An Agent-Based Service Oriented Architecture

Agostino Poggi, Michele Tomaiuolo, Paola Turci

Abstract — **Industry is more and more interested in executing business functions that span multiple applications. This demands high-levels of interoperability and a more flexible and adaptive business process management. The trend is to have systems assembled from a loosely coupled collection of Web services, which are ubiquitous and organically integrated. This technical area appears to be a natural environment in which the agent technology can be exploited with significant advantages.**

In the present paper, we propose a framework with the aim of supporting an agent-based SOA. The peculiar characteristic and strength of our research work is the integration of the agent technology with other strategic technologies, that is Web services, workflow, rule engine and semantic Web.

Index Terms — **Multi-agent systems, service oriented architecture, workflow, ontology, rule engine, trust management.**

I. Introduction

Most of the technology and market research companies, which provides their clients with advice about technology's 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.

Agent technology is more and more considered one of the most interesting technologies to successfully support SOA. In fact, besides being an ideal mechanism for implementing complex systems, agent technology is well-suited to applications that are communication-centric, based on distributed computational and information systems, and requiring autonomous components readily adaptable to changes.

Considering their peculiar features, the central role that agents should play in a SOA scenario is to efficiently support distributed computing and to allow the dynamically composition of Web services. To be successful, it is crucial to appropriately engineer and integrate agent technology with other technologies that have found and will find a purpose within enterprise computing: workflows, rule engines, the semantic Web and Web services.

The vision which is making its way into the research community is to encapsulate the organization's 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 of automatically discovering and composing Web services. An answer to these problems could come from the semantic Web technology.

Recently, we have seen an explosion of interest in ontologies as artefacts to represent human knowledge and as a critical component in several applications; among these the e-business applications. Moreover the "marriage" between agents and ontologies seems to be the kind of technology that can significantly change the face of enterprise software.

On the one hand, ontologies should accomplish the task of giving a precious support to solve two tricky problems: how to efficiently discover Web services and how to make possible the interoperability of heterogeneous Web services. In order to facilitate the resolution of such a structural and semantic heterogeneity, Web services, which play the role of workflow components, will have their interfaces semantically described by ontological concepts.

On the other hand, ontologies enable agents to communicate in a semantic way, exchanging messages which convey information according to explicit domain ontologies.

In this scenario agents represent the backbone of the system and the "glue" that could hold these pieces together and make them perform properly.

Assuming to adopt an agent-based approach, a typical agent-based SOA scenario would be characterized mainly by three actors: service providers, business process manager 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.

A. Poggi is with DII, University of Parma, Parco Area delle Scienze 181A, 43100, Parma, Italy (phone: +39 0521 905728; e-mail: poggi@ce.unipr.it).

M. Tomaiuolo is with DII, University of Parma, Parco Area delle Scienze 181A, 43100, Parma, Italy (phone: +39 0521 905712; e-mail: tomamic@ce.unipr.it)

P. Turci is with DII, University of Parma, Parco Area delle Scienze 181A, 43100, Parma, Italy (phone: +39 0521 905708; e-mail: turci@ce.unipr.it).

Clearly the researchers are well aware that such a scenario is quite ambitious and the outlined objectives difficult to achieve in a short period. Indeed there are several overlooked technical issues and the existing technology presents significant limitations. Nevertheless the realizations of prototype systems centred on the underlying infrastructure can be of great help in order to raise awareness of these issues and to delineate possible solutions.

Bearing in mind what said above, the aim of the present paper, which is an evolution of our previous paper [23], is to introduce a framework, under development at the University of Parma, for the realization of an agent-based SOA.

In the next section we discuss the related work in the fields of the emergent and more established technologies which we aim at integrating with agent technology. Section 3 describes the framework aiming at being the basis for the realization of successful and innovative agent-based SOA. In this section we focus mainly on its architecture, the BPEL engine, the ontological support, the integration with a rule engine and our proposal for a distributed trust management. The paper ends by drawing some conclusions around the results of the work done, and by outlining some considerations regarding our future research directions.

II. Related Work

There is evidence from several research studies [1],[24] that agents represent one of the most suitable technologies which can be used to meet the performance needs for innovative business applications. In particular the current interest in using agents for developing e-business applications, business process management and enterprise integration is rising mostly because different works have shown how agent technology can be leveraged if used together with technologies exploited in the Internet, that is, semantic Web, Web services and workflows [5],[7],[12],[19],[22].

Semantic Web technologies appear to be the right means to provide the semantic integration between data and processes across systems that can be owned by different enterprises [6]. This technology is not completely mature yet; some major activities related to the definition of languages for expressing the semantics of the Web are still in progress [16],[8]. Nevertheless different works have shown how the powerful synergism between agents and semantic Web could be very promising [19],[29] and some efforts have been made in order to define ontology models and develop tools suitable for agents aiming at being truly semantic aware agents. The research community contributions have been mainly devoted to cope with three different issues:

- The formal definition of a standard language for expressing semantics on the web which has led to the Web Ontology Language,
- The development of integral software infrastructures, for writing semantic web applications, offering a variety of tools to engineer ontologies.
- The development of ontological supports specifically thought for multiagent systems.

The second and third points are strictly connected to the first one since OWL is considered the reference language; therefore the work carried out, starting from OWL, has developed tools more suitable for different contexts.

