OrgMAP: An Organization-based Approach for Multi-Agent Programming (Extended Abstract)

Cuiyun Hu
School of Computer, National University of Defense Technology, China 410073
hcy56316@163.com

Xinjun Mao
School of Computer, National University of Defense Technology, China 410073
mao.xinjun@gmail.com

YinChen
School of Computer, National University of Defense Technology, China 410073
yinchen@163.com

Huiping Zhou
School of Computer, National University of Defense Technology, China 410073
icent@qq.com

ABSTRACT

This paper proposes a new organization-based multi-agent programming (OrgMAP) approach to constructing dynamic and flexible software systems. A computational and programming model named Oragent is defined following software engineering principles such as modularity, reusability and etc. Oragent model not only allows programmers to represent the systems with high-level abstractions in terms of organizations, rules, protocols and roles, but also provides a number of mechanisms, such as encapsulation, inheritance, enactment and event, to improve the dynamics and flexibility of MAS.

Categories and Subject Descriptors

D.3 [**Programming Language**]: Miscellaneous

General Terms

Languages, Theory

Keywords

MAS programming, Organization theory, Organization-oriented programming

1. INTRODUCTION

With organizational concepts, agent-oriented software engineering (AOSE) provides a natural representation for complex software systems. Recently a variety of organization-oriented approaches to multi-agent systems engineering have been brought forth including modeling approaches, methodologies, infrastructures and programming languages[1][2][3].While organization metaphor has made significant contributions to analysis and design multi-agent system (MAS), when it comes to implementation, the fact is that current MASs are usually implemented as a set of agents in terms of mental concepts, where information about organization structure and collective behavior is lost [4][5]. As a result, programmers have to manually translate and incorporate the organizational concept from design model to metal concepts, which leads to poor engineering practice and hinders engineers to exploit the full potentials of AOSE.

A recent trend in the AOSE is to employ organization concepts in the implementation of MAS [2][3]. However, until now, few languages have explicitly provided primitives for the organizational concepts. Current researches in this field usually focus on the normative MAS with the aims to deal with the openness and heterogeneity [1], but inadequate in handling dynamics and flexibility. This paper proposes a new organization-based multi-agent programming (OrgMAP) approach with a computational and programming model named Oragent, which allows programmers to constructs the systems with first-class organizational concepts, such as organizations, roles, protocols and rules. In addition, mechanisms supporting dynamics and flexibility are proposed.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

2. ORGANIZATION-BASED MULTI-AGENT PROGRAMMING

2.1 Oragent Models

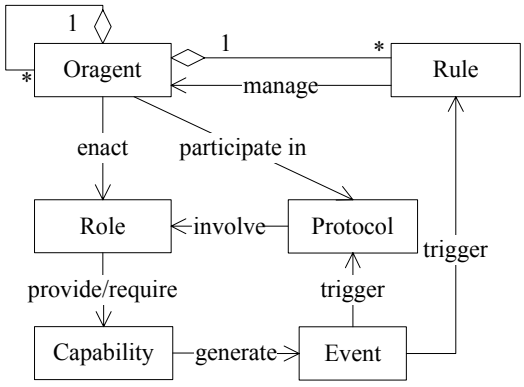

Figure 1. Oragent computational model.

Figure 1 shows the computational model of an oragent. Oragents are self-managed entities that can be either collective agency or individual agents. In the oragent model, an OrgMAP application is modeled with oragents that interact with each other with protocols based on roles. Each oragent can have a set of rules to

manage its structure and behavior, enacting roles that enable it to interact with others base on protocols and have some member oragents. The capabilities of an oragent are organized with a delegation mechanism, i.e. an oragent can delegate the implementation of its capabilities to its members. The execution of capabilities will generate a set of events which can trigger the protocols or rules. The protocols enable the involved role players (i.e. oragents) to interact with each other. With the rules, each oragent can manage the transformation of the roles it enacts and can also reorganize its structure.

