Defining the Proxy Design Pattern using UML Profile

Ana Garis, Daniel Riesco, Germán Montejano
Software Engineering Project of
National University of San Luis
Ejército de los Andes 950
(5700) San Luis-Argentina
{agaris, driesco, gmonte}@unsl.edu.ar

Abstract

Today software solutions are often modeled using UML. Design patterns are frequently instantiated within these particular solutions. However, in several situations, UML is not expressive enough to describe them properly.

UML profiles extend UML syntax and semantic in order to model elements of particular domains. As profiles extend UML vocabulary; design patterns define a common vocabulary for software designers. Because of this, it is possible to use a profile to define a design pattern's vocabulary in UML. Profiles can be used to solve particular problems in different domains. This work shows a way in which profiles can be used to define design patterns. The definition of the proxy design pattern is shown as an example of our proposal.

Keywords: UML Profiles, Design Patterns, OCL, Metamodels

1. INTRODUCTION

UML (Unified Modelling Language) is a popular language which allows specifying, constructing, visualizing, and documenting artefacts of software systems.

Sometimes design patterns are instantiated in UML design models. They help to transfer knowledge between developers, give them a common vocabulary; avoid to make mistakes in design (using the experience from other people) and document software [1].

UML is not expressive enough to model the semantic of a particular design pattern. It is required to extend UML using the "UML Profiles" mechanism. Profiles extend UML syntax and semantic in order to model specific elements of particular domains. Three elements are provided to extend it: stereotypes, tag values and constrains.

One of the reasons for defining profiles is that they add information to the model, transforming it to another model or to code. On the other hand, it increases the vocabulary of a specific domain, and it extends semantic and constrains of the UML metamodel.

This work show a way to define patterns using profiles. If a profile for each pattern is defined a library of patterns' profiles can be built. The advantages of the profiles can be used to define, document and visualize design patterns in UML models.

Profiles are not only used to specify particular domain such as real time or Enterprise Java Beans. Profiles for patterns can be used to solve particular problems in different domains (using profile for patterns transversally on dissimilar domains).

This work is structured as following. In the following section, "UML profiles" are introduced. Later several approaches for the specification of design patterns with UML are shown. Afterwards, the approach for defining "Proxy Pattern Profile" is described. Finally, the conclusions are presented.

2. UML PROFILES

The UML infrastructure is composed of four conceptual levels [2]. The elements of the M0 level are *instances* of a particular real system. The elements of M1 level are *models* of particular systems and its instances are M0 elements. M2 level is *the model of the model* and its elements are the languages for modelling (its instances are M1 elements; for example, "Person" is an instance of the "Class" element). M3 level is the *model of M2 model* (its instances are M2 elements; for example, "Class" is an instance of "Classifier"). Profiles change the UML metamodel in M2.

It is possible to define a package “Profile” [2] (see Figure 1) using UML version 2.0. Three mechanisms are included for profile definition: stereotypes, tag values and constrains. As a result the UML semantic is adapted to particular requirements without changing the UML metamodel.

Stereotypes extend the UML vocabulary. A stereotype is specified associating a stereotype name with an element of the metamodel (class, attribute, operation). It is denoted with double bracket in the following way <<stereotype-name>>. It is also possible associate it with tag values and constrains.

Tag values are attributes associated to elements extended by the profile. It is denoted by a pair with the attribute name and its associated value, i.e, {name=value}.

Constrains are semantic restrictions added to elements of the model. If constrains are attached to a stereotype, then all associated model elements must satisfy the restrictions of the stereotype. Usually natural language is used to define constrains, however Object Constraint Language (OCL) is a better option because it is more precise.

Profiles allow to define a new vocabulary for a specific domain, to add new semantic to UML