As far as the second point is concerned, an interesting approach is characterized by the definition of a meta-model that closely reflects the OWL syntax and semantics. This is the case of the modelling APIs of Jena, which is the most famous and widely used tool in the sphere of the semantic web (and recently also in the context of multiagent systems).

Considering the third point, the focus is on the specific needs of multiagent systems, and the objective is to provide a communication support enabling agent to perform the proper semantic checks on a given content expression. A significant example of the efforts made in this direction is represented by the ontological support of JADE, designed to represent, using Java objects, a taxonomy of concepts. Such semantically aware agents should then be able to discover, invoke, compose and monitor those Web resources that provide services. In order to make agents able to use a service, they need a computer interpretable description of the service itself and furthermore to know the means by which it is accessible. To that purpose, a community of researchers is developing an ontology of services, called OWL-S, with the aim of providing a semantic orientation to the description of Web services.

To enable software systems for innovative business applications, security issues have to be carefully analysed and sound solutions have to be deployed. A number of different solutions for the problems of authentication and authorization in open systems have been proposed in the scientific literature, and some standards have emerged through the years. Most of them are based on some kind of PKI and signed certificates issued by a Certification Authority. In particular, this is the case of X.509, which is the best known and adopted standard for authentication and authorization. However, its weaknesses have been clearly demonstrated in a number of works [15], above all related to its effort to create a global directory of unique names. Relying on an external entity as root of all certifications represents an additional, not directly controllable, point of failure for the whole system.

In contrast, different approaches have been proposed, based on local names. Both SDSI and PetName Markup Language allow local names to be used in a global scale by prefixing them with the public key of the principal defining them, in the form of (key, name) couples. This way, name conflicts are solved thanks to the uniqueness of the public keys. In [32], authors show that local names and YURLs are more robust than global names to phishing attacks, arguing the root for these attacks lie in the global namespace itself. Moreover, in [21], authors shows that local names and a subset of the SDSI/SPKI standard [9] can be used to implement a distributed RBAC infrastructure, in which local names are interpreted as distributed roles, whose name is localized to their defining principal (key).

Local names and delegation certificates are the key to build systems adhering *trust management principles* [18]. These systems are completely distributed as they avoid any centralized authority. This way they can easily scale to large peer-to-peer networks, where each node is in charge of protecting its own resources and to show proper credentials when accessing resources of other nodes.

In recent years a lot of research work has been undertaken ranging from the use of workflows in distributed systems to the use of agent technology for the management of workflows [7][24][19]. The most important contribution of our work is firstly the use of agents as a support of all the activities involved in the development and execution of a business process, i.e. the workflow generation, the distribution of workflow tasks, the control of their execution and finally the re-allocation of tasks in case of failure of some service components. Secondly, the integration of the agent technology with those technologies we consider crucial for accomplishing strategic business objectives.

To conclude just a few remarks on JADE since it is considered the reference implementation of the FIPA specifications and one of the most used and promising agent development framework. The present release of JADE tries to provide agent developers with a support integrating almost all these technologies, even if in our opinion only partially. As a matter of fact, JADE agents can exploit an ontological model of the application domain to improve their interactions, are able to interact with external Web services [12] and finally different works have shown how the integration of a JADE agent with the Jess rule engine is feasible. But this simply represents a first step towards an effective support of the SOA paradigm.

III. Towards a service oriented architecture

To overcome the limits of the present release of JADE, we have realized an agent based framework called MASE (Multi-Agent Service Environment), which is the evolution of our previous framework GAIN [22], that allows dynamically composing Web services. Its architecture is based on a society of agents, mostly composed of two kinds of agents: component managers and workflow managers.

Each component manager is associated to one or more Web services and is responsible for the interaction with them. Through the use of the WSIG JADE add-on [12], the component managers are able to invoke a Web service, converting ACL messages into WSDL descriptions and vice versa. Moreover, a component manager allows a flexible provision of services defining "on the fly" the features of the services (price, timing, etc.) through a set of business rules managed by a rule engine and modifiable by the operators of the service provider through a Web interface.

Workflow managers have the goal of supporting users in the process of building the workflows, composing external Web services and monitoring their execution. To accomplish this complex activity the workflow managers provide the users with two alternative automatic procedures:

Predefined workflow; the workflow is extracted from a repository of standard and common templates, e.g. templates used in previous computations. In this case the duty of the workflow manager is to support the user in the selection of the most appropriate Web services for the execution of the different workflow tasks. The workflow manager is able to select a matching service thanks to the exploitation of a shared ontology that gives a common knowledge background to all the agents in the system.

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 (we have realized it by extending the SGP planner [28]) that works on the operators extracted from the OWL-S descriptions of the Web services, provided by component managers. After the composition of the final workflow, the workflow manager is able to update it and possibly replace those Web services that are failed or no more available or cannot satisfy the execution time constraints.

Moreover, MASE offers to the users 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 interaction 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, coming from the related OWL-S descriptions. When a complete workflow is realized, the user can ask its personal assistant to delegate the workflow 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.

So far we have given a concise description of the system architecture and the responsibilities of the major system components, intentionally leaving out the treatment of the issues connected to the tools needed by the agents, in order to carry out their activities. In the following subsections, we will go into details, illustrating our proposals and the implemented tools.

