SERVICE-ORIENTED INTEGRATION OF COMPONENT AND AGENT MODELS

Nour Alhouda Aboud, Eric Cariou, Eric Gouardères and Philippe Aniorté
LIUPPA / Université de Pau et des Pays de l'Adour, France
{Nour-alhouda.Aboud, Eric.Cariou, Eric.Gouarderes}@univ-pau.fr,
Philippe.Aniorte@iutbayonne.univ-pau.fr

Keywords: Components, agents, services, specification process.

Abstract: Multiagent systems and component-based systems are two mature approaches; each one owns strengths and weaknesses points. Our goal is to integrate these two approaches by reaching a high level of connectivity between them to overcome their shortages. The concept of service plays a key role in their interoperability. Indeed, the relations between these three domains are manifold. From a service perspective, agents and components are considered as service providers or consumers while services can be seen as their functional abstractions. Therefore, the concept of service as the interaction point between agents and component is the base of our integration approach. We will define a specification process composed of several models. They are dedicated to specify an application through several aspects: abstract services, components, agents and mix of them. In this paper, we present a global view of this process and three of its models: service, agent and component ones.

1 INTRODUCTION

Nowadays, information systems are distributed, large-scaled, heterogeneous, open and complex. This leads to the emergence of more high-level technologies that interoperate between each other and break the software's isolation. We can cite multi agent systems in artificial intelligence domain and component-based approaches and service-oriented architecture in software engineering domain. Each approach owns strengths and weaknesses points. Our goal is to integrate these approaches by reaching a high level of connectivity between them to overcome their shortages.

Service approaches view applications as sets of services that interact between each other according to their played roles and independently of their locations, in order to satisfy heterogeneous and loose-coupled software systems. These systems can be built with any technology, for instance, component or agent ones.

Component approaches are based on the main interest of reusing blocks of code that implement well-specified interfaces (black boxes with access points). This approach provides efficient solutions for defining well-structured and robust applications by composing and reusing existing components (following for instance the Commercial Off the Shelf (COTS) approach). The interfaces of a component can be considered as the definition of its services, while service approaches can be viewed as logical extensions of component approaches as both of them meet reusability and composition purposes.

MultiAgent System (MAS) is a paradigm for understanding and building distributed systems, where it is assumed that the computational elements (the agents) are able to perform autonomous actions in some environment. They are characterized by the social ability to cooperate, coordinate, and negotiate with each other (Wooldridge, 2009). There are two main types of agents: reactive and proactive ones. A reactive agent waits until being asked or responds to changes in its environment, while a proactive agent takes the initiative in decision making and information gathering thanks to a goal-directed behavior.

Organization MultiAgent Systems (OMAS) are viewed as an effective paradigm for addressing the design challenges of large and complex MAS, where organizations are emergent whenever agents work together in a shared environment. Many similarities exist between OMAS and service oriented approaches. They both meet the loose-coupled, flexibility and dynamicity features. Organizations are ways to makeup systems of collaborative services (Singh and Huhns, 2005). The nature of

agents, as autonomous entities with auto-organized capabilities and high level interactions, facilitates automatic service discovery. For all these reasons, we restrict our research to OMAS models; so they are implicitly meant whenever we refer to agent models in the rest of the document.

Component and agent approaches have then some common points but each one has its own key features which are not shared between them. For instance, the work of (Lind, 2001) and (Schiaffino and Amandi, 2004) reflect the lack of reusability in agent approaches. The other limitation of agent approaches is the loss of control caused by autonomy properties of agent which reflects the need for robustness properties. On the other hand, components suffer from the lack of dynamicity and reasoning features (Bergenti and Huhns, 2004). Components need more open and abstract types of interactions when they depend on provided services of heterogeneous entities to accomplish composition requirements. To summarize, a component is quite equivalent to a reactive agent whereas an agent can be viewed as a component that interoperates with its peers by exchanging messages in a significant Agent Communication Language (ACL) (FIPA-ACL, 2002).

Figure 1 represents the triangle relationships among services, agents and components. Services can be abstract specifications of agents and components. Agents offer dynamical and behavioral features versus reusability and composability ones for components. Our goal is to implement this triangle allowing the integration of agent and component approaches to overcome their shortages by adding features coming from the other domain. Services play a key role as being an interoperability pivot between agents and components.

We propose to define a specification process allowing the specification of a same application through several models: only as abstract services, through agents implementing services, through components implementing services and through a mix of agents and components implementing services. In this paper, we present a global view of our specification process and we define three of its models (service, agent and component ones) and their relationships.