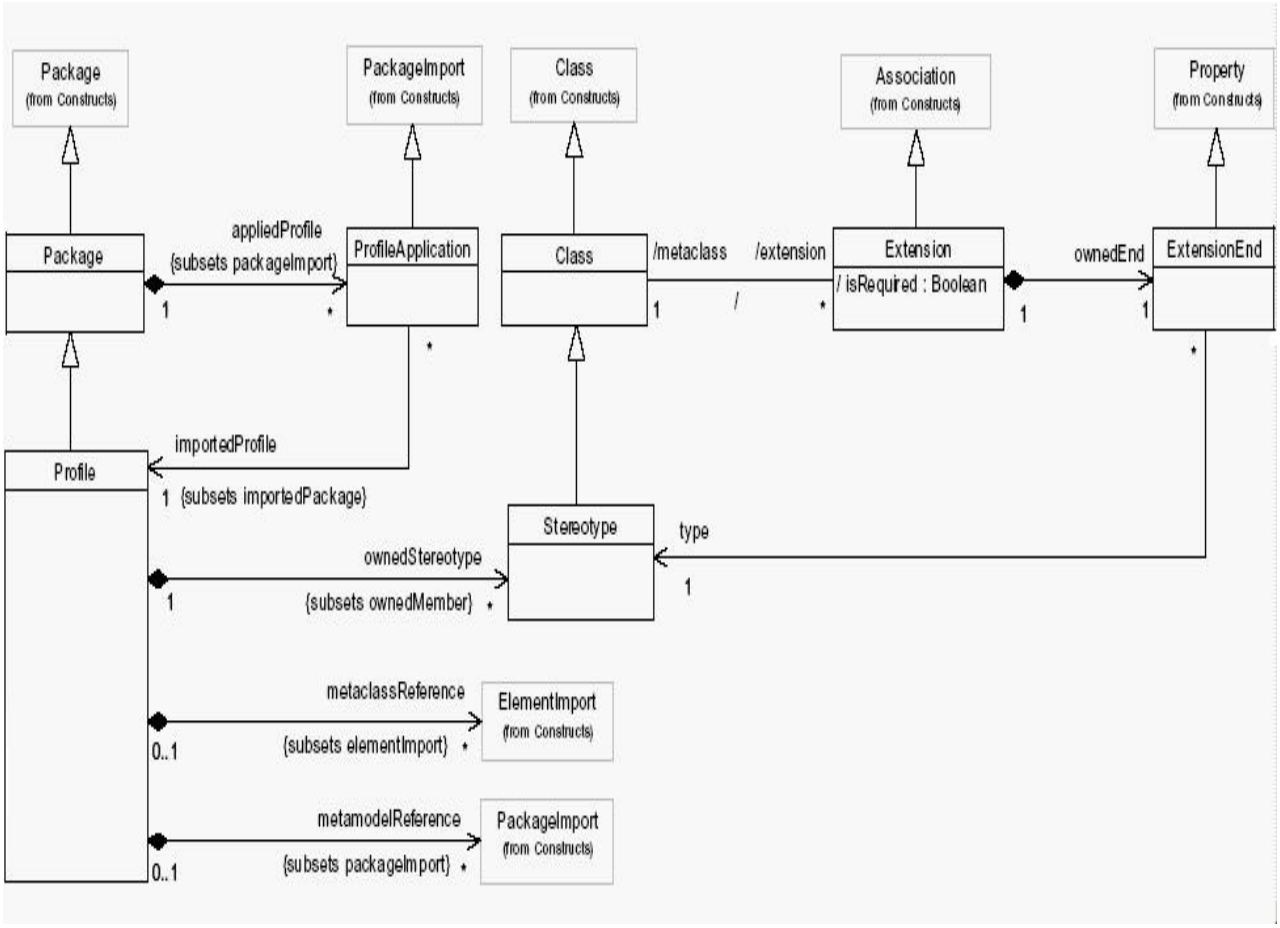

Figure 1: Profile Package

metamodel and to add constrains to existent metamodel elements. These advantages are clearly used in the MDA (Model Driven Architecture) context. MDA approach software development by the definition and transformation of models, propelling the definition of new elements for system's design and implementation [3].

3. DEFINING PROFILES FOR PATTERNS

This work approaches the design pattern's specification using UML profiles. A general structure for defining patterns is given by Profiles. However, it isn't possible to define semantics for all patterns in a single profile. A profile has to be defined for each pattern; in each profile the semantic of a particular pattern is described.