A. The BPEL Engine

The WS-BPEL specification defines an XML-based language for the formal description of a business process based on Web services orchestration. It is an open standard recently approved as an OASIS standard.

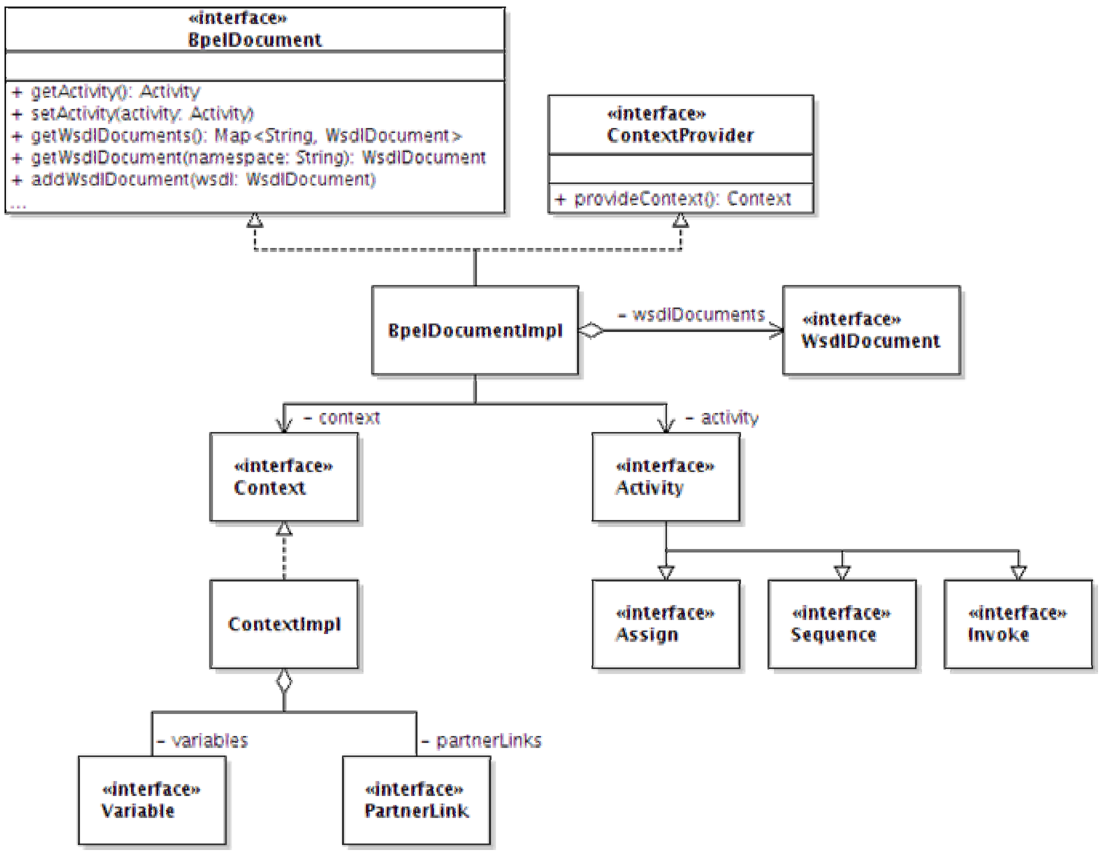

Figure 1 - Excerpt of the internal model representing the BpelDocumentImpl class and its relationship with the correlated classes

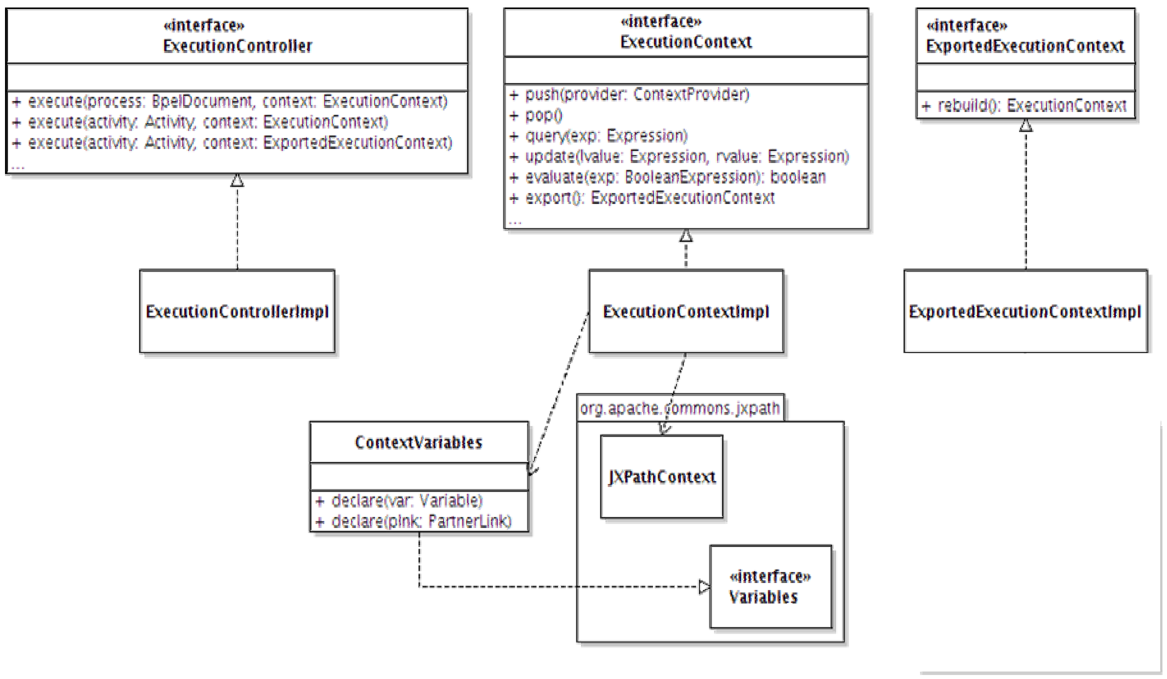
«interface» ExecutionController
+ execute(process: BpelDocument, context: ExecutionContext)
+ execute(activity: Activity, context: ExecutionContext)
+ execute(activity: Activity, context: ExportedExecutionContext)
...
«interface» ExecutionContext
+ push(provider: ContextProvider)
+ pop()
+ query(exp: Expression)
+ update(lvalue: Expression, rvalue: Expression)
+ evaluate(exp: BooleanExpression): boolean
+ export(): ExportedExecutionContext
...
«interface» ExportedExecutionContext
+ rebuild(): ExecutionContext
ExecutionControllerImpl
ExecutionContextImpl
ExportedExecutionContextImpl
org.apache.commons.jxpath
JXPathContext
ContextVariables
+ declare(var: Variable)
+ declare(plnk: PartnerLink)
«interface» Variables

