Towards a Component Agent Service Oriented Model

Nour Alhouda Aboud, Eric Cariou and Eric Gouardères

LIUPPA Laboratory – Université de Pau et des Pays de l'Adour
BP 1155 – 64013 Pau Cedex – France
{Nour-alhouda.Aboud, Eric.Cariou, Eric.Gouarderes}@univ-pau.fr

Abstract. Organizational multi agent 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; we consider it as the interaction point between agents and components. We will define a model-driven engineering process composed of several DSLs (Domain Specific Languages). They are dedicated to specify an application through several aspects: services, components, agents and mix of them. Several transformations and projections will allow the addition of agent or component features into an application specification. In this paper, we present a global view of this process and of its DSLs.

1 Introduction

Twenty years ago, information systems were homogeneous, monolithic and centralized. Traditional mature approaches such as object-oriented software engineering were sufficient. 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. Service approaches view applications as sets of services that interact between each other independently of their locations to satisfy heterogeneous and loose-coupled software systems.

Organizational Multi Agent Systems (OMAS) are viewed as an effective paradigm for addressing the design challenges of large and complex multi agent systems where organizations are emergent whenever agents work together in a shared environment (Beydoun *et al.*, 2009). One of the main features of OMAS is to provide interaction and social patterns (auctions, mediator ...) in order to coordinate autonomous and proactive entities (agents). Each agent manages its activity through goal-directed behavior based on mental states and commitments. Many similarities exist between OMAS and service-oriented approaches. They both meet the loose-coupled, flexibility and dynamicity features. The organizations in OMAS are ways to makeup systems of collaborative services. The nature of agents, as autonomous entities with auto-organized capabilities and high-level interactions facilitates automatic service discovery.

Component approaches are based on the main interest of reusing blocks of code that implement well-specified interfaces. This enables efficient solutions for defining well-structured and robust applications by composing and reusing components. Component interfaces can be considered as definition of services and service approaches can be viewed as logical extension of component ones as both of them meet reusability and composition purposes.

Component and agent approaches have each one their own key features in building applications but, unfortunately, they are not all shared. Lind (2001) and Schiaffino and Amandi (2004) reflect the lack of reusability in agent approaches. Their other limitation 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 because of their dependency on the provided services of other heterogeneous entities.

Our goal is to integrate component and agent approaches to overcome their shortages and being able to make use of all their features within a same application definition. As seen, services are present in both approaches. We propose to explicitly define the services that an agent or a component offers or requires and to make them interoperate through these services. Services will be the key point of integration through high-level of interaction between elements forming the application, it will help in clarifying the specification of what an element does. 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.

Following these principles, the specification of an application will be realized through a Model-Driven Engineering (MDE) process. It will be composed of several models dedicated to specify an application through several aspects: services, components, agents and mix of them in a Component Agent Service Oriented Model (CASOM). This latter model enables to specify an application by using interoperable agents and components through their exchange of services. In this position paper, we pave the road for our target CASOM model definition by browsing our MDE process, through a motivating example. Then, we conclude with our main perspectives.

2 Motivating example

Figure 1 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 (small hotels or airline companies) or by a composite one (hotels X). Many reasons stand behind the choices to represent the travel agency and client actors by agents and the hotels and airline companies' actors by components. 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.

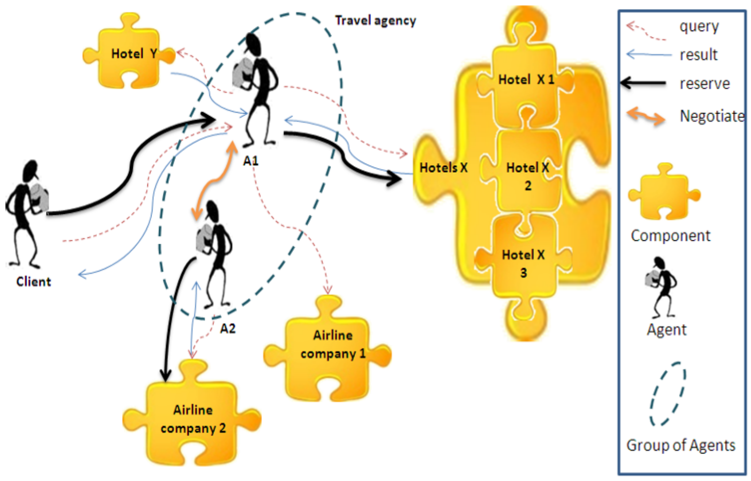

FIG. 1. *Architecture of a holiday reservation system*

We can see that there are many interactions between components and agents in order to exchange their services, like the interaction between a service of room reservation provided by a hotel component and a searching for room reservation service required by a travel agency agent (these services are not detailed on the figure). 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 Description of the global process

Our process is composed of a hierarchy of four models – that is four DSLs (Domain Specific Languages) – as shown on figure 2. Three models are at the same level: Components, OMAS and mix of the two approaches (the CASOM model). The more abstract model is the one based only on services without requiring to define the elements (agents or components) implementing these services. Then, we are able to define an application at a high level of abstraction, only through its defined services and their interactions. 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).

