multi-agent planning, the idea is to provide each agent with a model of other agents plans. Agents communicate in order to build and update their individual plans and the models of other agents until all conflicts are removed.

Multi-agent planning techniques are useful to identify the contracts that the component services must satisfy and to design the composite service, but cannot be used for identifying the providers offering the needed component services; therefore, it must be applied in conjunction with a means able to identify such set of components and, of course, contracting is the most appropriate one. Moreover, multi-agent planning can be used together with contracting during the execution of a composite service when there are one or more malfunctioned component service and for each of these service there is not another available service able to replace it. In these cases, multi-agent planning is able to redesign part of the composite service either replacing a component service with a set of component services realizing together the task assigned to the single service or replacing a set of component services with another set of services that execute the same task that have been assigned to the initial set of services.

Organizational structuring are some coordination techniques that are complementary to both contracting and multi-agent planning techniques. Organizational structuring techniques allow defining the organization that govern the interaction among the agents of a system, i.e., define the information, communication, and control relationships among the agents of a system (Hollinger & Lesser, 2005). Therefore, the use of such techniques together with contracting and multi-agent planning ones can be useful to guarantee a higher reliable and efficient provision of services by defining the most appropriate organization to manage and monitoring the distributed execution of a composite service and to localize the work needed to be done to cope with some component services.

**MULTI-AGENT SYSTEMS FOR WEB SERVICES COMPOSITION**

From the above considerations, it clearly emerges that the subject of service composition turns out to be vast and enormously complex and more work needs to be done in order to realize real flexible, adaptive intelligent service-oriented systems. As a matter of fact, current SOA implementations are still restricted in their application context to being an in-house solution for companies.

In the attempt to delineate an effective solution, some researchers have envisaged as strategic the integration of SOA with both semantic and Web technologies (Vitvar et al., 2007). Others have turned their attention towards the agent technology, integrated with semantic and Web technologies, as an interesting means for the realization of more effective and reliable service-oriented systems and for SOA to be successful on a worldwide scale (Huhns et al., 2005).
Integrating SoftwareAgents and Web Services

The problem of composing Web services exploiting agent technology can be reduced to three fundamental problems: (i) the management of the interactions between agents and Web services; (ii) the execution of a workflow or more in general of a plan that describes how Web services interact; (iii) the discovery of the Web services that perform the tasks required in the plan.

The first issue that the agent community has had to cope with has been the integration of the two technologies, which has implied a mapping between the different semantic levels of the two paradigms or patterns of communication. Before analyzing the approaches to the problem it is useful to point out the issues connected. In analyzing and comparing the two technologies, as far as the agent community is concerned, we have referred to the FIPA specifications, since its crucial role in the development of agent technology.

As regards to FIPA, we can assert that the inter-agent communication is dealt with in several documents and definitely represents an important part of the overall specifications. In our attempt to be concise and rigorous, we can state that FIPA specifications target autonomous agents expected to communicate at a high level of discourse, whose contents are meaningful statements about agents’ knowledge and environment. The FIPA Agent Communication Language is based on the speech act theory; messages are communicative acts that, by virtue of being sent, have effects on the knowledge and environment of the receiver as well as the sender agent. Furthermore, the language is described using formal semantics based on the modal logic. From this it clearly emerges the communication complexity which characterizes multi-agent systems compared to the very simple conversation patterns of the Web services.

For the sake of clarity, it is fair to reiterate that Web services are also suitable for high-level communication patterns and that in the last years several efforts have been carried out in order to provide description languages enabling service orchestration and choreography and moreover to make Web services semantically described, as already mentioned in the “background” section. Nevertheless, the main goal is still to improve interoperability between entities that are not necessarily characterized by sophisticated reasoning capabilities.

To sum up, the two entities, Web services on the one hand and agents on the other hand, and the corresponding communication patterns are quite different from a semantic point of view. That raises problems about the mapping between the two worlds, essentially because differences between the two communication patterns could lead to a loss of descriptive power in the mapping.

