A Service-Oriented Approach for Integrating Multiagent System Designs° (Extended Abstract)

Walamitien H. Oyenan
Multiagent and Cooperative Robotics Lab
Kansas State University
234 Nichols Halls,
Manhattan Kansas, USA
oyenan@ksu.edu

Scott A. DeLoach
Multiagent and Cooperative Robotics Lab
Kansas State University
234 Nichols Halls,
Manhattan Kansas, USA
sdeloach@ksu.edu

Gurdip Singh
Department of Computing and Information Science
Kansas State University
234 Nichols Halls,
Manhattan Kansas, USA
gurdip@ksu.edu

ABSTRACT

As agent technology acceptance grows, there is a need for software engineering approaches to deal with the design of large, complex multiagent applications. Currently, existing approaches work well for small systems but are not well suited for large, complex applications. Thus, we propose extending multiagent approaches with service-oriented principles to simplify the design of such systems. Our approach composes small multiagent-based services to build larger, more complex applications. This paper introduces the key concepts required to design reusable multiagent services and shows how the services can be composed to create complex multiagent applications.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems; D.2 [**Software Engineering**]: Architectures.

Keywords

Multiagent Organizations, Agent-Oriented Software Engineering

1 INTRODUCTION

Organization-based Multiagent Systems (OMAS) have been viewed as an effective paradigm for addressing the design challenges posed by today's complex systems [3, 5]. In those systems, the organizational perspective is the main abstraction, which provides a clear separation between agents and systems, allowing a reduction in the complexity of the overall system. To ease the development of OMAS, several methodologies have been proposed (for an overview, see [2]).

Unfortunately, those methodologies typically require the designer to handle system complexity alone, which tends to lead to ad-hoc designs that are not scalable and are difficult to maintain. Designing organizations for large multiagent systems is a complex and time-consuming task; design models quickly become unwieldy and thus hard to develop and maintain.

It has been long suggested that decomposing OMAS into loosely coupled sub-organizations would help cope with the complexity. However, current methodologies simply suggest that large organizations be decomposed without providing a rigorous process to recombine them (e.g. [3],[5]).

Therefore, we propose the use of reusable OMAS that are designed using service-oriented principles. The service-oriented approach allows us to decompose large multiagent organizations into smaller organizations that can be developed separately and composed when needed, thus providing designers with a scalable approach to designing large and complex OMAS. In our approach, we view services as basic elements to develop multiagent organizations. *Services* are independent multiagent organizations encapsulating common OMAS functionalities. Services are typically cooperative tasks that require the cooperation of several agents. These service organizations are then *composed* with other organizations to producer larger, more complex systems. A typical example of an OMAS service might be a routing service that requires several agents acting as hops to deliver a message to its destination. Hence, our work differs from other approaches that consider service only at the agent level.

Our approach is based on the Organization Model for Computational Adaptive Systems (OMACS) [1]. OMACS is a formal framework for describing the key elements of OMAS and is supported by a rigorous methodology tailored from the O-MaSE process framework [4]. There are many multiagent organizational models [2] and our proposed approach could be adapted to any of them.

In our approach, each organization exposes generic interfaces called *connection points*. Connection points are organizational goals and roles that can *provide* (supply) or *use* (consume) services. Organizations can then be *composed* such that required services match provided services. The "glue" for the composition process is a *connector*, which links connection points together. The composition process ensures the consistency of all the bindings and results in a valid composite organization.

2 SERVICE MODEL

In our approach, services are provided by multiagent organizations encapsulating functionalities commonly used in multiagent systems. Once services are designed, they can be used by other organizations in order to build larger systems. Figure 1 presents our Organizational Service metamodel, which integrates organization and service concepts. The central concept is that of *Service*. A Service is a logical entity that represents a coarse-grained multiagent functionality. This coarse-grained functionality

Cite as: A Service-Oriented Approach for Integrating Multiagent System Designs, (Extended Abstract), Walamitien H. Oyenan, Scott A. DeLoach, Gurdip Singh, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009),* Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. 1363–1364

° This work was supported by grants from the US National Science Foundation (0347545) and the US Air Force Office of Scientific Research (FA9550-06-1-0058).

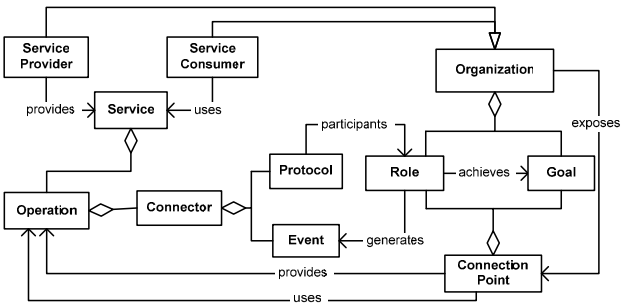

Figure 1. Organizational Service Metamodel

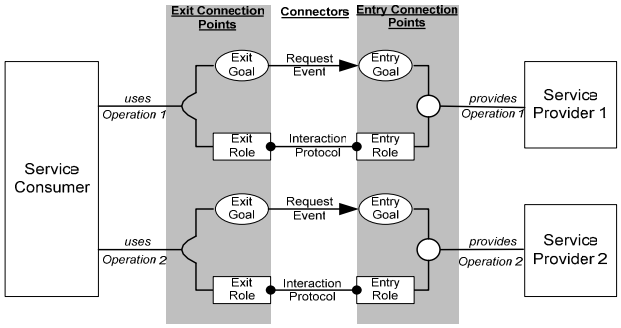

Figure 2. Composition of Organizations.

is made of a set of fine-grained functionalities, called *operations*, which can be requested during the design of other applications.

