

Enhancing Agent-Oriented Models with Aspects

Alessandro Garcia
Computing Dept, Lancaster Univ.
South Drive, InfoLab21
Lancaster, LA14WA, UK
garciaa@comp.lancs.ac.uk

Christina Chavez
Computing Dept, UFBA
Av. Adhemar de Barros, s/n
Salvador/BA, 40170-110, Brazil
flach@dcc.ufba.br

Ricardo Choren
SE/9, IME/RJ
Pça General Tibúrcio 80
Rio de Janeiro/RJ, 22290-270, Brazil
choren@de9.ime.eb.br

ABSTRACT

The modular representation of concerns is imperative in the design modeling of multi-agent systems (MASs). However, MAS designers typically face a number of concerns that inherently affect several system agents and their respective internal elements, such as actions and goals. These crosscutting concerns encompass both internal and systemic properties, such as learning, mobility, error handling, and security. Without an explicit modeling of such MAS properties, designers can not properly communicate and reason about them and their broadly-scoped effects. The result is a multitude of agent-oriented design breakdowns: poor traceability from requirements to design models, hindering of reuse opportunities, and decreased evolvability. This paper presents an enhancement of an existing agent-oriented modeling language with aspects. We illustrate the use of our approach through some classical examples of crosscutting concerns in agent-oriented design models.

Keywords

Agent-oriented software engineering, aspect-oriented software development, modeling languages, agent-oriented design.

1. INTRODUCTION

Several systemic and internal agent properties in the design of a multi-agent system (MAS) are typically *crosscutting concerns* [1, 5, 6, 8], such as learning, code mobility, context-awareness, error handling, and security. These concerns are MAS properties that involve and affect a consistent number of goals and actions, which are scattered and tangled to each other over an agent-oriented design. For example, the learning concern is composed of a number of specific goals and actions, which crosscut the goal hierarchies and actions' descriptions associated with several agents in an application. These crosscutting concerns often are fundamental driving factors in the software development process due to their broadly-scoped influence in the MAS design. Hence software engineers need to be empowered with modular representation of these concerns in order to properly communicate and reason about them and their crosscutting design implications.

Due to the very nature of some MAS concerns, existing agent-oriented modeling languages [2, 7, 9, 11] are not able to provide explicit support for the separation of crosscutting MAS-related concerns. As a result, their crosscutting effects are inevitably spanned over the resulting agent-oriented design artifacts. These

languages provide the basis to separately represent several MAS functionalities. However, they are not aimed at supporting the handling of crosscutting MAS concerns as separate "aspects". Hence these concerns cannot be modularized with the abstractions and composition mechanisms available. In fact, the need for explicitly modeling crosscutting concerns seems evident in an age where we are facing a growing number of modeling extensions with the sole purpose of coping with a specific MAS concern, such as mobility, security, and autonomy. However, each of these approaches addresses a specific concern by extending a modeling language.

In this context, this paper presents aspect-oriented enhancements of an agent-oriented modeling language, called ANote[3], in order to cope with crosscutting concerns and promote MAS designs with superior modularity. The enhancements blind aspectual abstractions [2] with fundamental agent-oriented design elements, such as agents, goals and actions. We introduce a new notation element, called *aspect*, and additional composition mechanisms to promote improved separation of MAS concerns. An aspect is a first-class, nameable entity that provides modular representation for a crosscutting concern and localizes both (a) the specification of (sets of) join points, and (b) the enhancements to be combined at the specified join points. The enhancements may add new structure and behavior to agents, goals and actions, refine or replace existing behavior.

The paper is organized as follows. Section 2 illustrates some crosscutting concerns in agent-oriented modeling in terms of an example. Section 3 presents our proposed enhancements to ANote, and presents an aspectual notation compliant to this language. Section 4 presents concluding remarks and future work.

2. CROSSCUTTING CONCERNS IN AGENT-ORIENTED MODELING

Figure 1 shows a partial design representation of a multi-agent system [4] specified with the ANote modeling language [3]. ANote defines modeling views that allow for the expression of a MAS design from different perspectives. For instance, it includes an agent view (Figure 1a), a goal view (Figure 1b), and a scenario view (Figure 1c), which are recurring perspectives in several modeling languages [1]. They support the representation of agents, actions, and goals respectively. Modeling agent systems often includes other system concerns and views, such as interaction and organizational, which have been omitted from Figure 1 due to space limitation and simplicity purposes.