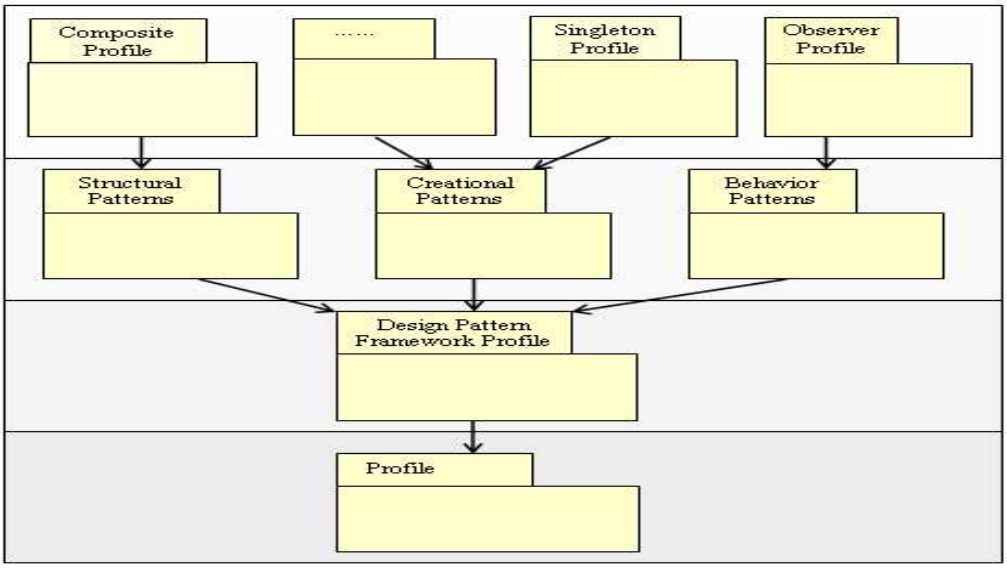

Figure 2: Architecture for patterns using UML profiles

An architecture organized into four levels is proposed. A hierarchy between levels of profiles (see Figure 2) is introduced allowing the reuse of definitions. The bottom level is the "Profile Package" of OMG standard, followed with a "Design Pattern Framework Profile" (DPFP). The next level is a "classes of patterns". The particular patterns are defined on the higher level.

Common features to all pattern profiles are specified in the DPFP level. The "classes of patterns" is inspired in one of the most popular catalogs of design pattern (see [1]).

The patterns are divided into Creational Patterns, Structural Patterns and Behavioural Patterns. The first group talks about the creation of objects. The second makes reference to how classes and objects are composed. And the third talks about the algorithms across collaborating objects.

A new profile should be defined on the higher level when a new pattern wants to be included. All patterns will satisfy the same generic structure but each profile will have its own particular semantic. The definition of “Proxy pattern” is shown as study case of this work.

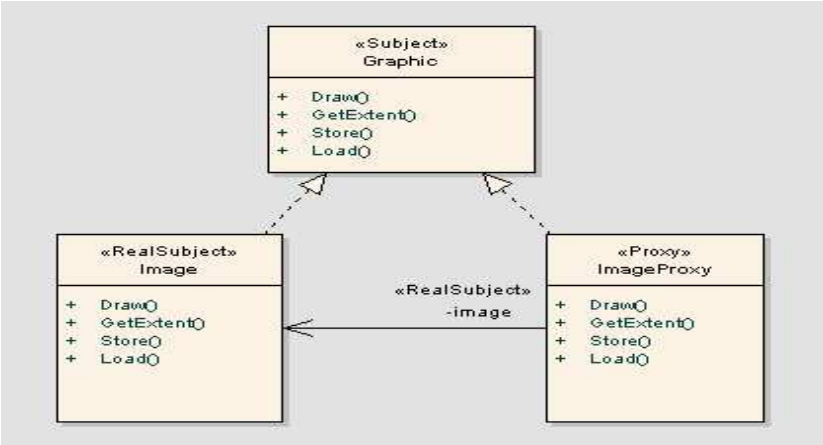

Figure 3: A model using stereotypes of Proxy Pattern

Suppose that the “Proxy pattern”[1] is instantiated during the design phase of document editor software. The capability of including graphics in a document is a typically required functionality for a document editor. The creation of graphic objects is an expensive operation, but to open a document should be fast.

The solution can be to define a “Graphic” superclass, an “Image” and an “ImageProxy” class, and use lazy initialization. Image proxy creates the real image only when the document editor needs to display it (invoking its “Draw” operation). If we define a profile for the proxy pattern, it is possible to introduce stereotypes in the model as it is shown in Figure 3. UML profile Proxy Pattern is explained below.

[1] Gamma E., Helm R., Johnson R., Vlissides J., “Design Patterns. Elements of Reusable Object-Oriented Software”, pag. 207.