Figure 2 - Excerpt of the internal model representing the ControlFlowActivity class and its relationship with the correlated classes

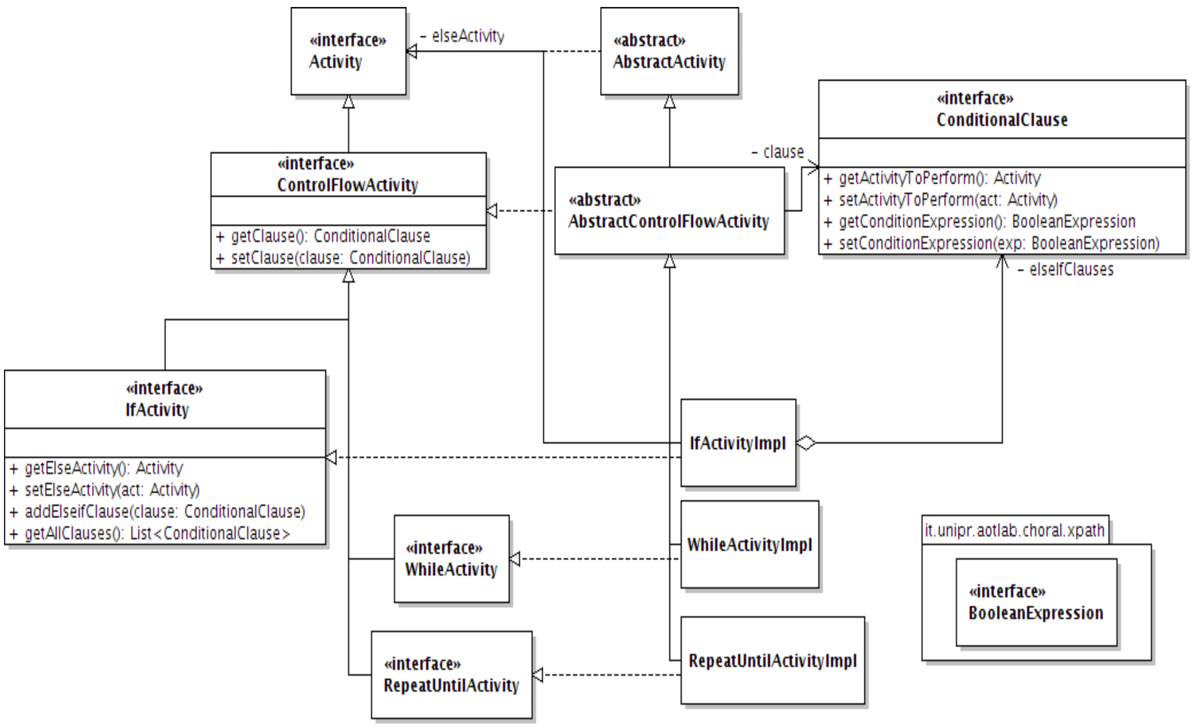
«interface» Activity
- elseActivity
«abstract» AbstractActivity
«interface» ConditionalClause
+ getActivityToPerform(): Activity
+ setActivityToPerform(act: Activity)
+ getConditionExpression(): BooleanExpression
+ setConditionExpression(exp: BooleanExpression)
- clause
- elseIfClauses
«interface» ControlFlowActivity
+ getClause(): ConditionalClause
+ setClause(clause: ConditionalClause)
«abstract» AbstractControlFlowActivity
«interface» IfActivity
+ getElseActivity(): Activity
+ setElseActivity(act: Activity)
+ addElseifClause(clause: ConditionalClause)
+ getAllClauses(): List<ConditionalClause>
IfActivityImpl
WhileActivityImpl
«interface» WhileActivity
«interface» RepeatUntilActivity
RepeatUntilActivityImpl
it.unipr.aotlab.choral.xpath
«interface» BooleanExpression

