

# An Aspect-Oriented Modeling Framework for Designing Multi-Agent Systems

Alessandro Garcia<sup>1</sup>, Christina Chavez<sup>2</sup>, and Ricardo Choren<sup>3</sup>

<sup>1</sup> Computing Department, Lancaster University - UK  
garciaa@comp.lancs.ac.uk

<sup>2</sup> Computing Department, Federal University of Bahia - Brazil  
flach@dcc.ufba.br

<sup>3</sup> Computer Engineering Department, Military Institute of Engineering - Brazil  
choren@de9.ime.eb.br

**Abstract.** A number of concerns in multi-agent system (MAS) design have a crosscutting impact on agent-oriented models. These concerns inherently affect several system agents and their internal modeling elements, such as actions and goals. Examples of crosscutting concerns in MAS design 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. This paper presents a meta-modeling framework for supporting the modular representation of crosscutting concerns in agent-oriented design. The framework is centered on the notion of aspects to describe these concerns. It also defines new composition operators to enable the specification on how aspects affect the agent goals and actions. The proposed framework is a result of our previous experience in both using aspect-oriented techniques for MAS design and implementation, and integrating aspect-oriented abstractions in an agent-oriented modeling language, called ANote.

## 1 Introduction

MAS developers usually face a number of concerns which have a crosscutting impact on agent-oriented design artifacts, such as goal models and agent models [2, 13, 14, 23, 28, 16]. A crosscutting concern at MAS design is any concern that cannot be modularly captured with the conventional agent-oriented modeling abstractions and composition mechanisms [16]. 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. Other examples of crosscutting MAS concerns are broadly-scoped properties, such as mobility, error handling, and security. These concerns consistently cut across the modularity of several MAS modeling elements, such as agents, goals, actions, and plans. Very often, the crosscutting property of such design concerns remains either implicit or is described in informal ways leading to reduced uniformity, impeding traceability between higher-level models and implementations, and hindering detailed design and implementation decisions. Hence there is a pressing need for the conception of a modeling framework that provides MAS designers with proper support for the modular representation and reasoning [19] of crosscutting MAS concerns.

In fact, there is a growing number of modeling extensions dealing with crosscutting concerns in multi-agent systems [6, 17, 20, 21, 29]. However, each of them supports one

specific agency concern, such as mobility or autonomy. It makes more evident the inadequacy of conventional agent-oriented design languages [7, 22, 26, 30] and respective abstractions to cope with the crosscutting nature of some MAS concerns. As a result, their crosscutting effects are inevitably spanned over the resulting agent-oriented design artifacts. To encompass all crosscutting MAS concerns, it is important to describe generic abstractions and to devise new composition rules.

This paper presents a meta-modeling framework to enable the modular representation and composition of broadly-scoped concerns in agent-oriented design modeling. Our framework blends core concepts from Aspect-Oriented Software Development (AOSD) [1, 11, 18] with recurring abstractions of agent-oriented design. AOSD is an emerging development approach aimed at promoting improved separation of concerns by introducing a new modular unit, called *aspect*. Although the notion of aspects looks promising for handling crosscutting MAS concerns in early phases of the software life-cycle, the existing aspect-oriented modeling approaches have been limited to the object-oriented and component-oriented paradigms. Our previous work has investigated the interplay of AOSD and agent-oriented software engineering, but it has focused on different contexts other than agent-oriented design and modeling, including architecture design [15, 23], detailed design [12, 13], and implementation [9, 23, 28]. In other recent work [16], we have enhanced ANote, a specific agent-oriented modeling language, with aspect-oriented notation.

The paper is organized as follows. Section 2 illustrates crosscutting concerns in agent-oriented modeling in terms of an example, and shows the inability of existing abstractions and composition rules to support their separation. Section 3 presents our aspect-oriented framework for agent-oriented design and modeling. Section 4 describes the applicability of the concept of aspects in agent-oriented design according to our approach. Section 5 discusses related work. Section 6 presents concluding remarks.

## 2 Crosscutting Concerns in Agent-Oriented Goal Modeling

This section presents some examples of crosscutting concerns in agent-oriented design modeling. Section 2.1 presents the main concerns associated with our running example. Section 2.2 illustrates and discusses some typical examples of crosscutting concerns in MASs in terms of our case study. Section 2.3 shows the consequences of not having explicit support for the modularization of crosscutting concerns in agent-oriented design models.