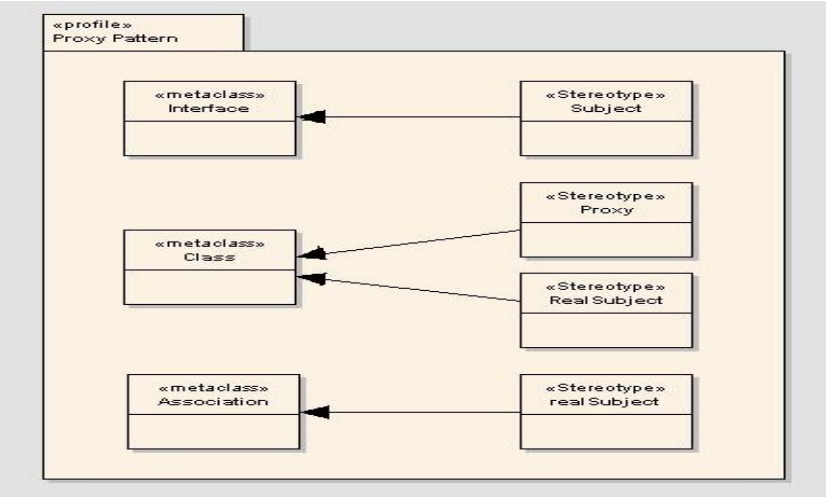

Figure 4: Proxy Profile

In order to verify the correctness of the model, OCL constrains should be associated with stereotypes in the profile. After adding constrains on UML metamodel elements, it is possible to check this constrains on model elements. Therefore, the model is correct according the syntactic and semantic rules defined by the profile.

Some OCL invariants should be checked while the stereotypes are introduced in a model. Also others OCL operations should be performed after the model is finished.

An OCL operation in the DPFP level is defined to check if all the required stereotypes of a particular profile are in a design model.

```
context Core::Profile::isElementsOfPattern(m:ModelElement):Boolean
isElementsOfPattern =
    self.ownedStereotype->forAll(s | m.model->
              exists( comp| comp= s.name) ) )
```

A particular OCL constrain of the Proxy Pattern Profile is shown below (The UML metamodel is shown in Figure 5 in order to understand the OCL specification)

```
-- The stereotyped “realSubject” relation is oriented from “Proxy” element to
“RealSubject” element --

       context realSubject inv:
             self.base.connection->exists(c1,c2|
                          c1.participant.isSterotyped(“Proxy”) and
                          c2.participant.isSterotyped(“RealSubject”) implies
                            c1.aggregation=#composite and c2.aggregation=#none )
```

The verification of OCL constrains with an UML tool can be done combining techniques. 1) A delayed consistency check and 2) a check of incorrect use of the profile.

One way to define a pattern profile can be as follows. First, it is important to identify the main participants of the pattern and its responsibilities. Then, a stereotype is included in the profile for each participant. At this moment it has not been necessary to use tag values. Finally the rules that must be satisfied because of the pattern behaviour is represented through OCL's constrains.