Figure 3 - The major classes of the BPEL engine

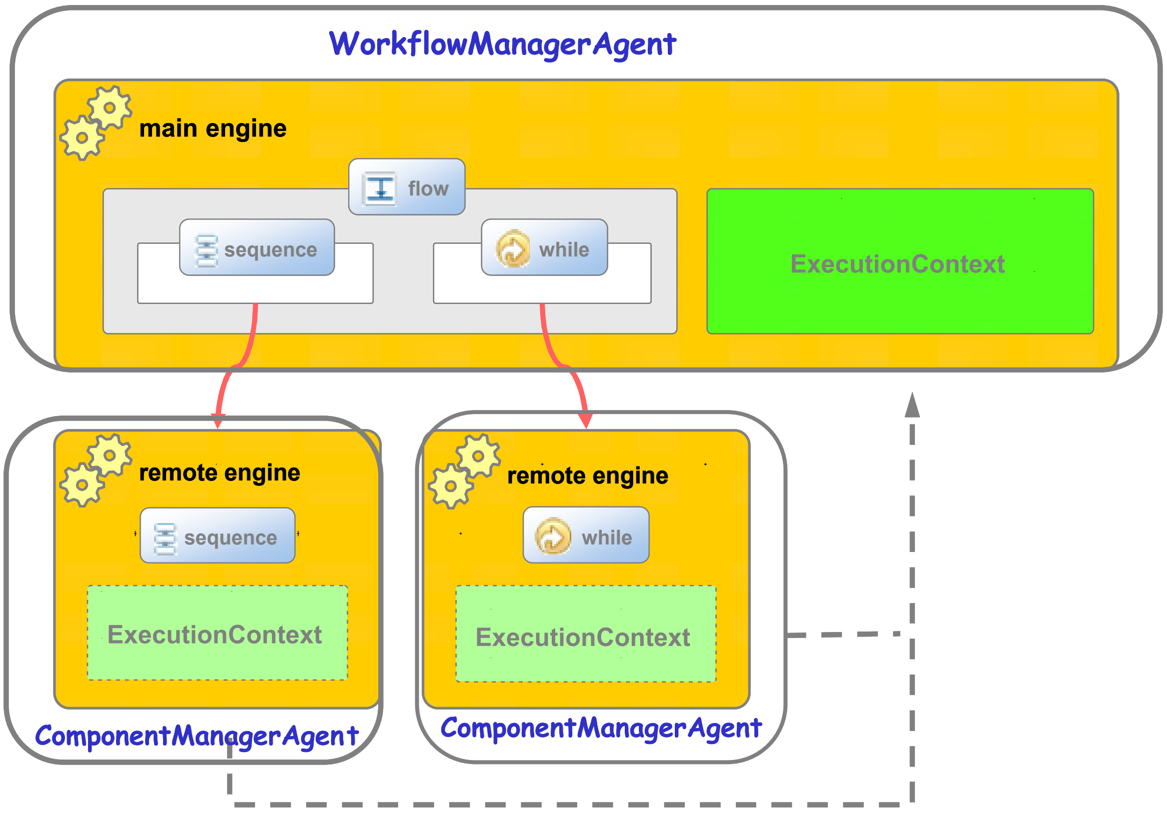

Figure 4 - An example of distributed execution

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. The main drawbacks of these tools are that they enact the workflow in a centralized manner and furthermore they are not able to dynamically exploit new Web services in case of unpredictable event.

In the attempt to give an answer to such problems, we have realized a framework for the distributed execution of a BPEL process and the dynamically composition of Web services. 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.

In the first phase of the execution process we have utilized XMLBeans (a framework, part of the Apache XML project.), which has the advantage of fully supporting XML Schema and XML Infoset. As far as a BPEL process is concerned there are two schemas that have to be provided to XMLBeans in order to parser the BPEL document: WS-BPEL schema and WSDL schema (we have referred to WS-BPEL 2.0 and WSDL 1.1).

Regarding the second phase, in order to give an idea of how the internal model is structured, in Figure 1 and Figure 2 two excerpts of the model are shown. In particular, they represent respectively the model core class, i.e. the BPELDocumentImpl class, and a representative BPEL structured activity, i.e. the AbstractControlFlowActivity class, together with their major correlated classes.

The engine is responsible for efficiently executing a BPEL process, instance of the model. The classes, playing a key role in its implementation, are shown in Figure 3.

The main engine, part of the workflow manager agent, is responsible for initiating and coordinating the entire execution process. It creates the execution context, an instance of the ExecutionContextImpl class, which will represent the reference context during the execution process. Next, it will identify those parts of the workflow (e.g. scope activities, sub-activities of the flow activities, and so on), that if executed remotely will positively affect the performance of the system, and will delegate their execution to specific component manager agents. Figure 4 shows an example of distributed execution.

From what said above, it emerges clearly that we have chosen to have stateless engines and thus to share the execution context. As a consequence the remote engines have to send messages to the main engine in order to update consistently the reference execution context. We have found out that this choice has several advantages, primarily it makes easier to handle the possibly dependency between activities.