Several researchers belonging to the agent community have dealt with the issues concerning the interconnection of agent systems with W3C compliant Web services, with the aim of allowing each technology to discover and invoke instances of the other.

The proposed integration approaches (Greenwood & Calisti, 2004; Nguyen, 2005; Shafiq et al., 2005) denote different shades of meaning of the same idea, i.e. a wrapper or an adapter module playing the role of mediator between the two technologies. Most of them have adopted the gateway approach, providing a translation of WSDL descriptions and UDDI entries to and from FIPA specifications, thereby limiting the communication to simple request-response interactions. One approach (Soto, 2006), which differentiates quite substantially from the others, realizes a FIPA compliant JADE Message Transport System for Web Services enabling agents to interact through the Web with Web services preserving the FIPA compliant communication framework. It only provides a solution for an integration at a low level, leaving a number of issues at higher levels still unresolved.

The most significant of these approaches is WSIGS (Web Services Integration Gateway...
Multi-Agent Systems for Semantic Web Services Composition

Service), a stand alone, encapsulated application that provides transparent, bidirectional transformations between JADE agent services and Web services (JADE Board, 2005). Through the use of the WSIG JADE add-on, JADE agents are able to expose their services, published in the JADE DF, as web services.

The WSIG supports registration and discovery of JADE Agents and agent services by Web service clients, automatic cross-translation of DF directory entries into UDDI directory entries and invocation of JADE Agent services by Web services. Two main processes are continuously active in the WSIG application: (i) the one responsible for intercepting DF registrations/deregistrations, converting them into suitable WSDLs and registering these information with a UDDI registry as tmodels; (ii) the process responsible for serving incoming web service requests, which consists of retrieving the appropriate tModel from the UDDI repository, translating the invocation message into ACL and sending it to the target agent. Any response from the agent will be translated back into SOAP and sent to the requesting Web service.

When an agent needs to invoke a Web service it directly creates the SOAP message and sends it to the provider, e.g. exploited AXIS2 API. In this case no particular support is needed possibly except for a framework that decouples the agent from the API used to invoke Web services.

To date, WSIG supports only simple WSDL description of Web services, without taking into account emerging technologies related to the semantic Web. Another limit of this integration service is that it does not provide any means that agents can use to automatically or even semi-automatically compose Web services.

Agent-Based Workflow Management Systems

Once the infrastructure, enabling a bi-directional connectivity between the two technologies, is in place an agent can play the role of the orchestrator of dynamic Web service compositions. This is also in line with what is reported in the Web Services Architecture specification (WSA, 2004), which introduces a set of concepts and abstractions for Web architectures, “… software agents are the running programs that drive Web services—both to implement them and to access them…”; and furthermore, “A choreography defines the sequence and conditions under which multiple cooperating independent agents exchange messages in order to perform a task to achieve a goal state”.

The current multi-agent solutions, aiming at realizing an effective agent-based service composition, are still in a preliminary phase and certainly need to be improved (Buhler & Vidal, 2005; Savarimuthu et al., 2005). However a lot of researchers and software developers are really interested in giving a significant contribution in this direction.

What clearly emerges is that to be successful, in a first phase it is crucial to appropriately engineer and integrate agent technology with other technologies, in addition to Web services, that have found and will find a purpose in this area, in particular semantic Web. In this scenario agents could represent the “glue” that hold these technologies together, leveraging them and making them perform properly.

Assuming to adopt an agent-based approach, a typical scenario of a service-oriented application would be characterized mainly by three actors: service providers, brokers and users, playing roles which would be allocated to different concrete agents. The system architecture would likely be organized in communities constituted by different kinds of agents: service providers, personal assistants and middle agents (e.g. service brokers, user profile managers, workflow managers, etc). In order to achieve their goals (semantic matching, service contracting and so on) these autonomous agents should be able to perform their tasks in cooperation or competition with other agents and to interoperate with external entities (e.g., legacy
software systems). Moreover they should show reasoning capabilities and should have a support for dynamic behaviour modification based on business rules. Finally they should be able to build workflows, compose the external Web services and monitor their execution. The entire process should be supported by a distributed trust management.

