

# Promoting Advanced Separation of Concerns in Intra-Agent and Inter-Agent Software Engineering

Alessandro Garcia    Christina Chavez    Otavio Silva    Viviane Silva    Carlos Lucena

Pontifical Catholic University of Rio de Janeiro (PUC-Rio)  
Computer Science Department/TecComm Group  
Rio de Janeiro, RJ, Brazil

{ afgarcia, flach, otavio, viviane, lucena@inf.puc-rio.br }

## ABSTRACT

Agent technology has been revisited as a complementary approach to the object paradigm. Although objects and agents have many similarities, the introduction of agents in the object model poses challenging problems because many system-level and agent-level properties are intrusive and overlapping. In this way, a disciplined approach is required for composition. We present our approach for dealing with the intricacies of developing agent systems using the recent advances of separation of concerns techniques, including aspect-oriented programming and computational reflection.

## Keywords

Software Agents, Separation of Concerns, Aspect-Oriented Programming, Computational Reflection.

## 1. MOTIVATION

Agent technology has been revisited as a complementary approach to the object paradigm, and has been applied in a wide range of realistic application domains. Software agents, like objects, include a specific set of capabilities for their users. In fact, objects and agents have many similarities [11], but the state of an agent is composed of beliefs, goals, plans and capabilities. Moreover, the behavior of software agents encompasses a number of *intra-agent properties* such as autonomy, adaptation, interaction, learning, mobility and collaboration. In addition, effective multi-agent systems (MAS) must include several *inter-agent (system-level) properties*, such as dependability and persistence.

The introduction of agents in the object model poses some problems because many of the intra and inter-agent properties are interactive and not orthogonal, and a disciplined method is required for their composition. For instance, the adaptation depends on autonomy since it is necessary to adapt the agent's state and behavior when the autonomy property decides to accept an incoming message (Figure 1). In addition, collaboration is viewed as a more sophisticated kind of interaction, since the former comprises communication and coordination. Interaction is only concerned with communication, i.e. sending and

receiving messages. Collaborating agents send and receive messages from each other. Therefore, the collaboration property additionally defines how agents can cooperate; i.e., it addresses the coordination protocol.

Traditionally, existing object-based proposals often focus on the implementation phase, and do not provide direct support for handling and reusing such intra and inter-agent properties separately (e.g. [1], [2] and [8]). Moreover, these proposals generally support a limited number of agent types, and the state and behavior of an agent are encapsulated as an object. Even though it is desirable for the internal structure and behavior of an agent to appear as a single object, this agent model results in agent design and implementation being complex and difficult to understand, maintain and reuse in practice. As a consequence, there is a need for a multi-agent software engineering approach that supports the handling of these intra and inter-agent properties directly at a preliminary design stage. This should be done in order to master the natural complexity of building multi-agent object-oriented applications.

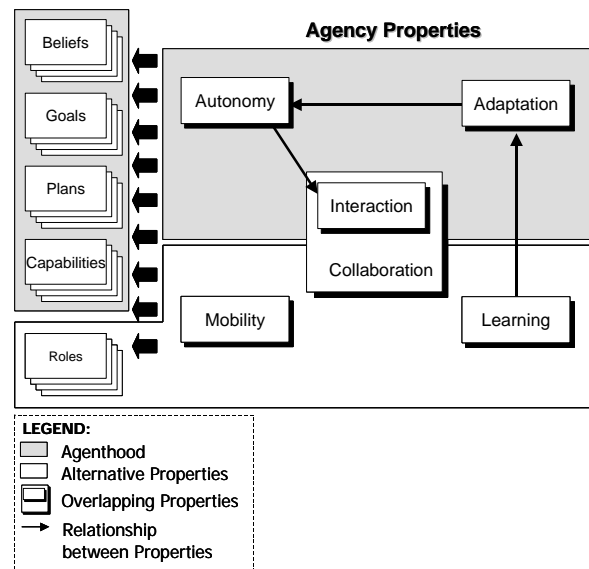


Figure 1 – Intra-Agent Properties

## 2. GOALS

Separation of concerns is one of the main tenets of software engineering using high-level abstractions to deal with software complexity. *Aspect-oriented programming* (AOP) [7] and *computational reflection* [9] are two techniques that can be used to promote better separation of concerns. Our research group [10] is working on the application of these techniques in order to develop a framework (figure 2) that handles the complexity of designing and implementing multi-agent OO software. In this context, we have the following specific goals:

- (i) proposing an aspect-based approach for designing and programming separately agency properties.
- (ii) developing a case study to compare an object-oriented and aspect-oriented design in order to support and validate (i).
- (iii) proposing a design model for aspect-oriented software development, which incorporates the main features of AOP and is language and process independent [4]. This model will be used as a conceptual framework in (i) and (ii).
- (iv) applying the notion of reflection upon tuple spaces architectures in order to manipulate separately inter-agent level properties.