### 2.1 The Expert Committee Example

The Expert Committee (EC) is a multi-agent application that supports the management of the reviewing process for research conferences. The EC system encompasses three types of software agents: (i) information agents, (ii) user agents, and (iii) the manager agent. Figure 1 shows a partial design representation for the EC system [13] specified with the ANote modeling language [7]. ANote defines modeling views that allow for the expression of a MAS design from seven different perspectives: goal, agent, environment, scenario, action, interaction and organization. For instance, the goal view diagram (Figure 1A) provides the identification of the hierarchy that outlines the system goals. In this diagram, complex goals can be functionally decomposed into more granular goals

until the designer reaches the desired level of goals (functionalities) to distribute among the agents. These two diagrams support the representation of the agents, goals and contexts respectively. A context is a scenario that indicates how agents should achieve goals.

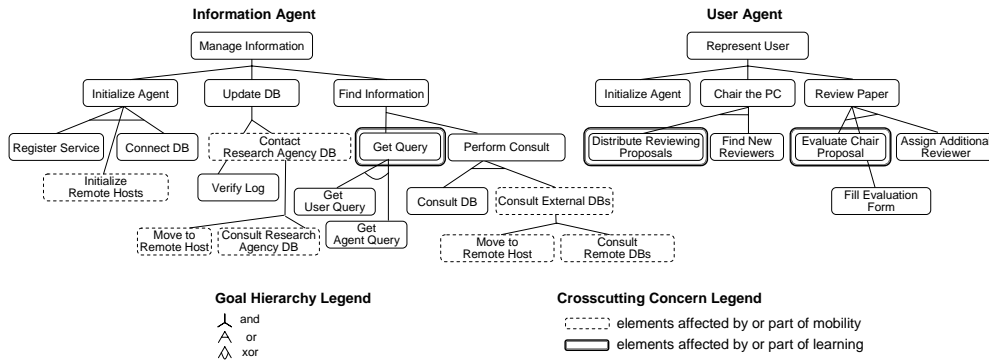


Fig. 1A. Crosscutting Concerns in ANote Goal Diagram [16]

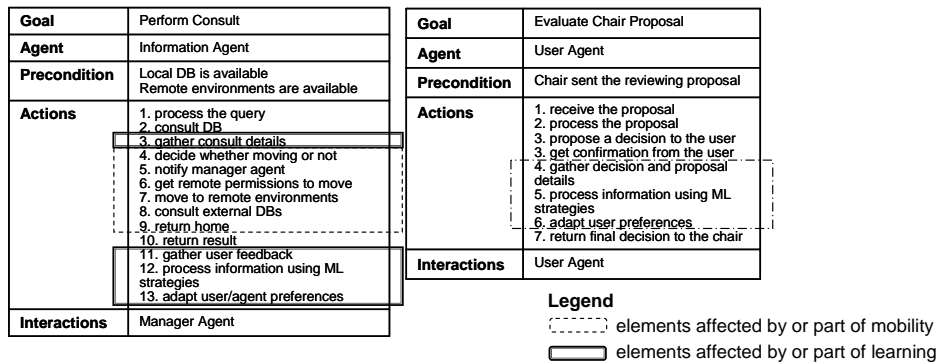


Fig. 1B. Crosscutting Concerns in ANote Scenario Diagrams [16]

Figure 1A illustrates some goals 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.

## 2.2 Modularity Issues in Goal Models

Designers are able to successfully use agent-oriented abstractions and composition mechanisms to represent several system concerns in a modular fashion. For example, the goal composition rules *and*, *or* and *xor* allow for consistently specifying how goals and sub-goals are combined to realize a given concern. The "finding information" and "reviewing papers" concerns, for instance, can be smoothly captured in separate goal hierarchies, as illustrated in Figure 1A.

However, we can notice in Figures 1A and 1B that there are some concerns which cannot be represented in a modular way 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, which is part of and affected by these broadly-scoped concerns, is spread over the agent-oriented design. The set of goals 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. A similar problem happens with the learning-specific goals and actions. These two concerns have a huge impact on the agent structure and behavior since they cut through the primary modularity of goal hierarchies representing other agent concerns.