B. Ontology Support

The idea which mostly inspired the design of the JADE content language and ontological support was to define an ontology independent abstract model of the content language that could be subsequently bound to any domain ontology representation expressed using an object-oriented data model. This ontological support has been conceived when the Semantic Web was on its very early stage of research and development and OWL was not already established as a standard. Consequently its expressive power is clearly limited with respect to OWL and basically allows expressing taxonomy of concepts, predicate and actions and therefore it is not able to represent completely the different application domains where JADE agent may be used.

In order to provide a JADE agent with an adequate expressive power (i.e., equivalent to the one offered by OWL DL), it is necessary either to replace or to integrate the JADE ontological support. In the attempt to find a suitable solution to this problem one has to choose among the proposals described above and others, each characterized by different domain knowledge modelling techniques and answering different needs. The majority of the research work in this field is thought for the semantic Web. But while in the vision of the semantic Web the increasing interest in ontologies is driven by the large volumes of information available and by the need of automating many information retrieval activities, in the agent context the focal point is slightly different and it is mainly on communicative acts - communications which implies actions. Agents would use ontologies to perform the proper semantic checks on a given content expression, and therefore ontologies should include concepts (objects of the domain of the discourse) but also predicates (assertions on properties of concepts) and actions (that agents can perform in the domain). Moreover a peculiar characteristic of the agent community is the heterogeneity of resources available and the roles played by different agents of a system. This leads us to choose different approaches in different contexts. Our solution was to realize a compound tool, called OWLBeans [5], that allows the use of ontologies described by using OWL DL. These ontologies can be used by agents for performing their tasks in cooperation with other agents, for interoperating with external entities (e.g., legacy software systems) and for performing a semantic matching of Web services, described by using OWL-S and having inputs and outputs associated with concepts belonging to a domain ontology.

OWLBeans is based on a two-level approach with the aim of coping with both the issues of managing complex ontologies and of providing ontology management support to lightweight agents, which seldom need to deal with the whole complexity of a OWL DL ontology. Therefore, lightweight agents maintain the simple JADE ontology support whereas one or more dedicated agents, acting as ontology servers, are able to use and manage complete OWL DL ontologies and provide the service to the agents that need it.

The main functionality of OWLBeans is to extract JADE ontologies from OWL DL ontologies realizing a set of ontologies usable by JADE agents, with the obvious shortcoming that not all the information maintained in the original OWL ontologies are taken into account. Therefore, for all those systems that need a complete support for OWL DL ontologies, OWLBeans offers a set of ontology server agents implemented as JADE agents, providing a common knowledge base and reasoning facilities. These ontology servers use the Jena toolkit to load, maintain and reasoning about OWL ontologies. The other agents of the system do not need to know anything about the Jena toolkit given that these ontology servers provide them with a set of simple actions for querying and manipulating the ontologies. Furthermore, ontology servers take into account proper authorization mechanisms. In particular, the underlying trust management support (discussed in the following subsection) has been leveraged to implement a certificate-based access control. Only authenticated and authorized principals will be granted access to managed ontologies. A delegation mechanism allows the creation of communities of trusted entities, which can share a common ontology, centrally managed by the ontology server.

Finally, despite the fact that the JADE ontological support is quite simple, it could still be complex for some devices with limited resources such as smart phones. This is the reason why we have decided to improve OWLBeans adding a further feature which allows agents to import taxonomies and classifications from OWL ontologies, in the form of a hierarchy of Java classes with the purpose of providing very simple artefacts to access structured information. Given its modular architecture, based on an intermediate ontology model, OWLBeans also provides further functionalities, e.g., saving a JADE ontology into an OWL file, or generating a package of JavaBeans from the description provided by a JADE ontology.

C. *Production Rule Management*

JADE provides the integration with the JESS rule engine, which is probably the most known Java rule engine implementing the Rete algorithm [11]. This integration is realized through a so-called JessBehaviour that allows the encapsulation of a JESS rule engine inside a JADE agent and has the duty of storing and retrieving information in/from the rule engine. The main limits of this solution are: i) the rule engine is completely hidden to the other agents of the system and there is not any support for the cooperation among different rule-based agents (i.e., agents encapsulating a JESS rule engine) and ii) JESS is a commercial software and so we have additional costs if we plan to realize commercial applications by using JADE together with JESS.

In order to cope with the limits of the current JADE support for rule engines, we realized a software library, called D4J (Drools4JADE) [4], that integrates JADE agents with the Drools rule engine. Drools is a well known, freeware implementation of the so-called Rete-OO algorithm. Apart of its open-source availability, one of the main advantages of Drools is exactly the fact that it is not just a literal implementation of the Rete algorithm, but rather an adaptation for the object-oriented world. This greatly eases the burden of integrating the rule engine and the application rules with the existing external objects. In Drools, asserted facts are simple Java objects, that can be modified through their public methods and properties. Where Jess requires hundreds of lines of code, for example to simply access an ACL message mapped into a Java object, Drools rules can obtain the same result in a dozen of easy-reading code lines.

D4J guarantees both the advantages of full rule-based agents, i.e., agent whose behaviour and/or knowledge is expressed by means of rules [17], and the advantages of rule-enhanced agents, i.e., agents whose behaviour is not normally expressed by means of rules, but that use a rule engine as additional component to perform specific reasoning, learning or knowledge acquisition tasks [14]. In facts, in D4J, the Drools rule-engine is integrated into an agent as a JADE behaviour, but it also provides an API for interacting with it through ACL messages allowing both remote storing and retrieval of knowledge and the cooperation among different rule-based agents. Moreover, this API allows rules mobility, i.e., a rule-based agent can move a rule to another rule-based agent.