An Oragent programming model is defined to describe the main programming facilities. Firstly, a set of oragents with the same structure and behavior can be specified as either an organization (for composite agency) or a role (for individual agent). Organizations are composed of four parts: state variable declarations, roles, interaction protocols and management rules. Secondly, each role defines the capabilities that contribute to its organization. It can either declare the capabilities required from its players or define the capabilities provided to its players. The roles with required capabilities are considered as abstract roles, and the organizations that enact them should provide the required capabilities. Thirdly, protocols describe interaction patterns as a sequence of messages based on roles. Each protocol can subscribe the events that are published by its involved roles. Finally, the management rules in an organization define how an oragent reorganize its structure based on the event from its members or context. In additions adaptation rules are defined within roles to describe how the oragent transfers its roles on events.

2.2 Mechanisms

This section introduces several programming mechanisms to support the construction and execution of an Oragent program, including encapsulation, inheritance, enactment and event.

Encapsulation is an important mechanism for information hiding in programming. To support the self-management, oragents encapsulate state, actions, behavior and management, so that they are able to decide how to construct and regulate their structures.

Inheritance is an important mechanism in OO programming for reuse. Similarly, the Oragent model also supports inheritance among roles, so that the sub-roles have the properties and capabilities of their super-roles. Furthermore, in the Oragent model if the roles have a common super-role, they are said to be sibling. The sibling roles are allowed to transfer from one to another according to the rules defined within their common super-role at run-time. Therefore, not only the properties and capabilities can be reused between a role and its sub-roles, but the state of oragents can also be reused among sibling roles.

Enactment is a new mechanism introduced in AOP by Mehdi Dastani in [3] with the aim to capture role dynamics. However, in the Oragent model, enactment mechanism describes the relationship between oragents and roles, contrast to the instantiation mechanism in OO. While an object has to adhere to one class that cannot be changed once it is instantiated, an oragent can possess multiple roles that can be changed dynamically during its lifetime. With the enactment mechanism, the state of the oragents can be reused during the role transformation.

Event mechanism enables the self-management of oragents. In Oragent model, each oragent owns an event management center that allows its content and context roles publish and subscribe events and a logic reasoning engine that takes as input the published events, together with programmer-defined management rules that map each event to the corresponding adaptation or management behavior. The output, then, corresponds to the published event provided with a sequence of actions including enacting, deacting, activating and deactivating of a given role and initiation, termination and regulation of a given protocol.

3. CONCLUSIONS AND FUTURE WORK

This paper proposes a new programming approach—OrgMAP and the core computational and programming model is provided as Oragent model. The Oragent model allows programmers to explicitly represent the structure of the system with high-level organizational abstraction so that the gap between design and implementation is bridged. Moreover, a set of mechanisms such as encapsulation, inheritance, enactment and event, are provided to facility the development of dynamic systems. Finally, from the software engineering perspectives, the Oragent model provides more high-level reusable entities, such as organizations, roles.

In order to implement the Oragent model, we are currently working on the programming language adhering to the Oragent model as well as a platform to execute the Oragent programs. Moreover, a programming methodology will be designed to guide the developers to code and deploy the OrgMAP programs.

4. ACKNOWLEDGMENTS

Acknowledge the financial support from Natural Science Foundation of China under granted number 61070034, 91024030 and 61133001, Program for New Century Excellent Talents in University, National Grand Fundamental Research 973 Program of China under granted number 2011CB302601.

5. REFERENCES

[1] Tinnemeier, N.A.M. 2011. Organizing Agent Organizations: Syntax and Operational Semantics of an Organization-Oriented Programming Language. SIKS Dissertation Series 2011(2), Utrecht University.

[2] Olivier Boissier, Jomi Fred Hübner, and Jaime Simão Sichman. 2007. Organization Oriented Programming: From Closed to Open Organizations G. In Proceeding of ESAW 2006. Springer Press, Heidelberg, 86–105.

[3] Dastani, M.M., van Riemsdijk, M.B., Hulstijn, J., Dignum, F.P.M., Meyer, J.-J.C. 2005. Enacting and deacting roles in agent programming. In Proceedings of AOSE 2004. Springer Press, Heidelberg, 189-204.

[4] Rafael H. Bordini, Mehdi Dastani, and Michael Winikoff. 2007. Current Issues in Multi-Agent Systems Development. In Proceedings of the Seventh Annual International Workshop on Engineering Societies in the Agents World. ESAW '07. 38-61.

[5] Nick Tinnemeier, Mehdi Dastani, John-Jules Meyer. 2009. Roles and Norms for Programming Agent Organization. In Proceedings of AAMAS 2009. 121-128.