The crosscutting manifestation leads to two major problems at the agent-oriented design level: scattering and tangling [27]. Scattering in agent-oriented models is the manifestation of design elements that belong to one specific concern, over several modeling units referred to other MAS concerns. For example, the mobility-related goals are scattered over multiple goal hierarchies, such as the ones under the goals “Perform Consult” and “Update DB” (Figure 1A). Tangling in agent-oriented models is the mix of multiple concerns together in the same modeling elements. For instance, tangling is evident in the “Perform Consult” scenario since it is realizing mobility-related and learning-related actions in addition to its primary actions of performing the consult.

### 2.3 Side Effects on MAS Design Modularity

Existing agent-oriented modeling abstractions and composition mechanisms do not provide proper support to isolate crosscutting MAS concerns as exemplified in Figure 1. This brings a number of substantial design pitfalls, as described below.

*Hindering of modular and compositional reasoning.* Tangling and scattering of MAS concerns hinder both modular and compositional reasoning at the design stage. Developers are unable to reason about a concern while looking only at its description, including its core goals and actions, and its structural and behavioral implications in terms of other MAS concerns. Hence its analysis inevitably forces developers to consider all the design artifacts in an ad hoc manner. For example, the designers treating the learning and mobility concerns in Figure 1 need to consult the goals associated with all other design concerns across the different views.

*Replication of agent-oriented design elements.* Replication of goals (and their actions) in the agent-oriented design is another side effect of tangling and scattering. Replication in turn decreases the system understandability, reusability and evolvability. For instance, the mobility-specific goal “Move to Remote Host” is duplicated in the goal view diagrams due to its crosscutting relationships with the goals “Update DB” and “Find Information”.

*Essential information missing.* Without appropriate abstractions and composition mechanisms for crosscutting concerns in agent-oriented models, MAS designers are not able to locally express the structural and behavioral implications of a given broadly-scoped MAS concern in several design elements and views. The result is that design information is irrecoverable just because the lack of support for properly specifying them. For example, the learning concern (Figure 1B) would clearly have a goal “Learn User Preferences” associated with it, which influences at least two goals: “Get Query” and “Evaluate Chair Proposal”. As the designer does not have support to describe this

crosscutting impact of a learning-specific goal, such important design information has been lost and such goal is not appearing in the design models. Even if the designers use the standard abstractions and composition rules of the underlying modeling languages to register such information, the learning-specific goal would end up being scattered over two or more goal hierarchies.

*Reduced evolvability and reuse opportunities.* Tangling and scattering are two of the main anti-reuse and anti-evolution factors in the MAS software lifecycle. For example, it is not easy to understand, in the EC system design, which agent types are mobile and which have learning abilities. In addition, the goals associated with learning and mobility concerns are not coherently documented in a single modular unit so that they can be easily reused in other design contexts. Also, if the designers need to evolve the system and introduce changes related to the mobility and learning properties, the evolution process will be cumbersome as those concerns are intermingled in the system design.

### 3 An Aspect-Oriented Modeling Framework for MAS Design

This section describes our approach to address the need for supporting the modular representation of crosscutting concerns in agent-oriented design. We present a meta-modeling framework that enriches agent-oriented models with aspects, which are the design first-class units to overcome tangling and scattering of concerns. The proposed framework is a result of our previous experience in both using aspect-oriented techniques for MAS design and implementation [12, 13, 23, 28], and integrating aspect-oriented abstractions in an agent-oriented modeling language, called ANote [16]. The proposed framework is independent of specific agent-oriented modeling languages, and is composed of 3 models: the Agent Model, the Aspect Model and the Composition Model. We use entity-relationship diagrams to illustrate each conceptual model in terms of entity sets and relations over these sets.

**The Agent Model.** The *Agent Model* is a conceptual meta-model for agent-based system modeling. The model presented here comprises a set of fundamental agent-oriented design elements, such as agents, actions, and goals. Figure 2 summarizes the agent model.