Typically, multiagent organizations implement a service by implementing all the operations declared for that service. We define an *operation* as a set of computations or actions required to achieve a goal that the organization needs to accomplish to reach a desired state. Operations can result in computations being made (e.g. computing an optimal path for a swarm of UAVs) or actions being performed (e.g. neutralizing an enemy target). The actual goals that need to be achieved for an operation depend on specific implementations. Each operation specifies a *connector* that is used to connect the service to another organization. To enable this connection process, we require organizations to expose standard interfaces called *connection points*.

A *connection point* is represented by a goal-role pair from an organization and can be used to *provide* or *use* operations. We say that *a connection point provides an operation* whenever the execution of its goal triggers the execution of others goals, resulting in the achievement of the operation. In this case, it is called an *entry connection point* and its goal and role components are called *entry goal* and *entry role* respectively. Likewise, we say that *a connection point uses an operation* if its goal requires the execution of the operation in order to be completed. It is then called an *exit connection point* and its goal and role components are called the *exit goal* and *exit role* respectively.

In our approach, *connectors* provide the "glue" allowing one organization to use the operation of a second organization. A connector consists of *a request event* and an *interaction protocol*. R*equest events* are used to invoke operations whereas *interaction protocols* specify the required interactions between operation consumers and providers. Hence, connectors supply all the necessary information to bind compatible connection points.

3 COMPOSITION OF ORGANIZATIONS

Composition is a design-time process that binds a consumer organization with a provider in order to create a single composite organization. This process is illustrated in Figure 2. Given an operation, the composition process connects the *exit connection point* of a consumer with the *entry connection point* of a provider using the operation's *connector*. This interconnection ensures that the *exit goal* from the consumer organization can trigger (via the *request event*) the initialization of the *entry goal* from the provider, thus triggering the execution of the operation. Once the operation initialized, *exit roles* and *entry roles* can interact via the *interaction protocol*. Formally, the *composition of organizations* org_1 *with* org_2 *over a connection point* cp_1 *requiring an operation op* is defined whenever cp_1 is an exit connection point from org_1 requiring *op* and org_2 exposes a connection point providing *op*. This composition is denoted $org_1 \vdash^{cp1,op} org_2$.

When designing an application, we need to iterate the composition process over adequate providers as long as the resulting composite organization still requires some operations. This iterative composition results in a standalone composite application.

4 CONCLUSION

We have presented an approach to ease the development of complex OMAS by developing reusable multiagent organizations. Current approaches only use decomposition to benefit separation of concerns during design. They lack of formalization concerning the composition of the sub-organizations, which makes such sub-organizations challenging to reuse and the resulting applications difficult to maintain. Our approach combines service-oriented principles with organizational concepts in order to provide multiagent agent system designers with predefined reusable multiagent organizations. We have described how to design such organization-based multiagent services so that they expose the appropriate interfaces in order for potential consumers to request the operations they provide. Moreover, we have described our composition process, which merges multiagent organizations into a single composite organization.

A significant advantage of our approach is the ability to compose multiagent organizations to develop a wide variety of complex applications. In addition, independent multiagent services are easily modifiable, offer an outstanding approach to reusing design models, help reduce development time, and provide better structuring of large, complex multiagent systems. Our approach also increased design flexibility since service providers can easily be replaced with little or no change in the core organization.

5 REFERENCES

[1] S.A. DeLoach, W.H. Oyenan, and E. Matson, *A capabilities-based model for adaptive organizations.* Autonomous Agents and Multi-Agent Systems, 2008. **16**(1): p. 13-56.

[2] A. Estefania, J. Vicente, and B. Vicente, *Multi-Agent System Development Based on Organizations.* Electronic Notes in Theoretical Computer Science, 2006. **150**(3): p. 55-71.

[3] J. Ferber, O. Gutknecht, and F. Michel, *From Agents to Organizations: An Organizational View of Multi-agent Systems*, in Agent-Oriented Software Engineering IV. 2004. p. 443-459.

[4] J.C. Garcia-Ojeda, et al. *O-MaSE: A Customizable Approach to Developing Multiagent Development Processes*, in 8th International Workshop on Agent Oriented Software Engineering 2007.

[5] F. Zambonelli, N.R. Jennings, and M. Wooldridge, *Developing multiagent systems: The Gaia methodology.* ACM Transaction. Software Engineering Methodolologies. 2003. **12**(3): p. 317-370.