The rest of this paper is structured as follows: in the next section, we present a motivating example showing the interest of mixing agents and components. An overview of the specification process is given in section 3. Its component, agent and service models are presented in section 4 with the definition of their main concepts and a discussion of their common points. Related works are discussed in section 5, before concluding and presenting some perspectives.

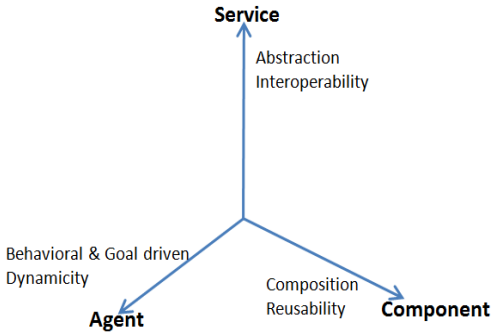

Figure 1: Triangle of relations between components, agents and services.

2 MOTIVATING EXAMPLE

Figure 2 presents our case study. It is a typical holiday reservation system. A client addresses the travel agency to find his appropriate vacation according to some criteria's like the number of persons, date, price, place and theme. This travel agency is based on an OMAS approach where it represents a group of agents (A1, A2). Each agent owns his personal network of hotels and airline companies according to geographical zones. However, if an agent does not find a corresponding hotel or flight reservation for the needs of the client, he may negotiate with other agents within his group. For instance, the agent A1 did not find the appropriate flight in his network, so he negotiates with A2 and makes a commitment with him to reserve the flight. This commitment may include a commission for A2 and an agreement of the quality and reliance of the reservation process. Hotels and airline companies are realized by components, where they may be presented by a primitive component or by a composite one (hotels X). Many reasons stand behind our choices of representation: we mention here that both of travel agency and client actors need spaces of autonomy in taking decisions and dynamicity in interacting with other parties in order to negotiate and coordinate with them. The tasks of these actors are not just about querying their databases or reusing services for their sub branches which are the case in hotels and airline companies' actors.

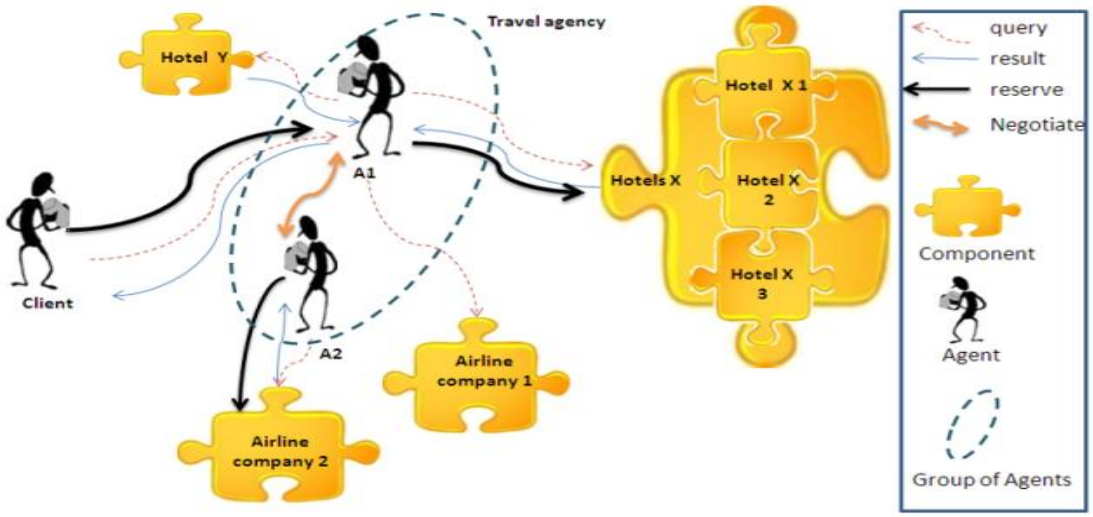

Figure 2: Architecture of a holiday reservation system

We can see that there are many interactions between components and agents in order to exchange their services. A simple type of interaction can be achieved by basic communication, such as a single and basic service call, but this is not sufficient when the parties need to negotiate for a price or a date to make certain compromise to gain the trust of the client. Unfortunately, there is no such flexibility in the communication with components' services. Then, we need to have more complex and dynamic communication protocols. At the same time, the service provided by agents in the travel agency (providing offers for vacation) can be useful in other contexts. For example, the travel agency may provide special offers for local products of the target destination. But we cannot reuse this service in different contexts as agents are not customizable. Then, we need to design the same service for each purpose. These two limitations reflect the need to raise the level of interaction between agents and components and to offer component features to agents and conversely.