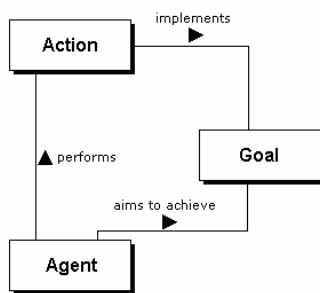


Fig. 2. The Agent Model

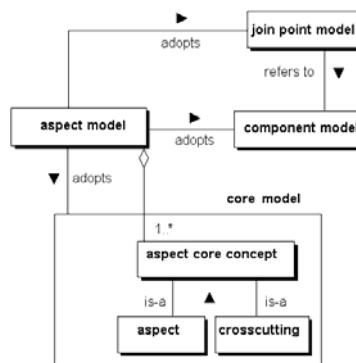


Fig. 3. The Aspect Model

An Agent is the module that is able to perform actions; it is the main abstraction of the agent paradigm. An action is a computation that results in a change in the state of an agent. An agent acts in the system in order to achieve a goal. While executing actions, an agent can interact with other agents. A goal is a system objective, and it defines a state that must be achieved by one or more agents. The execution of one or more actions allows for the achievement of a goal. A goal can be of different categories that are organized into a specialization hierarchy. This means that goals can be decomposed into several alternative combinations of sub-goals.

**The Aspect Model.** The *Aspect Model* is a conceptual framework for AOSD [4, 5] that subsumes concepts, relationships and properties for supporting the design of aspect-oriented modeling languages. These elements are organized around three interrelated conceptual models: (i) the component model, (ii) the join point model and (iii) the core model (Figure 3). As a generic conceptual framework, the aspect model needs to be instantiated in order to be used [4]. This means that the designer of a new aspect-oriented language must adopt a component model, a suitable joint point model, and appropriate structure and semantics for the core model concepts (aspect and crosscutting).

In our meta-modeling framework, the adopted component model is the agent model described above and join points are elements related to the structure or the behavior of agent models, referenced and possibly affected by an aspect. The join point model represents a conceptual framework that describes the kinds of join points of interest and the associated restrictions for their use. Agents, Goals and Actions are defined as join points, that is, locations in agent models that can be affected by aspects.

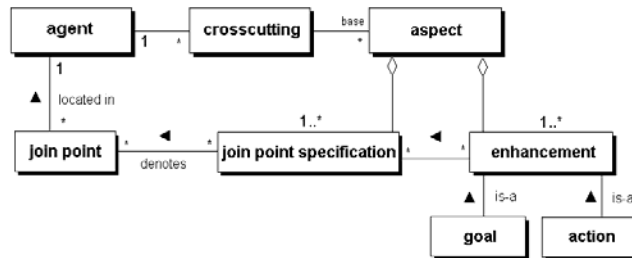
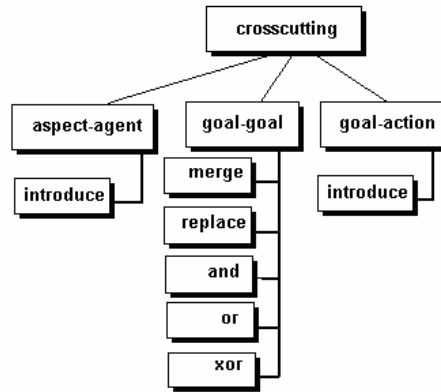


Fig. 4. An Aspect Model for an Agent Model

The core model represents a conceptual framework used for describing aspects and crosscutting. 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 kinds of enhancements depend on the kind of component model adopted. For the agent model, enhancements are goal-like and action-like elements; aspects modularize crosscutting goals and crosscutting actions.

Crosscutting denotes the generic composition mechanism used to compose aspects and agents, enhancing them at the designated join points (Figure 4). Aspects crosscut one or more agents, possibly affecting their structure and behavior.

**The Composition Model.** The *Composition Model* is a conceptual framework that provides semantics description of the crosscutting composition mechanism. In other words, this model characterizes the possible ways aspects may affect agents, their goals, and actions. Currently, our composition model organizes crosscutting operations according to three categories or dimensions (Figure 5): aspect-agent, goal-goal and goal-action composition. Table 1 presents the semantics description and the possible crosscutting operations for each composition category.