The system modeled in Figure 1 is a multi-agent application that supports the management of the reviewing process for research conferences. This system will be herein referred to as Expert Committee (EC). The EC system encompasses three types of

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005...\$5.00.

software agents (Figure 1a): (i) information agents, (ii) user agents, and (iii) the manager agent. Figures 1b and 1c illustrate some goals and actions associated with the information and user agents respectively. Information agents' goals are managing the system information that is mainly stored in a database, and providing information to the other system agents and users as requested. User agents are software assistants that represent system users in reviewing processes. Their basic functionalities are to infer and to keep information about the users' research interests and participations in scientific events.

We can notice in Figure 1 that there are some concerns which cannot be modularly represented across the design views. Learning and mobility are two examples of crosscutting concerns in the agent-oriented design of the EC system. The set of goals and actions, which are both part of and affected by these broadly-scoped concerns, are spread over the agent-oriented design. Each set of goals and actions for a specific crosscutting concern is surrounded by dotted rectangles in Figure 1. The mobility concern

consists of goals that crosscut distinct goal hierarchies, and of actions intermingled with diverse scenarios.

The crosscutting manifestation leads to two major problems at the agent-oriented design level, the so-called scattering and tangling [10]. *Scattering* in agent-oriented models is the manifestation of design elements, which belong to one specific concern, over several modeling units referred to other MAS concerns. *Tangling* in agent-oriented models is the mix of multiple concerns together in the same modeling elements

Existing agent-oriented modeling abstractions and composition mechanisms are not aimed at supporting the isolate handling of crosscutting MAS concerns as exemplified in Figure 1. It brings in turn a number of substantial design pitfalls, such as: hindering of modular and compositional reasoning; replication of agent-oriented design elements; essential information loosing; and, reduced evolvability and reuse opportunities