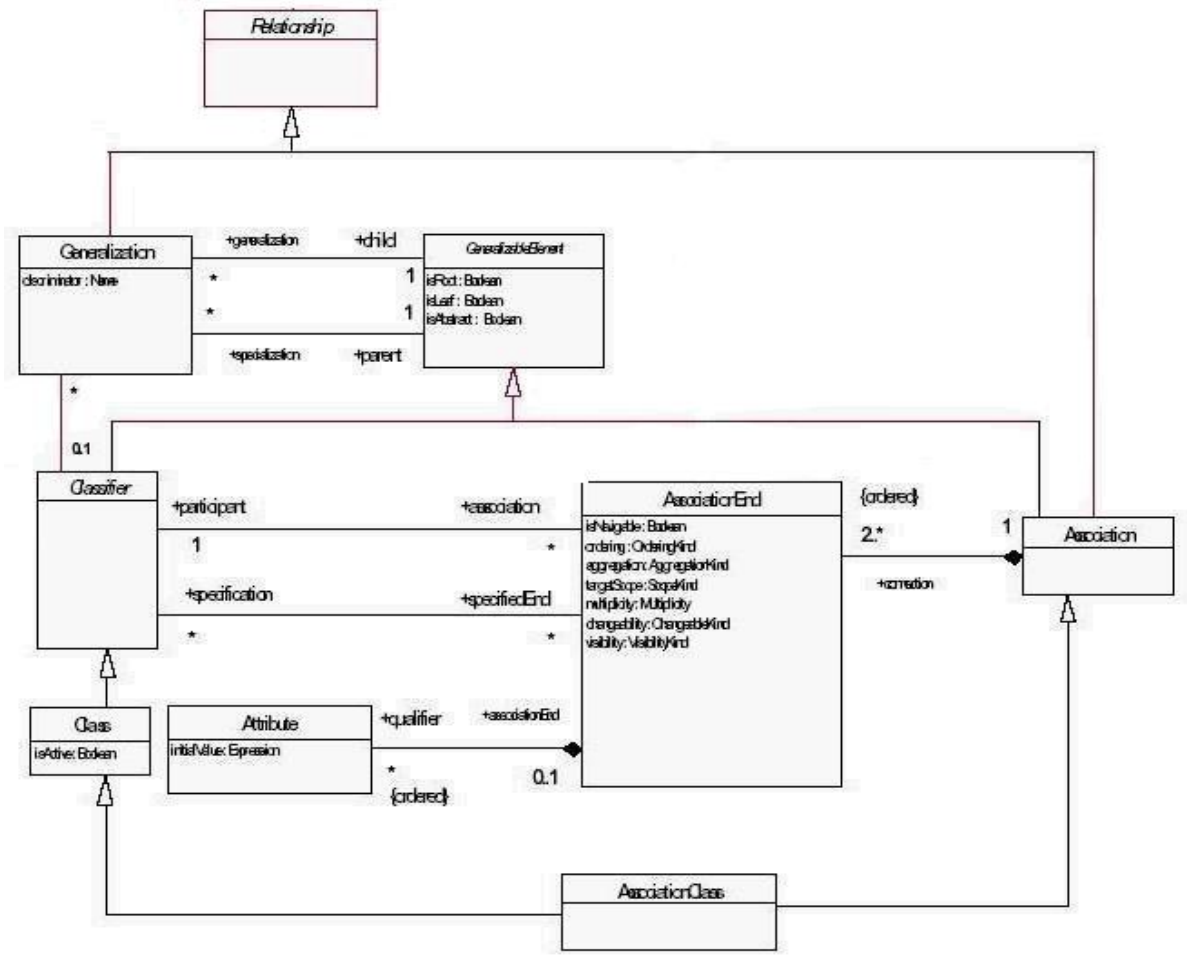

Figure 5: Core Package - Relationships

The profile of a pattern should be general enough to describe the essential spirit of the pattern. If the definition of a pattern is too restrictive, the benefits of using pattern could be lost when it is instantiated in different situations.

In a similar way as the "Proxy pattern" has been defined, others profiles of patterns can be defined too. All this profiles will constitute a library of profiles of patterns.

4. APPROACHES FOR THE SPECIFICATION OF DESIGN PATTERNS WITH UML

Several works like [4], [5], [6] and [7] studied the definition and documentation of patterns with UML. Lauder and Kent in [4] applied a visual modelling notation dividing the specification of patterns into three models: 1) an abstract model which catches the pattern's essential spirit, 2) a "type model" that refines the abstract model (adding domain specific constraints), 3) "class model" which documents a concrete deployment. The advantage of this approach is that it is possible to differentiate between several levels of abstraction. It uses both a visual and formal notation.

OMG considers in [5] that rules that govern a business concept can be represented with a pattern with constraints. Patterns can be expressed in any of three ways: a simple pattern, which defines a particular pattern, an inherited pattern which inherits from another pattern, or composite pattern which is a combination of other patterns. First templates with constrains are defined for representing a pattern, later it is bound in a model using collaborations. This mechanism is based on UML 1.4 version. The main disadvantage of this approach is that it doesn't allow defining stereotypes for each pattern's element.

Fontoura and Lucena in [6] extend UML defining a class of design patterns that they call "configuration patterns". These patterns are oriented to be applied to complex frameworks. An UML extension is developed to express "abstract diagrams". Abstract diagrams show several implementation approaches and they allow representing the pattern instantiation.

France, Kim and Song in [7] define a metamodeling language (using UML and OCL constrains), specifying perspectives of design patterns, such as static structure and interactions. A particular model satisfies the specification if its elements play the roles defined in the specification and they do not violate constrains.