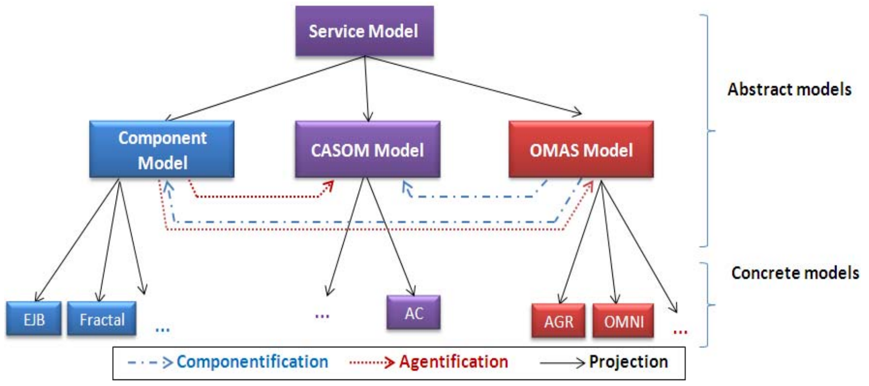

FIG. 2. *The main DSLs in our MDE process*

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 specifications 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). We have firstly unified the concepts that already exist and vary between the existing models under each component, agent or service approaches. We focus on the existence of the two key concepts of interaction and service, whether they appear implicitly, explicitly or are not present at all. As a result, we have already defined the three unified models for component, agent and service domains. We are currently working on the definition of the CASOM model, mixing agents and components.

From figure 2, we can see that a specification conformed to the abstract service model can be projected into another specification conformed to one of the three other models (component, OMAS or CASOM). This allows the specification of which elements (agents and/or components) are implementing the abstract defined services. More details on the implementation can further be added by projecting a specification conformed to one of these three models to a specification conformed to a concrete implementation model such as EJB[1] or Fractal (Bruneton *et al.*, 2003) for components, AGR (Ferber and Gutknecht, 1998) or OMNI (Vàzquez-Salceda *et al.*, 2005) for agents and AgentComponent (AC) for a mixed agent/component approach (Krutisch *et al.*, 2003).

Finally, once an application is specified under the form of a set of agents and/or component technology, a part of the final application code can be generated, such as code skeleton for agents and components, and the definition of the services (WSDL files if using Web services for instance).

The notions of *agentification* and *componentification* presented on figure 2 are browed from (Krutisch *et al.*, 2003). The authors define the agentification as the added value by agent properties to existing components, and the reverse for the componentification. They will enable to enhance easily any existing component or agent application specification.

Concretely, all above described projections and actions of agentification and componentification will be achieved through model transformations. The goal of our MDE process is

[1] Sun Microsystems. Entreprise Java Beans Technology, `http://java.sun.com/products/ejb/`

then, through automatic or guided transformations, in a vertical way to go from an abstract specification to a concrete one for a given technology, and in a horizontal way, to be able to add and mix agents and components depending on the application feature requirements.

4 Conclusion

In this paper, we presented an approach for integrating agents and components into an application specification. This approach is based on viewing services as key points in the interoperability between agents and components. We browsed a MDE process based on four DSLs: a service, an agent, a component and a mixed CASOM models. The three first models are already defined and the main remaining task is to design the CASOM model. It will facilitate the specification of heterogeneous complex systems, allowing the use of components and agents into a same application specification. CASOM will also smooth the progress of the transformation of any application based on components or organizational multi agent systems to a service oriented application and conversely. Once CASOM is defined, we will clearly specify the actions of agentification and componentification. Then, we will be able to implement the complete MDE process, including all required transformations and projections that will ensure interoperability between our four models.

References

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

F. Bergenti and M. N. Huhns (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.

E. Bruneton, T. Coupaye, and J. Stefani (2003). The Fractal Component Model, Object Web Consortium, Technical Report Specification.

J. Ferber, and 0. Gutknecht (1998). A Meta-Model for the Analysis and Design of Organizations in Multi-Agent Systems. In ICMAS '98: Proceedings of the 3rd International Conference on Multi Agent Systems, IEEE Computer Society, pp 128 - 135.

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

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

S. Schiaffino, A. Amandi (2004), User-Interface Agent Interaction: Personalization Issues, In Int. J. Hum.-Comput. Stud. 60(1), Academic Press, Inc, pp 129-148.

J. Vàzquez-Salceda, V. Dignum, and F. Dignum (2005). Organizing Multiagent Systems. In Autonomous Agents and Multi-Agent Systems, 11(3), Kluwer Academic Publishers, pp 307-360.