**Fig. 5.** Crosscutting dimensions for the Agent Model

Aspect-Agent crosscutting composition basically adds a new goal  $G$  to Agent  $A$ . For example, the mobility concern in the EC system design (Section 2.1) is an example of aspect that introduces the mobility-specific goals “Move to Remote Host” and “Consult Research Agency DB” in the goal hierarchy of Information Agent. Goal-Action crosscutting composition supports the addition of a new Action to some Goal  $G$ . The other two categories are further detailed in Table 2.

**Table 1.** Composition Model: crosscutting categories semantics and operations.

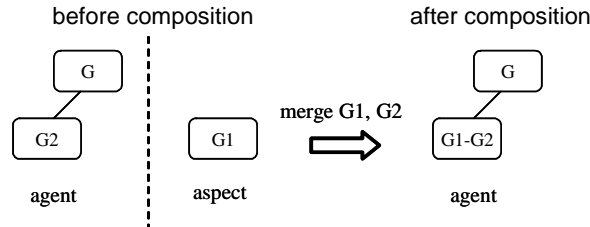
Category	Description	Operations
Aspect-Agent	Aspect $X$ introduces Goal $G$ (and related set of Actions) into Agent $A$	introduce
Goal-Goal	Goal $G_1$ of Aspect $X$ is composed with Goal $G_2$ of Agent $A$ (possibly giving a new Goal $G$ in $A$ )	merge, replace, AND, OR, XOR
Goal-Action	Aspect $X$ introduces Action $\alpha$ into Goal $G$ (of Agent $A$ )	introduce
Action-Action	Action $\alpha_1$ of Aspect $X$ is composed with Action $\alpha_2$ (of Goal $G$ , Agent $A$ )	merge-before, merge-after, replace

The Goal-Goal composition style combines an aspect goal  $G_1$  with an existing agent goal  $G_2$ . For example, the goal “Learn User Preferences” associated with the learning concern in the EC system (Section 2.1) needs to be composed with goals of Information Agent and User Agent, such as “Get Query” and “Evaluate Chair Proposal”. The “merge” operation describes well this kind of goal-goal composition since the learning aspect enhances the goals of querying the DB and evaluating the chair proposal without changing the semantics of the state to be reached (the agent goals).

**Table 2.** Crosscutting semantics for Goal-Goal and Action-Action composition.

Kind	Operation	Semantics
GOAL-GOAL CROSSCUTTING	X.G1 <b>AND</b> A.G2	Goal G1 of Aspect X is AND-ed with Goal G2 of Agent A
	X.G1 <b>OR</b> A.G2	Goal G1 of Aspect X is OR-ed with Goal G2 of Agent A
	X.G1 <b>XOR</b> A.G2	Goal G1 of Aspect X is XOR-ed with Goal G2 of Agent A
	X.G1 <b>MERGE</b> A. G2	Goal G1 of Aspect X is merged with Goal G2 of Agent A
	X.G1 <b>REPLACE</b> A. G2	Goal G1 of Aspect X replaces Goal G2 of Agent A
ACTION-ACTION CROSSCUTTING	X.α1 <b>MERGE</b> A. α2	Action α1 of Aspect X is merged before Action α2 of Agent A (within Goal G)
	A.α2 <b>MERGE</b> X.α1	Action α1 of Aspect X is merged after Action α2 of Agent A (within Goal G)
	X.α1 <b>REPLACE</b> A. α2	Action α1 of Aspect X replaces Action α2 of Agent A (within Goal G)

Figure 6 presents a diagrammatic representation of merge semantics between Goals. Note that the learning-specific goal “Learn User Preferences” has been omitted in the EC design (Figure 1b) because the lack of support of the modeling notation to express such aspectual influence of the learning concern (Section 2.3).



**Figure 6.** Goal-Goal composition: Merge.

## 4 Discussions

The Aspect Model abstracts from the kinds of component an aspect affects. In [5], the Aspect Model was instantiated with the UML object model to provide an aspect-oriented conceptual framework for object-oriented design and modeling. Furthermore, the Aspect Model can be instantiated with different agent models and sets of agent-oriented abstractions.

The applicability of the concept of aspects in agent-oriented design, and the usability of our modeling approach [16] have been evaluated in different contexts and with respect to different modeling criteria as the ones described below.