3 OVERVIEW OF OUR PROCESS

Our process is composed of a hierarchy of four models as shown on figure 3. Three models are at the same level: component, agent and mix of the two approaches (in CASOM: Component Agent Service Oriented Model). The more abstract model is the one based only on services without requiring to define the elements (agents or components) implementing these services. As a result, these four models allow the specification of an application by several ways: with only services, with only components, with only agents or with a mix of agents and components (in the last three models, agents and components implement services). This process can be interpreted with different entry points:

- A **top-down** approach: we can project any application viewed as sets of collaborative services corresponding to our abstract service model into another specification conforming to one of the three other models (component, agent or CASOM). More details on the implementation can further be added by projecting a specification conforming to one of these three models to a specification conforming to a concrete implementation model such as EJB (Michiel et al, 2001) and Fractal (Bruneton et al, 2003) for components, AGR (Ferber et al, 2004) and OMNI (Vàzquez-Salceda et al, 2005) for agent models and AgentComponent (AC) for a mixed agent/component approach (Krutisch et al, 2003).
- A **bottom-up** approach: any application implemented by any existing component or agent model or mix of them can be abstracted and viewed as a service oriented application. For example: we can start from an application implemented by the Fractal component model. This application must be made conforming to our component model, and then it can be abstracted as only formed of collaborated services, conforming to our abstract service model.
- A **middle-out** approach: the emphasis here is put on the interaction aspects between the components and agents of the system to reach

their integration. Services play here the key role in their interoperability, as services clarify the specification of what an agent or a component does. (Krutisch et al, 2003) defines the notions of *agentification* and *componentification.* The agentification is the added value by agent properties to existing components, and the reverse for the componentification. These two actions enable the transformation of any application specification into an agent only one, a component only one or a mixed one. They will enable to enhance easily any existing component or agent application specification. The agentification (resp. componentification) transformation rules can be applied directly between the two component and agent models or passing through CASOM.

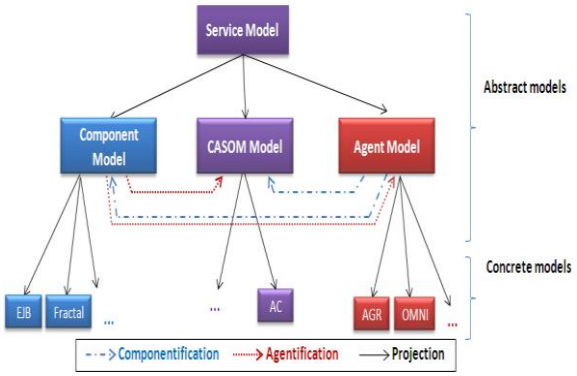

Figure 3: The main models of our process

As none of existing models for agents, components and services completely fulfills the requirements from our point of view (we aim to highlight the interactions and service definitions in an application specification through either components, agents or both entities), we need to define our own unified models for each domain (component, agent, service and CASOM). The next section provides more details about the design of the three component, agent, and service models (we are still working on the definition of the CASOM model).

4 AGENT, COMPONENT AND SERVICE MODELS

4.1 Studied models

We provide here a brief overview of the study of already existing models in each domain, which led us to define our own models. In order to design these models, we firstly unified the concepts that already exist and vary between the essential models under the domains of component, agent or service. Then we focused on the existence of the two key concepts of interaction and service, whether they appear implicitly, explicitly or are not present at all. Indeed, as explained previously, we want to make interoperate agents and components; interaction is therefore a key point. Moreover, services will play a key role as being the pivot for specifying what an agent or a component does. Finally, we defined our own models containing the main significant concepts according to our requirements and consistently to all studied models.

Regarding components, we studied DARWIN (Magee et al, 1995) and ACME (Garlan et al, 1997) for the Architecture Description Language (ADL) models, Enterprise Java Beans (EJB) and Corba Component Model (CCM) (OMGc) for industrial component models, and the Fractal model as an academic model. UML 2.0 (OMGa, 2007) includes also a general component model for conceptual purpose. As a brief result for this study, we found that the concept of service exists implicitly in most of component models, through the interfaces of a component. Considering the interaction concept, we found that components use mainly basic types of interaction in their binding and delegation for a vertical or hierarchical composition, typically remote procedures call or message passing. Even if the concept of connector exists in some models, allowing the definition of complex interactions, from a practical point of view, it is not always considered or used as a first class entity.