## 3. THE APPROACH

### 3.1 AN ASPECT-BASED APPROACH FOR INTRA-AGENT PROPERTIES

We propose an aspect-based approach (Figure 2) to provide separation of concerns among the different intra-agent properties [5,6]. Classes and inheritance are used to model the core state and behavior of agents as well as different types of agents (e.g. information, user and interface agents). Aspects are used to model agency properties to be combined (or weaved) to these classes. Our approach allows the composition of these agent aspects in a disciplined and non-intrusive manner. As a consequence, we believe it supports the construction of multi-agent software with improved structuring for evolution and reuse of design and implementation solutions. The generic aspect-oriented design model presented in [4] is used as a conceptual framework for aspect modeling.

### 3.2 COMPARING OBJECT-ORIENTED AND ASPECT-ORIENTED MODELING

We are developing a comparative case study to assess and evaluate the benefits of applying aspect-oriented techniques and advanced object-oriented techniques to the design and implementation of the Portalware MAS [5,6]. Our case study purpose is characterize and evaluate those design models from the perspective of developers and maintainers.

Those design models were examined in a single project scope, where two teams have designed and implemented OO and aspect-based solutions for the same problem. After that, these teams evolved the system in order to incorporate some additional intra-agent and inter-agent properties by reusing characteristics already designed and implemented. The case study design incorporates descriptive methods to capture the different phases of system evolution: development, reuse and maintenance. On phase one, the two teams have designed and implemented the application. On phases two and three, both teams have maintained and reused intra and inter-agent properties. The measurement process considered several criteria, such as writability, readability, maintainability and reusability.

### 3.3 A REFLECTIVE ENVIRONMENT FOR INTER-AGENT PROPERTIES

A multi-agent environment is an infrastructure that provides inter-agent features to the development of MAS. In order to support these features, we propose a unified architecture based on reflective tuple spaces [12,13]. In our approach (Figure 2), mobile and persistent agents, as well as messages to be exchanged between agents are treated as tuples in programmable tuple spaces. These tuple spaces are programmed by the use of a reflective meta-layer attached to the base-level space. Each meta-tuple inserted in the meta-level space deals separately with each of the inter-agent properties. We believe our approach provides the separation of agent concerns because the inter-agent properties are treated at the system-level, simplifying the development at the agent-level. Moreover, tuples in reflective spaces are the basic units to tamper with when error detection and recovery are concerned.

## 4. RESULTS AND ONGOING WORK

This research work has begun in December 2000, and up till now we were able to achieve some results that were published in [3,4,5,6,12,13]. We have found our aspect-based and reflection-based approaches (Sections 3.1 and 3.3) establish an appropriate design language for MAS developers, allowing them to manipulate separately each intra-agent and inter-agent property. According to our comparative study (Section 3.2), our aspect-based proposal facilitates the development of MAS since different agency aspects reify the several concerns for the agent domain. During this study, we have detected the need for high-level models for aspect-oriented software design ([4]), so that we have focused our effort on defining such models. Finally, since the achievement of good separation of aspects is not a trivial task, we are currently investigating a set of design principles and aspect-based design patterns that provide good design solutions for AOP.