Clearly the researchers are well aware that such a scenario is quite ambitious and the outlined objectives difficult to achieve in a short period. Nevertheless the realizations of prototype systems centred on the underlying infrastructure can be of great help in order to raise awareness of these issues, to delineate possible solutions and therefore to make progress in the development of service-oriented multi-agent systems.

It is in this ambit that our research work (Negri et al., 2006) is situated. During the last years, we have been implementing prototypes of agent based frameworks that cope with the static and dynamic composition of Web services through the use of workflow technologies. The transition from a prototype to another has been due to the evolution of the standards related to semantic Web service composition. What characterizes the prototypes is the architecture, while what differentiates them is the referenced standards for the description of semantic Web services.

The architecture is based on a heterogeneous society of agents, where different members have different internal complexity. In such a heterogeneous society, hierarchical collaboration between reasoning capable agents is achieved mainly through goal delegation. From the point of view of this dissertation, the most interesting types of agents, which compose the society, are: the component manager agent and the workflow manager agent. Each component manager agent is associated to one or more Web services and is responsible for the interaction with them. Workflow managers have the goal of supporting users in the process of building the workflows, composing external Web services and monitoring their execution. The workflow manager agent assumes the role of the delegate agent in a goal delegation protocol (Bergenti et al., 2003), subdivides its goal in sub-goals, generates a utility function from each sub-goal and sets up a negotiation process with the component manager agents.

To accomplish its activity in the most appropriate way the workflow manager provides the users with two alternative automatic procedures:

1. **Predefined workflow**: the workflow is extracted from a repository of standard and common templates, e.g. templates used in previous computations. The workflow manager is responsible for supporting the user in the choice of the most appropriate Web services for the execution of the various workflow tasks. The workflow manager is able to select a matching service thanks to the exploitation of a shared ontology. Even though we were aware that Web services supplied by different providers usually have individual and unique semantics, described by independently developed ontologies, in our attempt towards a semantic support, we considered the simplified, but still significant, case of a shared ontology that gives a common knowledge background to the entities in the system. In order to facilitate the resolution of structural and semantic heterogeneities, semantic Web services have their interfaces semantically described by ontological concepts belonging from this shared ontology.

2. **Dynamic workflow**: the workflow manager, according to the user’s requirements, creates a new workflow, composing the atomic services available in the system. This is done by applying a planner, which we have realized extending the Sensory Graphplan planner (SGP, 2000), that works on the ontological concepts related to the semantic service descriptions provided by component managers. After the composition of the final
The workflow manager is able to update it and possibly replace those Web services that are failed or no more available during the enactment phase.

In addition, the users have been offered the possibility of manually building workflows. In this case, a personal assistant (i.e. an agent, associated with each user active in the system, responsible for the interactions between the user and the other parts of the system) helps its user presenting her/him the tasks (Web services) that can be composed and possibly informing her/him when the realized workflow does not satisfy the composition rules, deduced from the service descriptions. When a complete workflow is realized, the user can ask its personal assistant to delegate its execution to a workflow manager. The enactment is clearly a problematic phase. When a workflow is going to be executed, a Web service could be no more available due to the expiration of a timeout, a failure of a resource or other unpredictable problems. In this case the workflow manager helps the user finding a new solution, creating a new contract phase with all the component managers that are able to satisfy the task and suggesting to the user the replacement of the failed service with the new one.

As far as the semantic aspects are concerned, we have coped with the issue of supporting agents both in the action of looking for a service on the basis of the requirements to be met by the service itself and during the service invocation phase, handling the mapping between the semantic description of the service and its real invocation. In the latter, the agent is therefore asked to provide the semantic input parameters, which for instance refer to ontological concepts, irrespective of the concrete data format required to invoke the real service.

