# The Modeling of New Technological Domains in UML

E. Belloni, C. Marcos and J. Pryor

ISISTAN, Facultad de Ciencias Exactas, UNICEN
Paraje Arroyo Seco, B7001BBO Tandil, Argentina

Tel/Fax: + 54–2293–440362/3 - http://www.exa.unicen.edu.ar/~isistan/

E-mail: [ebelloni, cmarcos, jpryor]@exa.unicen.edu.ar

**Abstract**. UML is a language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. UML supports the typical software modeling problems. However, due to the diversity of software development domains, there may be occasions when the model will require elements or notations not provided by standard UML. This project proposes the definition of new UML profiles. Among different types of applications that these new UML profiles will support, are those based on agent technology and aspect-oriented software development.

## 1. Introduction

The UML (Unified Modeling Language) is a language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. These artifacts can be seen from two different perspectives: the UML definitions themselves, and how they are used to build software applications. The first perspective refers to the formal specification of UML, that is, the semantics, the notation, etc. of each of its elements. The second perspective refers to the fact that the designer´s choice of models and diagrams influences how the problem is approached and how the corresponding solution is shown.

The vocabulary of the UML consists of three kinds of building blocks: things, relationships, and diagrams. Things are the abstractions that are first-class citizens in a model; relationships tie these elements together; diagrams group interesting collections of things. Diagrams are the graphical representation of a set of elements, and they are used to visualize a system from different perspectives.

The elements which make up the UML diagrams have a well-defined semantics which is represented by a meta-model. Each element provided by UML has its corresponding class in the meta-model.

A UML diagram, such as a class diagram, does not usually contain sufficient information in order to describe all the aspects which are relevant to its specification. It is therefore necessary to describe additional restrictions with respect to the objects in the model. In order to express unambiguous restrictions, OCL has been developed. OCL - Object Constraint Language - is a formal language which consists of expressions, easy to read and write, and which has no collateral or secondary effects on the model.

UML supports the typical software modeling problems. However, due to the diversity of software development domains, there may be occasions when the model will require elements or notations not provided by standard UML. Quite frequently, users may also need to add non-semantic information to their models. These needs may be satisfied by UML through the extension mechanisms the language provides. The user may add new types of modeling elements with different semantics, characteristics, and notation, to those specified in the metamodel. On the other hand, new information may be added to the existent UML modeling elements. The main extension mechanism is the stereotype, which provides the means to define virtual subclasses to the UML meta-classes, with new meta-attributes and additional semantics.

The extension mechanisms provided by UML are *constraints*, *stereotypes*, *tag definitions*, and *tagged values*. When an element is stereotyped, UML is actually being extended by creating a new building block just like an existing one, but with its own special properties (each stereotype may provide its own set of tagged values), semantics (each stereotype may provide its own constraints), and notation (each stereotype may provide its own icon). These extension mechanisms may be used in an isolated fashion or in coherent sets which have been defined for a purpose or specific domain. A *Profile* is a packet stereotype which contains one or more (related) extensions to the standard semantics of UML. Profiles may contain stereotypes, tag definitions and restrictions; they may also contain data types to be used by the tag definitions.

The extension mechanisms of UML are useful for one or more of the following reasons:

- To add new modeling elements to those provided by UML.
- To define standard items that are not sufficiently interesting or complex to be included as elements of the UML meta-model.
- To define specific extensions for some domain, such as a development process or programming language (e.g. Profiles for the Unified Process and for Business Modeling).
- To incorporate semantic and non-semantic information to elements of the model.

A variety of new technologies and approaches for the development of software are not currently supported by UML diagrams. Additionally, the existent tools do not provide flexible and simple mechanisms that permit the incorporation of new modeling elements and the handling of profiles. Also, most of these tools are not open source.

This project proposes the incorporation of a new plug-in in the Eclipse platform to define new UML profiles. This project proposes the definition of new UML profiles. Among different types of applications that these new UML profiles will support, are those based on agent technology and aspect-oriented software development.

Section 2 and 3 introduce the multi-agent system and aspect-oriented software development concepts, respectively. Section 4 describes the objectives of the proposed project. The remaining section presents the conclusions.

## 2. Modeling Multi-Agent Systems

An agent is a software component allocated in a run-time environment to assist or represent someone or something, and it acts by itself with some degree of autonomy [Bra97; Jen98; Nwa96]. Other than this autonomy, there are several additional attributes by which it is possible to characterize a software agent, such as reactivity, pro-activity, social ability, adaptability, and mobility.

An agent-based application can have only one agent, but frequently these applications have several agents interacting and collaborating among themselves, becoming a multi-agent system. In this context, it is possible to identify a multi-agent system as having at least two different kind of agents, each of them belonging to one of the above five agent categories. Moreover, hybrid agents can also be involved in this kind of system.

Because of the diversity of multi-agent systems, their design is becoming more complex. In consequence, developers need tools and methodologies for the construction of this type of system. Agent oriented software engineering is still lacking adequate modeling support; in particular, UML does not provide the basic modeling elements that are needed.