Regarding agents, we studied well known agent models like AGR (Agent Group Role model) (Ferber et al, 2004, (Odell et al, 2004)), MOCA (Model Organizational and Componential for Multi Agents) (Amiguet, 2003), MOISE/MOISE+ (Model of Organization for Multi Agent Systems) (Hannoun et al, 2000), OMNI (Organizational Model for Normative Institutions). We studied more recent agent models such as FAML (Beydoun and Low, 2009) and GORMAS (Argente et al, 2009), and some methodologies for Agent Oriented Software Engineering, like GAIA (Zambonelli et al, 2003) and PASSI (Cossentino, 2005). We found that the concept of service is implicit in most of agent models under the concepts of role, capability or behavior. However, it exists explicitly in the cited methodologies and also in FAML and GORMAS models. By nature, agent models embed complex and high-level interaction protocols, such as Auction protocol (Vetter and Pitsch., 1999).

Concerning services models, the basic conceptual Service Oriented Architecture (SOA) model consists of two main actors: service provider and consumer. The Service oriented architecture Modeling Language (SoaML) (OMGb, 2009), provided by the OMG, concentrates on the collaboration between participants through service contracts. Another well known model is the Service Oriented Architecture Reference Model (SOARM[(1)]) provided by OASIS. It is a general model which defines the main concepts that should be considered in the design of any system adopting a SOA approach. Compared to agent and component models, for which we directly unify their main concepts according to our requirements, we have a specific constraint for the service model. The goal of this model is to specify an application at an abstract level, that is, without defining the kind of elements that are implementing the services. Then, the concept of participant that exists in some service models is not relevant from our point of view. Considering the concepts of interaction between services, it is achieved either by sending messages or by business protocols to achieve the aggregation of services (a choreography) or to make recursive composition of services to a new service that has central control over the whole process (an orchestration[(2)]).

4.2 Defined models

Figure 4 presents the three models of agent, component and service we defined as a result of the study of existing models for each domain. These models are presented through a single diagram (following the UML class diagram notation). The main reason that stands behind this representation is that it helps in visualizing common concepts between these three models. This emphasizes the possibility of integration of agents, components and services. The second reason is pragmatically to deal with the paper page limitation. All the concepts presented on the three models are defined in table 1.

Interaction and service concepts are key points in our approach for ensuring interoperability among agents and components. As seen, they exist, explicitly or implicitly, in all studied approaches: they are then present in our three models. Services are associated with interactions of roles played by elements (agents, components or services) and with their operations. The concept of protocol specifies the behavior of an interaction. As all these concepts are the same (or very similar) in the three domains, they are shared by the three models.

Others concepts are in the same way shared by two domains, among components and services, such as the basic interaction type and the service point (provided and required) concepts. However, if we exclude the common concepts between the three domains, the agent model does not share concepts with neither the service model nor the component one.

The last category of concepts concerns the specific concepts to each model. For instance, agent, goal, group, task or organization concepts are dedicated to the agent model and component, connector, connector component concepts to the component model. Some of these concepts are also specialization of general and shared concepts. For instance, the agent model defines the concepts of negotiation, coordination and communication that specialize the interaction concept and an agent capability is a specialization of the service concept. In the same way, for the component model, a connector is a specialization of interaction. Finally, some concepts are mainly the same but are not reified in the same way: services for the service model and components for the component model can be either primitive or composite.

As a conclusion on these models, we see that most of the service model concepts are shared by the agent and the component models. This is consistent with the fact that services are a pivot for interoperability among agents and components and that, in our specification process, the service model specifies in a more abstract way applications built with agents and/or components. We also retrieve the characteristics of agents and components described in the introduction: agents offer more high level and variety of interactions and components are more structurally defined, allowing a better reusability (through composition and service point features). Finally, component and service models are close; they mainly define the same concepts. The main difference is in the translation of the composition feature from abstract entities (the services) to concrete one (the components). In the same way, the reification of the interaction is made within an additional concrete element in the component model: the connector. The component model is somehow a concrete specialization of the service model. On the contrary, the agent model owns more specific features and is relatively different from the component and service models.