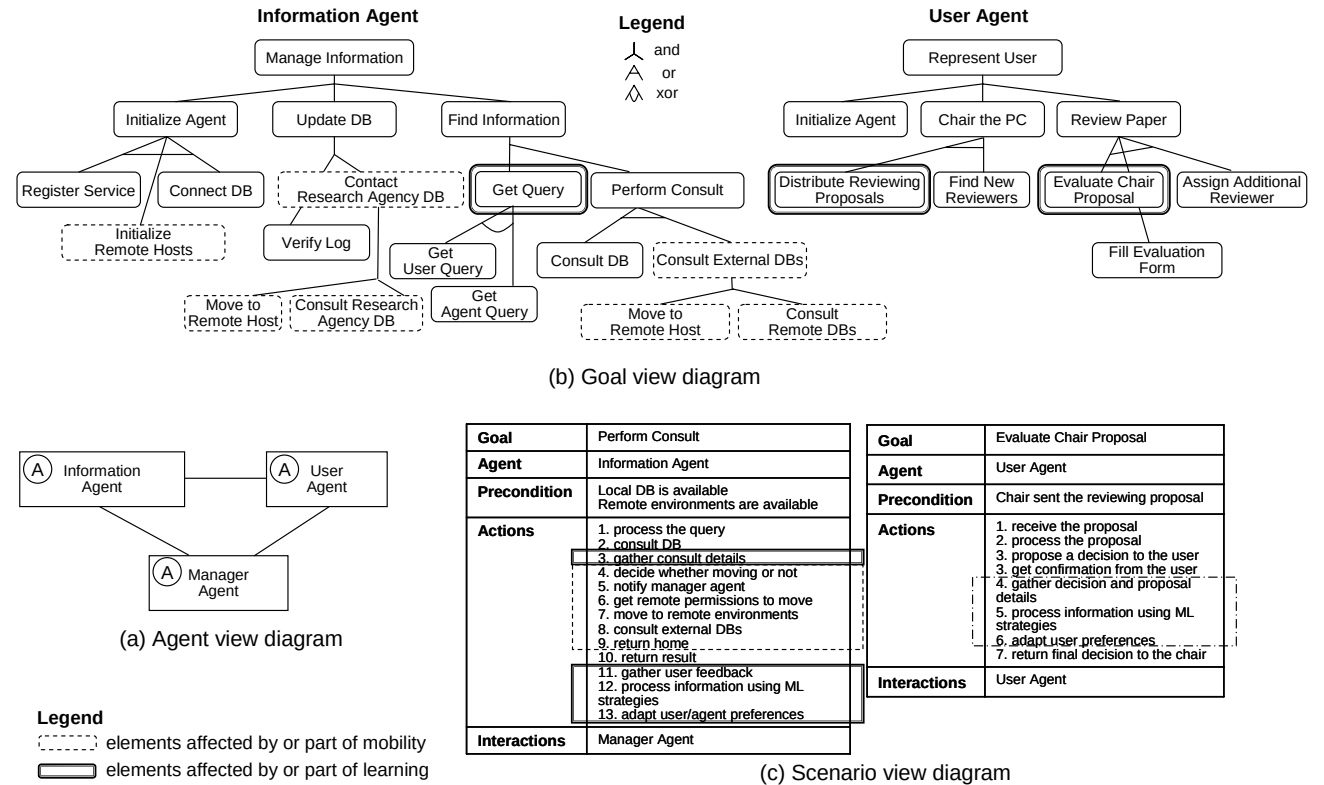


Figure 1. Crosscutting Concerns in ANote Modeling Views.

3. AN ASPECT-ORIENTED NOTATION

This section describes our approach to support agent-oriented modeling enhanced with aspects. We show the use of our notation in the context of the ANote modeling language [3]. ANote has 7 diagrams divided into three packages. In this paper, we will focus on the structural package, which encompasses the Goal, Agent and Planning diagrams.

The goal view diagram presents the system goals, i.e. the agent goals (Figure 2). This diagram has been enhanced with an aspect-oriented notation to present how the goals of aspects influence the

agent goals at well-defined join points. The extension of the goal view diagram notation has been based on and adapted from our aspect-oriented notation for object-oriented design proposed in [2]. An aspect is represented as a diamond. Some circles are attached to it. Each circle captures a way (interface) that an aspect crosscuts the agent goals.

Figure 2 shows that the Mobility aspect has an interface "IMobility" that introduces the goals "Move to Remote Host" and "Consult Remote DB" to the goals "Consult Research Agency DB" and "Consult External DB" of Information Agents. In this

case, the composition operation “introduce” (Table 1) is used to denote an aspect-agent crosscutting.

The agent view specifies the agent types that exist in a multi-agent application solution and their relationships. In ANote, an agent is the module that is able to perform actions. However, aspects also contain a number of actions that implement the crosscutting behaviors, thus becoming system modules. Figure 3 shows how the agent diagram has been enhanced with aspect-oriented notation. Here, the aspects are also represented as diamonds. The

arrows represent the crosscutting relationship. We can see in Figure 3 that Information Agents and User Agents are influenced by the Learning aspect, while the Information Agent is also affected by the Mobility aspect.

The ANote dynamic package encompasses the scenario, interaction and planning diagrams that have also been enhanced with aspects, but for brevity reasons, are not presented here.