When we started our research work, SAWSDL was not a recommendation yet and OWL-S was the most visible of the several proposals. Our initial choice fell down on OWL-S. We developed a framework enabling agents to invoke Web services on the basis of the ontological concepts belonging to OWL-S documents describing the services and we exploited OWLS-UDDI matchmaker for supporting the discovery phase. Now we are extending the unifying application model in order to include a support for the SAWSDL specification and at the same time we are working on a semantic UDDI registry based on the SAWSDL specification.

In a very early phase workflows were described by using the XPDL workflow language, an XML-based language defined by the Workflow Management Coalition. The manually building workflows were generated using JaWE, a graphical XPDL editor. An XPDL workflow is composed of a set of linked activities. The ‘activity’ element in the XPDL language is the basic building block of a workflow process definition. In our domain each activity corresponded to a Web service and could be linked to other elements of the workflow through a set of ‘transitions’. XPDL supports hierarchical decomposition of activities as the execution of a sub-flow: in this specific case an activity is a link to another XPDL file and workflow. This functionality was used to realize some required loops and to delegate the execution of a specific part of a workflow to another workflow manager agent.

Afterwards, we have moved to WS-BPEL specification. On the one hand WS-BPEL has a better support for Web service, actually its focus is on Web services, but on the other hand it does not support the definition of sub-workflow so far.

In the attempt to give an answer to such problem, we have realized a framework for the distributed execution of a BPEL process. The BPEL process execution is constituted of three phases: (i) interpretation of the BPEL document, (ii) creation of an internal process model, aiming at describing in a consistent way the business process characteristics and at the same time to make easy and efficient the execution of the business process itself, (iii) preparation of the execution context and
distributed execution, possibly providing for the exploitation of new Web services.

The main engine, part of the workflow manager agent, is responsible for initiating and coordinating the entire execution process. It creates the execution context, which represents the reference context during the execution process. Next, it identifies those parts of the workflow (e.g. scope activities, sub-activities of the flow activities, single activities and so on), that if delegated to other agents of the systems (component manager agents but also other workflow manager agents) would positively affect the performance of the system.

Finally, in order to allow agents to be able to produce and consume semantically annotated information and services, it is necessary to provide them with an ontology management support. An interesting approach is characterized by the definition of a meta-model that closely reflects the OWL syntax and semantics. The best effort in this field is represented by the modelling APIs of Jena, an open source semantic Web framework for Java, which is the most famous and widely used tool in the sphere of the semantic web and recently also in the context of multi-agent systems. Initially we used the Jena toolkit to load, maintain and reasoning about OWL DL ontologies, while agents necessitating simple artefacts to access structured information were provided with a light ontology support, called OWLbeans (Tomaiuolo et al., 2006). To date an implementation of a full OWL DL support through a home-made framework supplying ontology management and reasoning functionalities is under development, with the main purpose of reducing the amount of computational resources and time required (compared to the Jena engine).

An interesting real-world agent-based application in the area of business process management is reported in (Greenwood & Rimassa, 2007). The paper describes a system consisting of several integrated components for modeling, executing and administering business processes using a goal-oriented and autonomic approach. A real and current customer case in the domain of Engineering Change Management, from Daimler AG, is used to explore the approach. The authors make a comparison against traditional workflow engines. In particular, as far as the BPEL standard language is concerned, they highlight how it is a rather static language and it does not offer the major advantage of goal-oriented autonomic business process management such as the superior flexibility and dynamic self-management. The infrastructure is not based on a service-oriented architecture but since using a goal-oriented approach separates the statement of what the desired system behavior is, from the possible ways to perform such behavior, in a future it will be possible and probably fundamental to adopt such architecture.

Distributed Collaboration and Coordination

The agents’ ability of operating in dynamic and uncertain environments and the capability of learning from experience have mainly driven the research activities of the agent community in the area of service-oriented computing so far.