(1) http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf

(2) http://www.soa-in-practice.com/soa-glossary.htm

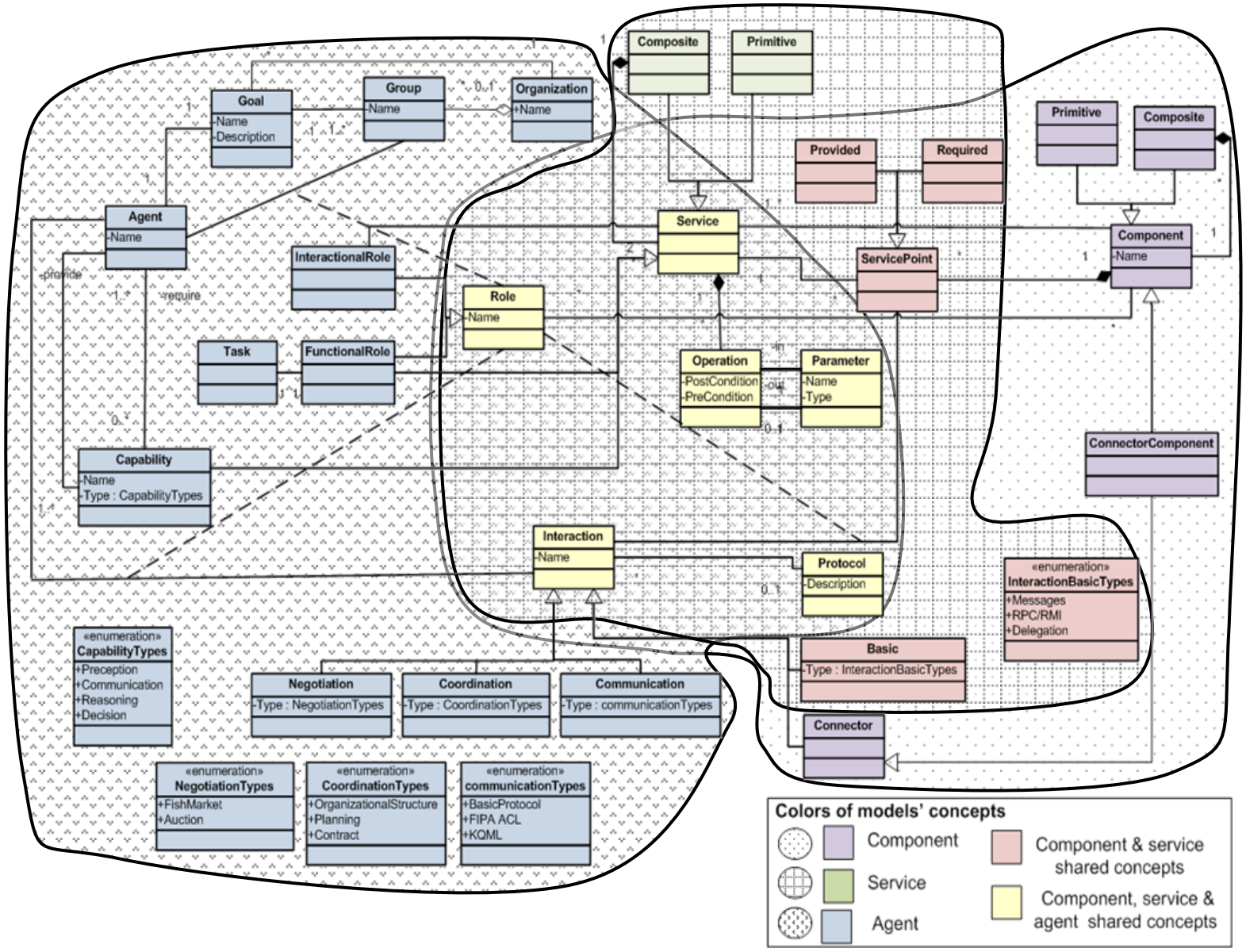

Figure 4. Models of service, component and agent with their shared concepts

Table 1. Definition of the concepts of service, component and agent models

Concept	Model	Definition
Service	Service	A logical representation of a repeatable business activity that has a specified outcome. It can be **composite** or **primitive**.
	Component	Definition of a static unit of functions (a classical component interface).
	Agent	Specification of agent's behavior and capabilities (provided or required ones).
ServicePoint	Service	Access point to the service, where it determines the way to use it (as **provided** or **required**).
	Component	A port of a component either exposing some of its services (**provided** specialization) or specifying services it has to use (**required** specialization).
Operation	Service Component Agent	An operation associated with a service, including the required pre and post conditions for its application.
Parameter	Service Component Agent	Inputs parameters for an operation or output ones for exposing its results.
Role	Service	The responsibilities a service takes through its service points within an interaction with other services.
	Component	The responsibilities a component takes through its service points within an interaction with other components.
	Agent	The role that an agent plays within a group or an organization. It represents also the responsibilities and the tasks that an agent assumes within an interaction with other agents. It can be of two types: **Functional** or **interactional**.
Interaction	Service	A kind of action or influence in the dynamic relation between services in order to respond to their needs, which enable them to achieve their expected results.
	Component	Communication between components through their service points to exchange their services. It can be from different types: **basic** or more complex through a **connector**.
	Agent	The dynamic relation between agents through their played roles. It has different types: **communication**, **coordination**, **negotiation**.
Protocol	Service	An orchestration or choreography process flow specification between services.
	Component	A complex interaction specification between components.
	Agent	Specification of the types of interactions between agents from basic types like Message Transferring Protocol to the negotiation and coordination ones.
Basic	Service, Component	A low-level communication, such as RPC/RMI (Remote Procedure Calls, Remote Method Invocation), message passing or delegation between services/components for a service/component composition.
Component	Component	A reusable entity with well specified access points (service points) to expose or use services. It can concretely be **primitive** (simplest type of component) or **composite**. The primitive component is the basic entity in an assembly of components (horizontal composition) or their hierarchical composition (vertical composition). A composite component is built by the composition of primitive or composite components, without nesting limit.
Connector	Component	The explicit representation of a complex interaction. A connector can be itself a component through the connector component entity. Its behavior is specified by a protocol.
Connector Component	Component	An interaction at the same level of a component (Cariou et al, 2002). It can be primitive or composite.
Agent	Agent	An autonomous rational entity.
Group	Agent	A structural entity composed of roles and agents. An agent can be member of one group if and only if he plays a role associated with this group.
Organization	Agent	The overall architecture of the system that is the position of each role in the organization and its relationship with other roles. It defines also the authority between set of agents in a group or between groups.
Capability	Agent	The knowledge or capacities that an agent owns to play his role in a group or to participate within an interaction. Some capabilities are created with the agent and others are acquired through the agent life. An agent may need to use other agent capabilities if he does not own them. Perception, communication, reasoning and taking decisions are from the essential capabilities of agents.