In previous works we have already propose an extension of use case diagrams in order to support the modeling of agent-based systems, differentiating an application´s specific functionality and the functionality of each agent [Mar02]. Also, we have been working on a comprehensive set of views and models, which extends UML by contributing to the analysis and design phases of the development of mobile-agent applications [BM03].

## 3. Modeling Aspect-Oriented Applications

Aspect-oriented software development [AOSD02] is a new and widespread approach to the separation of concerns in software engineering [Dij76], in order to provide explicit support for modularising the design decisions that overlap or crosscut the functional decomposition of a system. As these features crosscut the primary functionality of the application, their code is spread throughout the basic functional components. In AOSD crosscutting concerns are encapsulated in separate modules called aspects, whose code is woven into the functional components of the system at predetermined join-points. Consequently, AOSD improves software quality attributes such as maintainability, legibility and reuse. As standard modeling techniques do not support AOSD, a notation is being defined in UML to support the specification of aspects when modeling aspect-oriented applications [PM03].

In order to extend UML with the above-mentioned two concepts with profiles and any other profiles that may be defined, it would be necessary to build a tool which supports the UML meta-model and the handling of profiles. This would allow developers of software for non-traditional domains or processes to extend the UML meta-model according to their needs and to create profiles by configuring new ones or extending and composing existent ones.

## 4. Objectives

Our proposal consists of the design and construction of a UML environment based on the Eclipse platform. The main objective of this environment is to permit the definition, configuration, composition, and administration of UML profiles, therefore supporting the modeling of software developed using novel approaches or techniques, or for non-traditional domains.

The editing environment would provide the following features:

- The definition of profiles of a particular domain for modeling applications, e.g., profiles for modeling agent and aspect-oriented software.
- The incorporation of new modeling elements to a new or existent profile, semantically defined by the UML meta-model.

- The support of a visual mechanism for the specification of new elements in OCL, as a complement to the meta-model description.
- The definition of new profiles or the composition of existent ones in order to satisfy the modeling requirements of a new application domain.

## 5. Conclusions

The modeling environment will provide, on the one hand, the concepts, characteristics and elements pertaining to UML diagrams for their use in typical development domains. On the other hand, it will also provide the mechanisms necessary for the definition of new profiles and their elements to be used in different application domains. Additionally, the proposed plug-in will allow developers to formally define and document these new elements in order to maintain the consistency of the meta-model and, consequently, the tool.

When there is a need to model new concepts and technologies, the developer must study their modeling requirements in order to identify what UML is lacking. He/she will then define new elements to support these requirements in corresponding profiles. These profiles can then be used to model, visualize, build, and document any applications developed with these new technologies.

One of the benefits of the proposed Eclipse plug-in is that a developer may extend the UML meta-model without doing any programming, simply by using a visual tool for the specification of the new concept.

This modeling environment will be integrated to the Eclipse platform such that it may be used with other plug-ins. One possibility is to complement this UML tool with another plug-in that generates the application code according to the specified models.

## 6. References


[AOSD02] 1st. International Conference on Aspect-Oriented Software Development. Enschede. Gregor Kiczales, ed., (ACM Press, The Netherlands, 2002).

[BM03] Belloni, E. and Marcos, C. “Modeling of Mobile-Agent Applications with UML”. In Proceedings of the Fourth Argentine Symposium on Software Engineering (ASSE´2003). 32 JAIIO (Jornadas Argentinas de Informática e Investigación Operativa), Buenos Aires, Argentina. September 2003. ISSN 1666-1141, Volumen 32.

[Bra97] Bradshaw J. An Introduction to Software Agents, in Software Agents, Bradshaw, J.M. (ed.), Cambridge, MA: MIT Press, (1997) 3 – 46

[Dij76] E. Dijkstra , A Discipline of Programming. (Prentice Hall, 1976).

[Jen98] Jennings N., and Wooldridge M. Applications of Intelligent Agents. In "Agent Technology: Foundations, Applications, and Markets", Springer-Verlag, eds Nicholas R. Jennings and Michael J. Wooldridge - pages 3--28, (1998)

[Mar02] Marcos, C. "Extensión del Diagrama de Casos de Uso para el modelamiento de Agentes" (Extending Use Case Diagrams for the Modeling of Agents). Facultad de Ciencias Exactas. ISISTAN. RT 14-2002. In Spanish.

[Nwa96] Nwana H. Software Agents: An Overview. Knowledge Engineering Review, Vol. 11 No 2 - (1996) 205 – 244

[PM03] Pryor, J. and Marcos, C. “Solving Conflicts in Aspect-Oriented Applications”. In Proceedings of the Fourth Argentine Symposium on Software Engineering (ASSE´2003), 32 JAIIO (Jornadas Argentinas de Informática e Investigación Operativa), Buenos Aires, Argentina. September 2003.