*Integrability and Extensibility.* As presented in Section 3.1, We have used the meta-modeling framework to introduce aspect-oriented capabilities in the ANote modeling

language, which was straightforward [16]. In fact, we did not experience particular conflicts while integrating the aspect-oriented meta-model and the ANote meta-model. During this integration process, we observed that additional join points were necessary to be defined due to particularities of the ANote’s agent model. For example, a certain precondition can be also part of a crosscutting behavior in ANote models.

The precondition “Remote Environments are Available” in Figure 1c is part of the mobility concern. Hence, preconditions need to be also defined as join points in the aspect-oriented meta-model for the ANote language. The accommodation of crosscutting preconditions and other join points in the extended ANote meta-model was a smooth step, as our *Aspect Model* is flexible enough. It supports this extensibility through a chain of associated meta-abstractions, such as “join point” and “enhancement”, and a comprehensive set of composition styles under the “crosscutting” concept.

*Design Knowledge Management, Evolvability, and Reusability.* We have also assessed the aspect-oriented modeling approach in three case studies. These MASs encompassed different characteristics, different degrees of complexity, and diverse domains: the Expert Committee system (Section 2.1), a system for the Trading Agent Competition [25], and a portal development management system [12, 13]. The design of these systems encompassed different crosscutting concerns in their agent-oriented models, including adaptation, mobility, learning, and proactive autonomy.

In fact, we were able to explicitly model the implications of these broadly-scoped MAS properties in the same way we model the basic MAS behaviors. This externalized better the knowledge present in the agent-oriented design process, and provided an improved basis for further evolution and reuse. For example, the learning-specific goals and its synergistic relationships with other system goals have been better captured based in our aspect-oriented notation [16]. These aspectual goals have been lost in the “non-aspectized” agent-oriented models (Section 2.3).

## 5 Related Work

Existing approaches that aim to provide an unifying conceptual framework for AOSD are tightly coupled to the object-oriented paradigm and the modularization problems related to its main abstractions and composition mechanisms. As far as we know, our work is the only aspect-oriented conceptual framework that is agnostic to the base component model and that can be easily instantiated to deal with different sets of abstractions, including agent abstractions.

Since our conceptual framework is the base for enhancing the ANote modeling language with aspect-oriented notation [16], we also discuss some related work that deals with existing modeling languages and methodologies (Section 5.1), and modeling extensions to address specific concerns (Section 5.2). The list of related work here is not intended to be exhaustive. We have focused on the ones we believe have explicit links with our aspect-oriented modeling framework. As elucidated below, what makes our approach distinct from all these approaches is the expressiveness and precision with which it allows capturing and describing crosscutting concerns at the agent-oriented design stage.

**Modeling Languages and Methodologies.** Tropos provides to some extent abstractions for expressing crosscutting concerns [3]. However, the focus is on the representation of inter-goal influences in the early and late requirements development phases. It provides

a very-high level set of abstractions (goals and soft-goals), which allow the description of positive and negative contributions between goals representing functional and non-functional MAS properties. Soft goals can be viewed as MAS aspects at the requirements level. Although the goal models could be refined in late development stages, Tropos does not provide a complete composition framework to design stages as described in our framework.

As pointed out in Section 2, crosscutting relationships naturally emerge in agent-oriented modeling beyond inter-goal relationships and, as a result, more concrete composition rules are required. In fact, we believe our agent-oriented design framework is complementary to the Tropos notations. Soft goals at the requirement levels can be easily traced backward and forward from our aspect-oriented agent models (Section 3.1)

Another approach that shows interesting aspect-oriented ideas in the modeling language is [8]. This approach focuses on representing the system-to-be according to several different perspectives; each one of them promoting an abstract representation of the system. Nevertheless, it only sketches the characteristics of an autonomy perspective for MAS specification. In addition, it does not address a comprehensive concern-independent composition framework for modularizing crosscutting structure and behavior in agent-oriented design.

**Specific Extensions.** Several other approaches try to develop extensions and notations that focus on specific concerns for MAS development, which are typically crosscutting. For example, the work on [21] reports on a secure architectural description language (ADL) for MAS. It focuses on the provision of ADL constructs to specify security issues in MAS architectures. The specification of security issues is however entangled to the core components of a MAS architecture. Another work, presented by Weiss [29], intends to capture autonomy in agent roles using a formal schema called RNS (standing for “Roles, Norms, and Sanctions”) which allows for a specification of an agent’s autonomy. It deals only with autonomy and it is mostly concerned with providing a formal schema to describe it, i.e. it is not focused on how autonomy crosscuts other agent functionalities.