Goal	Agent	Functional requirements of the organization. It could be divided in sub goals related to agents or to groups. Agents may have their individual goals which lead to possible conflicts with the collaborated ones.

Interactional Role	Agent	A classical role used in the definition of interaction protocols. This role enables an agent to expose or to request the needed services.
Functional Role	Agent	The definition of the tasks that must be carried out in the system. It may also present the agent behaviour.
Task	Agent	A unit of action that the agent performs and that does not require interaction with any other agent.
Communication	Agent	Communication through an agent language like ACL.
Coordination	Agent	Coordination among agents that share some resources, to avoid conflicts, such as coordination by planning, coordination through the organizational structure or by signing contracts (FIPA, 2002)
Negotiation	Agent	Negotiation when a compromise has to be reached between some agents to solve occurred conflicts, such as auction and Fish market protocols (Fishmarket, 1999).

4.3 Integration of agent, component and service models

There are two main future steps concerning the specification process. The first one is to define the CASOM model, allowing the specification of an application with agent and component features simultaneously. With this model, a component will make use of the advanced types of interaction of agents. On the other side, an agent will become customizable and reusable by using the service point and composition concepts.

By using the CASOM model, we will be able to specify the mixed agent/component application example of the section 2.

The second step consists in defining the semantics mappings and transformations among all these four models. We have seen above that there are strong links between the component and the service models; transformations between these models will then be almost easy to define. However, concerning the agent model, there are more important differences with the component or the service model. Then, mappings will be more complex. As an illustration, here are some ideas of what they can be: the concept of agent can be mapped to a primitive component. Then, the concept of group can be mapped to a composite component. The concept of capability of an agent can be mapped to a service point concept in the component model and so on.

5 RELATED WORK

In this section, we present existing works that are interested in mixing agent, component and service approaches, or at least two of them.

Some works deal with the integration of component and agent approaches depending on delegation between these entities. (Krutisch et al, 2003) uses the componentification approach for defining new entities named AgentComponent (AC) which are originally agents but encapsulated into components. We can find in (Aniorté and Lacouture, 2008) another interesting work where components are automatically adapted by attaching them to agents. This work can be listed under the agentification notion where components gain the dynamicity feature from agents.

Other works use both of component and service as key concepts. A detailed comparison between the service and component oriented software engineering is provided in (Breivold and Larsson, 2007). Service technologies (such as Web services) are considered as a special type of components. The reverse is also true in other works like in (Herault et al, 2004), where the component is used to specify subtasks of a service. A new entity named ServiceComponent is proposed in (Zhang et al, 2008). This entity represents business requirements and is at the same time a functional unit based on traditional ADL components.

Agents and services are also key concepts for many researches, such as (Preist et al, 2001) which assume that the agent technology is the best way to reach service composition by negotiation. An agent behaves also as a service aggregator (or market maker) in (Papazoglou et al, 2005).