Given their nature, business rules often refer to domain specific concepts and, especially when dealing with data on the semantic Web, these concepts are part of a domain ontology. To better support this scenario, rule-enhanced JADE agents should be augmented with a tool for the automatic transformation of concepts, relations and individuals of an OWL ontology to java classes, properties, and instances. The D4J framework can be integrated with OWLBeans, which enables the extraction of JavaBeans from an OWL ontology. The JavaBeans can then be directly asserted as facts into the working memory of Drools.

D. *Distributed Trust Management*

Out of the box e-business applications are not certainly possible if security problems are not analyzed and addressed. Our framework supports the implementation and deployment of secure systems, adhering trust management principles. For this purpose, local names, interpreted as distributed roles, and delegation certificates are made available, to build peer-to-peer networks of trusted entities.

The accurate release of authorizations is often the most critical point of security systems. Ideally, systems should respect the principle of *least privilege*, but this often contrasts with other requirements, as easiness of understanding, scalability and manageability. In this respect, the RBAC model has proven to be a good abstraction to manage large and complex systems, up to corporate and virtual-organizations environments.

Following the RBAC model, each resource manager of our system (i.e. each node in the peer-to-peer network) has to deal with three main concepts: principals (i.e. authenticable entities which act as users of resources and services), permissions (i.e. rights to access resources or use services) and roles.

A many to many relationship binds principals and the roles they are assigned to. In the same way, a many to many relationship binds permissions and the roles they are granted to, thus creating a level of indirection between a principal and his access rights. This also leads to a better separation of duties (between the assignment of principals to roles and the definition of role permissions), to implement privilege inheritance schemes among superior and subordinate roles and to permit temporary delegations of some of the assigned roles towards other principals. The fundamental principle here is that each node is in charge of defining its own roles, and of assigning principals to them.

Dynamic delegation of access rights is made possible through the use of delegation certificates, whose structure is based on the theory of [9]. But we have avoided s-expressions, preferring XML to them, as it provides a better ground to exploit and integrate standard technologies. In particular, in recent times SAML [27] have emerged as a language to express properties of authenticable principals, encoded in the generic form of signed security assertions. SAML assertions can easily bind public keys to local names, and certify the links between two different local namespaces, as it happens in SDSI/SPKI certificates.

Delegation is particularly important to deal with the activation of intermediate agents, acting between the human user and the concrete service providers, i.e. personal agents, workflow managers and agents providing composite services. In this case, privileges must be forwarded in the form of delegation certificates from the user toward each agent in the chain, up to the final service provider, which will check them for consistency with local security policies. These policies are stored and managed locally as XACML documents [31].

To cope with the security problems coming from the remote utilization of the rule engine and from the mobility of rules, also D4J exploits this security layer to enforce security policies at two different levels: proper authorization is necessary to modify the working memory and the rule set of an agent; moreover, each rule is associated with a specific protection domain, limiting the resources made accessible when it is scheduled for execution.

IV. Conclusion

In this paper, we have presented an agent-based framework for SOA that integrates agent technology with other technologies that have found, and will find, a purpose within enterprise computing: web services, workflows, ontologies and rule engines.

Up to now, we have not experimented the system with real users and real services, but we have tested and evaluated the system functionalities implementing some "artificial" services and involving a group of students, acting either as service provider operators or as customers. Some of the information used by the service providers, implemented just for the experimentation of the system, comes from the Web site of some real service providers (e.g., flight companies, hotel brokers, etc.). The results, though still preliminary, are quite encouraging

We are well aware that the current multi-agent solutions need to be improved since the technologies used are still not completely mature. However a lot of researchers and software developers are really interested in giving a significant contribution in this direction, driven by the motivation of providing a strengthening of the related standards and new methodologies, algorithms and implementations to realize real flexible, adaptive SOA [1],[10].

Our future activities will be oriented towards the aforementioned goal. In particular, we will continue working on the JADE software environment in order to both improve the integration of the JADE agents with the most interesting knowledge and internet-oriented technologies and realize real adaptive agents that will be the basis of next and future business applications. At present, we are working in three main directions: i) to finalize the implementation of a full OWL DL support through a home-made framework supplying ontology management and reasoning functionalities, with the main purpose of reducing the amount of computational resources and time required (compared to the Jena engine), ii) to enhance the distribution algorithm in order to achieve a more efficient execution of a workflow and iii) to finalize the implementation of a framework for the dynamic composition of semantic Web services.

References

[1] AgentLink III. Agent Technology Roadmap. Available from http://www.agentlink.org/roadmap/index.html

[2] Akkermans, H. Intelligent E-Business - From Technology to Value. IEEE Intelligent Systems, 16(4):8-10, 2001.

[3] Bechhofer, R. Volz, and P. Lord. Cooking the semantic web with the OWL API. In Proc. Int Semantic Web Conference, pp. 659-675, Sanibel Island, FL, 2003.

[4] Beneventi, A., Poggi, A., Tomaiuolo, M., & Turci, P. Integrating Rule and Agent-Based Programming to Realize Complex Systems. WSEAS Trans. on Information Science and Applications, 1(1):422-427, 2004.