Some graphical notations have also been extended to cope with specific crosscutting concerns. For instance, the work reported in [17] demonstrates how Activity Diagrams in UML 2.0 can be readily used to model dynamic behaviors of mobile agent systems and point out why they are effective for them from its underlying computational model.

In our point of view, the growing number of modeling approaches dealing with specific crosscutting concerns in agent-oriented systems denotes that there is a pressing need to define a generic aspect-oriented metamodel and associated notation that provide support for their proper specification. To encompass all crosscutting MAS concerns, it is important to describe generic abstractions and to devise composition rules, as proposed in this paper. The existing AOSD approaches have been limited to the object-oriented and component-oriented paradigms. Our previous contributions have focused in other development phases other than agent-oriented design.

## 6 Conclusions

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 mat-

ter 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. As discussed here, widely-scoped properties can bring deeper problems to the designers; they can even be scattered and tangled in more than one design view, as it is the case for the learning and mobility concerns. Because a clear separation of concerns is a main tenet in software engineering, the lack of modularization support for those concerns generates undesirable burdens on agent-oriented design reuse and evolution.

The contributions of this paper were the following. We described a comprehensive list of problems associated with the non-modularized handling of crosscutting concerns in agent-oriented modeling. To address those problems, we have proposed an aspect-oriented framework for agent-oriented modeling. Our previous work [16] is an example of aspect-oriented notation, which instantiates our proposed modeling framework by integrating it into the ANote language [8]. Since our work is a first step towards enhancing agent-oriented design models with aspects, we cannot guarantee that our set of composition operators is necessarily complete. Our goal here was to provide a core meta-modeling framework to support the central abstractions and composition mechanisms. Further case studies are necessary to evaluate the coverage degree of our composition operators.

**Acknowledgements.** Alessandro is supported by European Commission as part of the grant IST-2-004349: European Network of Excellence on Aspect-Oriented Software Development (AOSD-Europe), 2004-2008. This work has been also partially supported by CNPq-Brazil under grant No. 479395/2004-7 for Christina.

## References

1. AOSD Steering Committee: aosd.net main page. <http://aosd.net/> (2006)
2. 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, Berlin Heidelberg New York (2004)
3. Castro, J., Kolp, M., Mylopoulos, J.: Towards Requirements-Driven Information Systems Engineering: the Tropos Project. *Information Systems* 27, n. 6 (2002) 365–389
4. Chavez, C., Lucena, C.: A Theory of Aspects for Aspect-Oriented Development. In: *Proc. 17th Brazilian Symposium on Software Engineering* (2003) 130-145
5. Chavez, C.: *A Model-Driven Approach to Aspect-Oriented Design*. PhD Thesis, Computer Science Department, PUC-Rio (2004)
6. Cheong, C., Winikoff, M.: Hermes: Designing Goal-Oriented Agent Interactions. In: *Proc. of the 6th International Workshop on Agent-Oriented Software Engineering* (2005)
7. Choren, R., Lucena, C.: Modeling Multi-agent Systems with ANote. *Journal of Software and Systems Modeling* 4, n. 3 (2005) 199-208
8. Cossentino, M., Zambonelli, F.: Agent Design from the Autonomy Perspective. In: Nickles, M., Rovatsos, M., Weiss, G. (eds.): *Agents and Computational Autonomy: Potential, Risks and Solutions*. LNCS, Vol. 2969. Springer-Verlag, Berlin Heidelberg New York (2004) 140-150
9. D'Hondt, M., Gybels, K., Jonckers, V.: Seamless Integration of Rule-Based Knowledge and Object-Oriented Functionality with Linguistic Symbiosis. In: *Proc. of the 2004 ACM Symposium on Applied Computing* (2004) 1328-1335
10. Dijkstra, E.: *A Discipline of Programming*. Prentice Hall, Englewood Cliffs (1976)
11. Filman, R., Elrad, T., Clarke, S., Aksit, M.: *Aspect-Oriented Software Development*. Addison-Wesley (2004)