(Hahn et al, 2010) provides a model driven approach for the integration of agent and services. It shows the possibility of the transformation between SoaML and a defined platform independent (meta-) model of agents (PIM4AGENT). This work has been previously started in (Hahn et al, 2006) where the authors studied the Believe Desire Intention (BDI) models of agents (Rao and Georgeff, 1995). It is extended in (Hahn et al, 2010) to integrate the organizational dimension. Excepting the absence of components, this work is quite similar to our approach with a main point of difference: SoaML is chosen to represent the SOA model where it contains the notion of agent for concretely representing a participant in an application specification. While in our approach, the service

model is more abstract and does not integrate this concept of participant.

As a conclusion, as far as we know, there is no work similar to ours where agents, components and services are consistently integrated together.

6 CONCLUSION

In this paper, we present an approach for integrating component and agent approaches using interoperable services as the pivot of this integration. Our goal is to overcome the respective shortages of each domain by adding features coming from the other one. To achieve this goal, we propose a specification process. It is composed of four models allowing the specification of an application with agents, components, mix of agents and components, or only as abstract services. Three of them have been presented in this paper: the component, agent and service ones. They focus on the interaction concept, in addition to the service one, for ensuring interoperability between agents and components. The main contribution of our work is in studying the three domains of component, agent and service simultaneously, while many other approaches just study pair of them as seen in the related work section.

Our next steps are: a) the design of the CASOM model enabling using both concepts of agent and component through interoperable services in the same application specification and b) the definition of the semantic mappings among the four models. Specifying notably the two essential actions of agentification and componentification is from our main perspectives. We wish to apply the agentification process on any existing component application to reach an application specified with only agents or with agents and components together and conversely.

The specification process is currently being implemented as a Model-Driven Engineering platform using the EMF framework (Eclipse). Each model is a Domain Specific Language (DSL) and all mappings, projections and actions among models (including agentification and componentification) will be realized through automatic model transformations.

REFERENCES

Amiguet, M., 2003. MOCA: Un modèle componentiel dynamique pour les systèmes multi-agents organisationnels phd thesis. Université Neuchatel Institut d'Informatique et d'Intelligence Artificielle, Suisse.

Aniorté, P. and Lacouture, J., 2008. CompAA: A Self-Adaptable Component Model for Open Systems. In ECBS '08: Proceedings of the 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems, Washington, DC, USA, pp.19–25. IEEE Computer Society.

Argente, E., Botti, V. and Julian,V., 2009. GORMAS: An Organizational-Oriented Methodological Guideline for Open MAS, 10th Int. Workshop on Agent-Oriented Software Engineering (AOSE) pp. 85-96.

Bergenti, F., Huhns, M., 2004. On the use of agents as components of software systems. In Methodologies and Software Engineering for Agent Systems The Agent-Oriented Software Engineering handbook, New York. Kluwer Academic Publishing, pp 19-32.

Beydoun, G., Low, G., Henderson-Sellers, B., Mouratidis, H., Gomez-Sanz, J., Pavon, J. and Gonzalez-Perez, C. 2009. FAML: A Generic Metamodel for MAS Development. In IEEE Transactions on Software Engineering, vol 35, pp 841-863.

Breivold, H. P. et Larsson, M., 2007. Component-Based and Service-Oriented Software Engineering: Key Concepts and Principles. In euromicro'07: Proceedings of the 33rd euromicro Conference on Software Engineering and Advanced Applications, Washington, DC, USA, pp. 13–20. IEEE Computer Society.

Bruneton, E ., Coupaye, T. et J. Stefani, 2003. The Fractal Component Model, Object Web Consortium, Technical Report Specification V2.

Cariou E, Beugnard, A. and Jezequel J.-M, 2002. An Architecture and a Process for Implementing Distributed Collaborations. In The 6th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2002). IEEE Computer Society.

Cossentino, M , 2005. From Requirements to Code with the PASSI Methodology, in Agent-Oriented Methodologies, B. Henderson-Sellers and P. Giorgini, IdeaGroup Inc, Hersey, PA, USA.

Eclipse, http://www.eclipse.org/emf/.

Ferber J., Gutknecht O., Michel F., 2004. From Agents to Organizations: an Organizational View of MultiAgent Systems, in Agent-Oriented Software Engineering (AOSE) IV, P. Giorgini, Jörg Müller, James Odell, eds, Melbourne, LNCS 2935, pp. 214-230.

FIPA-ACL, 2002. Message structure specification. http://www.fipa.org/specs/fipa00061/

FIPA, 2002. Contract net interaction protocol specificationFIPA specifications SC00029H, Foundation for Intelligent Physical Agents. http://www.fipa.org/specs/fipa00029/SC00029H.htm.