[5] Bergenti, F., Poggi, A., Tomaiuolo, M., Turci, P. An Ontology Support for Semantic Aware Agents. In Proc. Seventh International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2005 @ AAMAS), Utrecht, The Netherlands, 2005.

[6] Berners-Lee, T., Hendler, J., Lassila O. The Semantic Web - A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. 284(5):34-43, 2001.

[7] Buhler P.A., Vidal, J.M. Towards Adaptive Workflow Enactment Using Multiagent Systems. Information Technology and Management, 6(1):61-87, 2005.

[8] de Bruijn, J., Polleres, A., Lara, R., Fensel, D. OWL DL vs. OWL Flight: Conceptual Modeling and Reasoning for the Semantic Web. In Proc. of the 14th Int. World Wide Web Conference (WWW2005), pp. 623-632, Chiba, Japan, 2005.

[9] Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., Ylonen, T. SPKI Certificate Theory. RFC 2693, 1999.

[10] Fensel, D., Bussler, C. The Web Service Modeling Framework WSMF. Electronic Commerce Research and Applications 1(2): 113-137, 2002.

[11] Forgy, C. L. (1982). Rete: A Fast Algorithm for the Many Pattern / Many Object Pattern Match Problem, Artificial Intelligence 19(1), pp. 17-37.

[12] Greenwood, D., Callisti, M. Engineering Web Service-Agent Integration. In IEEE Conference of Systems, Man and Cybernetics, 2004. Available from http://www.whitestein.com/resources/papers/ieeesmc04.pdf.

[13] Gibbins, N., Harris, S., Shadbolt, N. Agent-based semantic web services. In Proc of the 12th International World Wide Web Conference (WWW2003), Budapest, Hungary, 2003.

[14] Gutknecht, O., Ferber, J., Michel, F. Integrating tools and infrastructures for generic multi-agent systems. In Proc. of the 5th International Conference on Autonomous Agents. Montreal, Canada, 2001.

[15] Gutmann, P. (2000). X.509 Style Guide. Available from http://www.cs.auckland.ac.nz/~pgut001/pubs/x509guide.txt

[16] Horrocks, I. and Patel-Schneider, P. F. A proposal for an OWL rules language. In Proc. of the Thirteenth International World Wide Web Conference (WWW 2004), pp. 723-731, 2004.

[17] Hindriks, K.V., de Boer, F.S., van der Hoek, & W., Meyer, J.C. Control Structures of Rule-Based Agent Languages. In Lecture Notes In Computer Science, Proceedings of the 5th International Workshop on Intelligent Agents V, Agent Theories, Architectures, and Languages, Vol 1555, pp. 381-396, 1998, London, UK: Springer-Verlag.

[18] Khare, R., Rifkin, A. Weaving a Web of trust, World Wide Web Journal Special Issue on Security, 2(3):77–112, 1997.

[19] R. Kishore, H. Zhang & R. Ramesh, Enterprise integration using the agent paradigm : foundations of multi-agent-based integrative business information systems, Decision Support Systems, 42 (2006) (1), pp. 48–78..

[20] Labrou, Y. Agents and ontologies for e-business. Knowledge Engineering Review, 17(1):81-85, 2002.

[21] Li, N., Mitchell, J.M.. RT: A Role-based Trust-management Framework. In Proc of the Third DARPA Information Survivability Conference and Exposition (DISCEX III), pp. 201-212, 2003. Washington, D.C.

[22] Negri, A., Poggi, A., Tomaiuolo, M., Turci, P,. Dynamic Grid Tasks Composition and Distribution through Agents,. Concurrency and Computation: Practice and Experience, 2005.

[23] Negri A., A. Poggi, M. Tomaiuolo, P. Turci, Agents for e-Business Applications, In Proc. AAMAS 2006, Hakodate, Japan 2006

[24] Papazoglou, M.P.. Agent-oriented technology in support of e-business. Communication of ACM, 44(4):71-77, 2001.

[25] Papazoglou, M.P. The World of e-Business: Web-Services, Workflows, and Business Transactions In Lecture Notes In Computer Science, CAiSE '02/ WES '02: Revised Papers from the International Workshop on Web Services, E-Business, and the Semantic Web, Vol 2512, pp. 153-173, 2002. London, UK. Springer-Verlag

[26] Poggi, A., Rimassa, G., Tomaiuolo, M. Multi-user and security support for multi-agent systems. In Proc. of WOA 2001, pp. 13-18, 2001. Modena, Italy: Pitagora.

[27] SAML - Security Assertion Markup Language. Available from http://xml.coverpages.org/saml.html.

[28] Sensory Graph Planner software and documentation. Available from http://www.cs.washington.edu/ai/sgp.html.

[29] Silva, N., Rocha, J., Cardoso, J. E-Business Interoperability Through Ontology Semantic Mapping. In Proc. of the Processes and Foundations for Virtual Organizations, pp. 315-322, 2003. Lugano, Switzerland.

[30] Weikum G. Special Issue on Infrastructure for Advanced E-services, IEEE Data Engineering, 24(1), 2001.

[31] XACML - Extensible Access Control Markup Language. Available from http://xml.coverpages.org/xacml.html.

[32] YURL - Decentralized Identification. Available from http://www.waterken.com/dev/YURL.