12. Garcia, A., Sant'Anna, C., Chavez, C., Silva, V., Lucena, C., von Staa, A.: Separation of Concerns in Multi-Agent Systems: An Empirical Study. In: "Software Engineering for Multi-Agent Systems II". LNCS, Vol. 2940. Springer-Verlag, Berlin Heidelberg New York (2004) 49-72
13. Garcia, A., Lucena, C., Cowan, D.: Agents in Object-Oriented Software Engineering. *Software: Practice and Experience* 34, n. 3 (2004) 489-521
14. Garcia, A., Kulesza, U., Sant'Anna, C., Chavez, C., Lucena, C.: Aspects in Agent-Oriented Software Engineering: Lessons Learned. In: *Proc. of the 6th International Workshop on Agent-Oriented Software Engineering* (2005)
15. Garcia, A., Kulesza, U., Lucena, C.: Aspectizing Multi-Agent Systems: From Architecture to Implementation. In: "Software Engineering for Multi-Agent Systems III". *Lecture Notes in Computer Science*, Vol. 3390. Springer-Verlag, Berlin Heidelberg New York (2005) 121-143
16. Garcia, A., Chavez, C., Choren, R.: Enhancing Agent-Oriented Models with Aspects. In: *Proc. of the 5th International Conference on Autonomous Agents and MultiAgent Systems* (2006)
17. Kang, M. et al: Modelling Mobile Agent Applications in UML 2.0 Activity Diagrams. In: *Proc. of 3rd SELMAS Workshop at ICSE 2004* (2004) 104-111
18. Kiczales, G. et al: Aspect-Oriented Programming. In: Akcsit, M., Matsuoka, S. (eds.): *Proc. European Conference on Object-Oriented Programming. Lecture Notes in Computer Science*, Vol. 1241. Springer-Verlag, Berlin Heidelberg New York (1997) 220-242
19. Kiczales, G., Mezini, M.: Aspect-Oriented Programming and Modular Reasoning. In: *Proc. of the 27th International Conference on Software Engineering* (2005) 49-58
20. Mallya, A.U., Singh, M.P.: Incorporating Commitment Protocols into Tropos. In: *Proc. of the 6th International Workshop on Agent-Oriented Software Engineering* (2005)
21. Mouratidis, H. et al: A Secure Architectural Description Language for Agent Systems. In: *Proc. of 4th Intl. Conference on Autonomous Agents and Multiagent Systems* (2005) 578-585
22. Odell, J., Parunak, H., Bauer, B.: Extending UML for Agents. In: *Proc. of the Agent-Oriented Information Systems Workshop at AAAI 2000* (2000) 3-17
23. Pace, A., Trilnik, F., Campo, M.: Assisting the Development of Aspect-based MAS using the SmartWeaver Approach. In: Garcia, A.F., Lucena, C.J.P., Zambonelli, F., Omicini, A., Castro, J. (eds.): *Software Engineering for Multi-Agent Systems. Lecture Notes in Computer Science*, Vol. 2603. Springer-Verlag, Berlin Heidelberg New York (2003) 165-181
24. Parnas, D.: On the Criteria to Be Used in Decomposing Systems into Modules. *Communications of the ACM* 15, n. 12 (1972) 1053-1058
25. SICS AB: Trading Agent Competition page. <http://www.sics.se/tac/page.php?id=1> (2006)
26. 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, n. 1-2 (2004) 145-189
27. Tarr, P., Ossher, H., Harrison, W., Sutton Jr., S.M.: N Degrees of Separation: Multi-Dimensional Separation of Concerns. *Proc. 21st International Conference on Software Engineering* (1999) 107-119
28. Ubayashi, N., Tamai, T.: Separation of Concerns in Mobile Agent Applications. In: *Third International Conference REFLECTION 2001. Lecture Notes in Computer Science*, Vol. 2192. Springer-Verlag, Berlin Heidelberg New York (2001) 89-109
29. Weiss, G., Rovatsos, M., Nickles, M.: Capturing Agent Autonomy in Roles and XML. In: *Proc. Intl. Conference on Autonomous Agents and Multiagent Systems* (2003) 105-112
30. Zambonelli, F., Jennings, N., Wooldridge, M. Developing multiagent systems: The Gaia methodology. *ACM Trans. on Software Engineering and Methodology* 12, n. 3 (2003) 417-470