Fishmarket Project, 1999. http://www.iiia.csic.es/Projects/_shmarket/, 04/01/1999.

Garlan, D., Monroe, R. and Wile, D., 1997. Acme : an architecture description interchange language. In CASCON '97: Proceedings of the conference of the Centre for Advanced Studies on Collaborative research. IBM Press.

Hahn, C., Jacobi,S. and Raber, D. 2010. Enhancing the interoperability between multiagent systems and

service-oriented architectures through a model-driven approach. In Proceedings of the 8th German conference on Multiagent system technologies (MATES'10), J 252;rgen Dix and Cees Witteveen (Eds.). Springer-Verlag, Berlin, Heidelberg, pp. 88-99.

Hahn, C., Madrigal-Mora, C., Fischer, K., Elvesaeter, B., Berre, A.-J. et Zinnikus, I.,2006. Meta-models, models, and model transformations : Towards interoperable agents. In MATES, pp. 123–134.

Hannoun, M., Boissier, O., Sichman, J. S. et Sayettat, C. , 2000. MOISE : An organizational model for multi-agent systems. In Proceedings of the International Joint Conference,th Ibero-American Conference on AI 7, 15th Brazilian Symposium on AI, pp. 152–161. Springer.

Herault, C., Lecomte, S. and Delot,T., 2004. New Technical Services Using the Component Model for Applications in Heterogeneous Environment. In LNCS 3473,Workshop of Innovative Internet Community Systems.

Krutisch, R. Meier, P. and Wirsing, M., 2003. The Agent Component Approach, Combining Agents, and Components. In MATES, volume 2831, pp. 1-12.

Lind. J., 2001. Relating Agent Technology and Component Models. http://www.agentlab.de/documents/Lind2001e.pdf.

Magee, J., Dulay, N., Eisenbach, S. and Kramer, J., 1995. Specifying Distributed Software Architectures. In Proceedings of the 5th European Software Engineering Conference, London,UK, pp. 137–153. Springer-Verlag.

Michiel, L., Yalcinalp, L. et Krishnan, S., 2001. Enterprise Java Beans Specification Version 2.0.

Odell, J., Nodine, M., Levy, R. , 2004. A Metamodel for Agents, Roles, and Groups, AOSE, LNCS 3382, pp. 78–92.

OMGa, 2007. UML 2.0 Superstructure Specification. http://www.omg.org/cgi-bin/doc?ptc/2003-08-02,2007.

OMGb , 2009 . http://www.omg.org/spec/SoaML/1.0/Beta2/PDF/09-12-09.pdf.

OMGc, CORBA Component Model Specification Version 4.0.

Papazoglou, M., Aiello, M., Giorgini, P., 2005. Service-Oriented Computing and Software Agents. In L. Cavedon, Z. Maamar, D. Martin, B. Benatallah editor(s). Kluwer.

Preist, C., Byde, A., Bartolini, C. and Piccinelli, G., 2001. Towards agent-based service composition through negotiation in multiple auctions. AISB Journal 1(1).

Rao, A. S., Georgeff, M., 1995. BDI Agents : from theory to practice. In First International Conference on Multi-Agent Systems (ICMAS-95), S. Francisco, CA, pp. 312–319. MIT Press.

Schiaffino, S., Amandi, A., 2004. User - interface agent interaction: personalization issues, int. j. hum.-comput. stud. 60(1), pp.129_148.

Singh, M.P. and Huhns, M.N., 2005. Service-Oriented Computing: Semantics, Processes, Agents, John Wiley & Sons.

Vàzquez-Salceda, J., Dignum, V. et Dignum, F., 2005. Organizing Multiagent Systems. Autonomous Agents and Multi-Agent Systems 11, pp. 307–360.

Vetter, M., and Pitsch, S., 1999. An Agent-based Market Supporting Multiple Auction Protocols. In Workshop on Agents for Electronic Commerce and Managing the Internet-Enabled Supply Chain, Third International Conference on autonomous agents (Agents '99) in Seattle, Washington (USA).

Wooldridge, M., 2009. An Introduction to MultiAgent Systems (2nd édition). John Wiley & Sons Ltd.

Zambonelli F. Jennings N. and Wooldridge M., 2003. Developing Multiagent Systems: the Gaia Methodology. ACM Transactions on Software Engineering and Methodology 12(3), pp. 417-470.

Zhang, T., Ying, S., Cao, S. and Zhang, J., 2008. A modelling approach to service-oriented architecture. Enterp. Inf. Syst., 2(3):pp. 239-257.