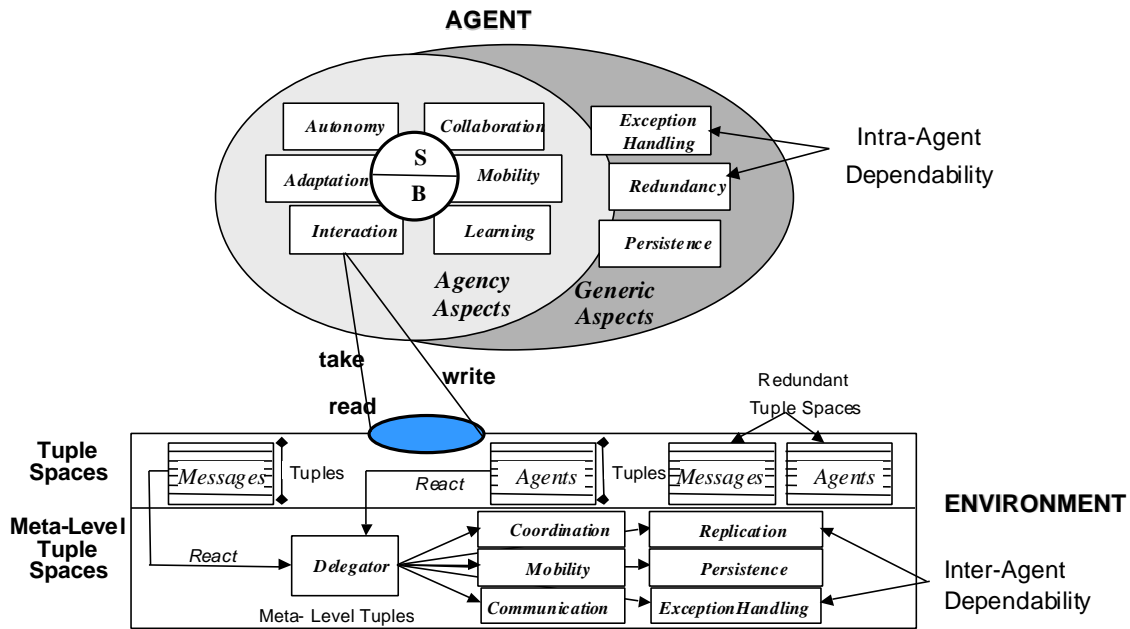


Figure 2 – Framework for dealing with intra and inter-agent properties

## 5. REFERENCES

- [1] J. Bigus and J. Bigus. "Constructing Intelligent Agents with Java". Wiley, 1998.
- [2] D. Brugali and K. Sycara. "A Model for Reusable Agent Systems". Implementing Application Frameworks, M. Fayad et al. (editors), John Wiley & Sons, 1999.
- [3] C. Chavez, A. Garcia, C. Lucena, "Some Insights on the Use of AspectJ and Hyper/J". Tutorial and Workshop on AOP and SoC, Lancaster, UK, August 2001 (to appear)
- [4] C. Chavez, C. "Design Support for Aspect-oriented Software Development". Doctoral Symposium and Poster Session. OOPSLA 2001, Tampa, October 2001 (to appear)
- [5] A. Garcia, C. Lucena, "An Aspect-Based Object-Oriented Model for Multi-Agent Systems". Advanced Separation of Concerns Workshop at ICSE'2001, Toronto, May 2001.
- [6] A. Garcia, V. Torres, C. Lucena, R. Miliđiu. "An Aspect-Based Approach for Developing Multi-Agent Object-Oriented Systems". Brazilian Symposium on Software Engineering, Rio de Janeiro, October 2001 (to appear)
- [7] G. Kiczales et al. "Aspect-Oriented Programming". In Proceedings of the ECOOP'97, Finland. Springer-Verlag LNCS 1241. June 1997
- [8] D. Lange, M. Oshima. "Programming and Developing Java Mobile Agents with Aglets." Addison-Wesley, August 1998.
- [9] P. Maes. "Concepts and Experiments in Computational Reflection". ACM SIGPLAN Notices, 22(12):147-155, 1987.
- [10] "SoC and MAS Research Group Web Page": <http://www.teccomm.les.inf.puc-rio.br/SoCAgents>
- [11] Object Management Group – Agent Platform Special Interest Group. "Agent Technology – Green Paper". Version 1.0, September 2000
- [12] O. Silva, A. Garcia, C. Lucena. "A Unified Software Architecture for System-Level and Agent-Level Dependability in Multi-Agent Object-Oriented Systems". ECOOP'01 MOS Workshop, Budapest, June 2001.
- [13] O. Silva, A. Garcia, C. Lucena. "T-Rex: A Reflective Tuple Space Environment for Dependable Mobile Agent Systems". 3<sup>rd</sup> WCSF Workshop at IEEE MWCN 2001, Recife, August 2001