Other works as [8] and [9] are less oriented to pattern's specification for documentation purpose, but they show other important features. Sanada and Adams in [8] extend UML to support design patterns and frameworks in class diagrams. All the extensions are defined as UML profiles; one profile for frameworks and other for patterns. Dong and Yang in [9] present an UML profile to improve the visualization of design patterns in UML diagrams. Therefore, it is possible to represent in the diagram the role of each element in a design pattern. Both works do not define a profile for each pattern, but they define features towards the improvement of the visualization of patterns when they are shown in UML models.

Barotto, Demonte and Riesco in [10] propose the definition of patterns modifying the UML metamodel. They use stereotypes with OCL constrains for the specification of pattern elements. This approach is very similar to the present work but they don't use the "UML Profile" notion, they use concepts previous to UML 2.0.

One of the advantages of using UML profiles to model patterns is that if UML tools are profile aware, is not required to construct a specific tool for patterns. This is an important advantage against the others approaches, such as [4]. In [7] the authors are concerned in making their work compliant with existing UML CASE tools. However, they had to implement tool's extensions. In [8] an UML tool was extended to add the required functionality.

Our work has shown that defining a profile for each pattern is an option for the specification of design patterns. It is different to [6] and [7] because it doesn't use an especial syntax for representing patterns; it uses an UML component: "the profile". Therefore, a profile is used not only to define specific domain but it is used to define a general domain too, as it is the definition of patterns.

During last years, UML profile concept has been used in different domains. Nevertheless, it wasn't used to define patterns such as it is proposed in [11], [12] and this work.

5. CONCLUSIONS

This work proposes an approach that uses profiles to define design patterns. Each profile of pattern is within a layered architecture. The hierarchy between levels of profiles allows the reuse of definitions. The profile is used as a tool to document and define design patterns.

In order to illustrate the specification proposed, the definition of Proxy pattern and later a resumed technique for defining a pattern profile are shown. A library of patterns can be built with all the pattern profiles.

There are advantages in defining profiles for particular domains; such as to add information to the model to transform it to other models, and extending semantic of the UML metamodel.

Although profiles were often used to model specific domains, they can be used to model general domains as design pattern is. Furthermore, if profiles are used to represent patterns then it is not required to define a special notation.

If developers can introduce stereotypes in their models, they can see clearly the pattern that they use, to improve communication with their colleagues and to establish a common vocabulary.

REFERENCES

[1] Gamma E., Helm R., Johnson R. and Vlissides J. Design Patterns. Elements of Reusable Object-Oriented Software. *Addison-Wesley*. 1995.

[2] UML 2.0 Infraestructure Specification. http://www.omg.org/technology/ documents. August 2004.

[3] Object Management Group, MDA Guide Version 1.0.1. OMG document, http://www.omg.org/docs/omg. December 2004.

[4] Lauder A. and Kent S. Precise Visual Specification of Design Patterns. *ECOOP'98, 12th European Conference on Object-Oriented Programming*. Vol.1445 LNCS, pg. 230-236. July 1998.

[5] OMG. UML Profile for Patterns Specification. http://www.omg.org/technology/ documents. 2004.

[6] Fontoura M. and Lucena C. Extending UML to Improve the Representation of Design Patterns. *Journal of Object Technology*. 2000.

[7] France R., Kim D. and Song E., A UML-Based Pattern Specification Technique. *IEEE Transactions on Software Engineering*. Vol.30(3), pg.193-206. 2004.

[8] Sanada Y. and Adams R. Representing Design Patterns and Frameworks in UML. Towards a Comprehensive Approach. *Journal of Object Technology*. Vol.1(2), pg.143–154. 2002.

[9] Dong J. and Yang S. Visualizing Design Patterns With A UML Profile. *IEEE Symposium on Human Centric Computing Languages and Environments (HCC 2003)*: Pg.123-125. 2003.

[10] Barotto V., Demonte M. and Riesco D. Definición de Patrones de Diseño a través del metamodelo UML. Tesis de Licenciatura en Ciencias de la Computación. Universidad Nacional de Río IV, Argentina. 2005.

[11] Garis A., Riesco D., Montejano G. and Debnath N. UML Profiles for Design Patterns. *ISCA 20th Internacional Conference on Computers and their Applications*, pg 435. Marzo del 2005.

[12] Garis A., Riesco D., Montejano G. and Debnath N. Defining Patterns using UML Profiles. *ACS/IEEE International Conference on ComputerSystems and Applications*, Proceedings publicados por la IEEE Press, www.ieee.org. Marzo del 2006.