Other interesting features, which can be the main ingredients for automatic cooperation between enterprise services, are the agents’ capabilities of collaborating and coordinating themselves. In a business environment, an example would be a broker that has frequently to seek providers as well as buyers dynamically, to collaborate with them and finally to coordinate the interactions with and between them in order to achieve its goals. An intelligent service-oriented infrastructure could do it automatically or semi-automatically, within the defined constraints.

As far as the automatic cooperation between enterprise services is concerned, to the best of the authors’ knowledge, there are very few ongoing studies. The most significant is the one reported in (Paschke et al, 2007). In this work, the authors combine the ideas of multi-agent systems, dis-
tributed rule management systems and service-oriented and event-driven architectures. The work is mainly focused on the design and implementation of a pragmatic layer above the syntactic and semantic layers. Taking advantage of this layer, individual agents can form virtual organizations with common negotiation and coordination patterns. An enterprise service bus is integrated as a communication middleware platform and provides a highly scalable and flexible application messaging framework to communicate synchronously and also asynchronously with external services and internal agents. To date, the authors have developed an interesting methodology and an architectural design but they have only outlined a possible implementation of the system.

FUTURE TRENDS

Multi-agent systems, semantic Web and Web services are still evolving towards a complete maturity. In particular, the evolution and the strengthening of the semantic Web technologies and of the related technologies for providing semantic Web services should influence the evolution of multi-agent technologies given that, on the one hand, the use of agents for the composition of Web services is considered one of the most promising area where applying agents and, on the other hand, one of the requirements for the success of multi-agent systems is that they have to guarantee an easy integration with other widely used industrial technologies as Web services already are and semantic Web ones have a lot of chances to become.

Moreover, multi-agent systems will be able to become the most important mean for Web services composition if the current studies on coordination, knowledge management, distributed planning and learning will have as one of their results the definition of a set of techniques able to simplify and at the same time to enhance the realization of systems where parties have multiple, perhaps competing, objectives and services have to be provided satisfying the required quality of service and coping with the possible failures due to either the unavailability or the malfunctioning of some components of the system.

CONCLUSION

In this chapter we have tried to show how the agent technology together with Web service and semantic Web technologies allow the conception and realization of an advanced solution to Web service composition, paving the way for real flexible, adaptive intelligent service-oriented systems.

The research in distributed artificial intelligence has been addressing for several years the problem of designing and building coordinated and collaborative intelligent multi-agent systems. This interesting and advanced work can be fruitfully exploited in the area of service-oriented computing if agent technology is appropriately engineered and integrated with the other key technologies. To support this claim, we have addressed the benefits in applying multi-agent systems and we have shortly introduced the solutions that multi-agent systems can provide trying to show how the powerful synergism between these technologies could be very promising.

REFERENCES


Multi-Agent Systems for Semantic Web Services Composition


**KEY TERMS AND DEFINITIONS**

**Autonomic Computing:** It is an initiative started by IBM in 2001 and it is about an approach for realizing self-managing systems, i.e. systems characterized by self-configuration, self-healing, self-optimization and self-protection properties.

**Coordination:** Coordination is a process in which a group of agents engages in order to ensure that each of them acts in a coherent manner.

**Contracting:** A process where agents can assume the role of manager and contractor and where managers tries to assign tasks to the most appropriate contractors.

**Goal Delegation Protocol:** An interaction protocol allowing an agent to delegate a goal to another agent in the form of a proposition that the delegating agent intends its delegate to bring about.

**Multi-Agent Planning:** A process that can involve agents plan for a common goal, agents coordinating the plan of others, or agents refining their own plans while negotiating over tasks or resources.

**Multi-Agent System:** A multi-agent system (MAS) is a loosely coupled network of software agents that interact to solve problems that are beyond the individual capacities or knowledge of each software agent.

**Negotiation:** A process by which a group of agents come to a mutually acceptable agreement on some matter.

**Organizational Structuring:** A process for defining the organizational structure of a multi-agent system, i.e., the information, communication, and control relationships among the agents of the system.

**Software Agent:** A software agent is a computer program that is situated in some environment and capable of autonomous action in order to meet its design objectives.