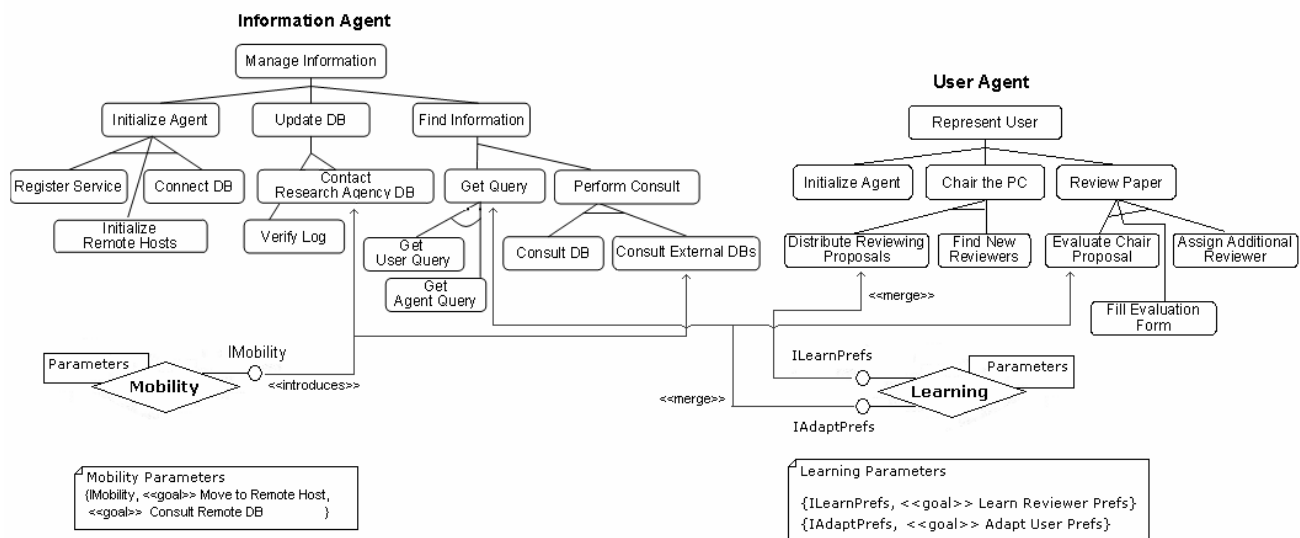


Figure 2. Aspect enhanced goal diagram.

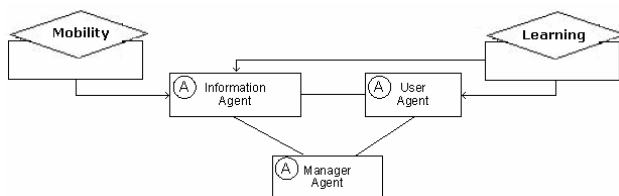


Figure 3. Aspect enhanced agent diagram.

4. CONCLUSIONS AND FUTURE WORK

This paper is a first attempt to systematically tame broadly-scoped concerns in agent-oriented design modeling. Many internal agent properties and systemic properties in the design of MAS are typically crosscutting, and they need to be handled as such. No matter what kind of decomposition and abstractions the agent-based software developers are relying on; there are always MAS concerns that crosscut the boundaries of other concerns. The lack of modularization support for crosscutting concerns generates undesirable burdens on agent-oriented design reuse and evolution.

This paper presented an aspect-oriented notation and integrated it into the ANote language. Our work is a first step towards enhancing agent-oriented models with aspects and we cannot guarantee that our set of diagrams is necessarily complete; more case studies are still necessary. Another issue is the behavioral modeling of agents, i.e. how to model crosscutting actions. We are currently working on enhancing the ANote planning view diagram to properly address other action-action composition categories.

5. REFERENCES

- [1] Bergenti, F., Gleizes, M.-P., Zambonelli, F. (Eds.). Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook, volume 11. Springer-Verlag, 2004.
- [2] Chavez, C., Lucena, C. A Metamodel for Aspect-Oriented Modeling. Proc. Workshop on Aspect-oriented Modeling with UML, 1st Intl Conf. on Aspect-Oriented Software Development, 2002.
- [3] Choren, R., Lucena, C. Modeling Multi-agent Systems with ANote. Software Systems Modeling Journal 4, 2005, p. 199-208.
- [4] Garcia, A. From Objects to Agents: An Aspect-Oriented Approach. PhD Thesis, Computer Science Dept., PUC-Rio, Brazil, April 2004.
- [5] Garcia, A., Lucena, C., Cowan, D. Agents in Object-Oriented Software Engineering. Software: Pract and Experience 34(3), 2004, p. 489-521.
- [6] Garcia, A. et al. Aspects in Agent-Oriented Software Engineering: Lessons Learned. Proc. 6th Workshop on Agent-Oriented on Software Engineering, AAMAS 2005, July 2005.
- [7] Odell, J., Parunak, H., Bauer, B. Extending UML for Agents. In: Proceedings of the Agent-Oriented Information Systems Workshop, 17th Conf. on Artificial Intelligence (AAAI 2000), p. 3-17, 2000.
- [8] Pace, A., Trilnik, F., Campo, M. Assisting the Development of Aspect-based MAS using the SmartWeaver Approach. In: Software Engineering for MASs, LNCS 2603, Springer, p. 165-181, 2003.
- [9] Silva, V., Lucena, C. From a Conceptual Framework for Agents and Objects to a Multi-Agent System Modeling Language. Journal of Autonomous Agents and Multi-Agent Systems 9(1-2), 2004.
- [10] Tarr, P. et al. N Degrees of Separation: Multi-Dimensional Separation of Concerns. Proc. 21st ICSE 1999, p. 107-119, 1999.
- [11] Wooldridge, M., Jennings, N., Kinny, D. The GaiaMethodology for Agent-oriented Analysis and Design. Autonomous Agents and Multi-Agent Systems 3(3), 2000, p. 285-312.