Defining Patterns Using UML Profiles

N. C. Debnath
Winona State University
Department of Computer Science
Winona, MN 55987 USA
ndebnath@winona.edu

A. Garis, D. Riesco, G. Montejano
Universidad Nacional de San Luis
Ejército de los Andes 950
5700 San Luis – Argentina
{agaris | driesco | gmonte}@unsl.edu.ar

Abstract

Sometimes, UML is not enough expressive to describe patterns property. UML profiles allow extending UML syntax and semantic for modeling elements of particular domains. As profiles extend UML vocabulary, and design patterns define for designers a common vocabulary, so it is possible using profile for defining a pattern vocabulary in UML. Profiles can be used for solve particular problems in different domains.

This work shows the way for defining design patterns with profile, proposing architecture in levels. It shows how the definition of a profile for a particular pattern is, and how an UML tool can be enough for introduce profile for patterns. It analyzes the advantages of using profiles to define, document, and visualize design patterns.

1. Introduction

Nowadays, UML (Unified Modeling Language) is one of the most used modeling languages for the design of system 0. Many times UML is not enough expressive for modeling specific elements of particular domain. In these situations is necessary to extend UML. Profiles are introduced in UML to extend its syntax and semantic; then it is possible to represent more specifics concepts [2]. One of the reasons for defining profiles is that they add information to the model to transform it to other models 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.

Many software developers use UML to model design patterns in their solutions. A pattern expresses the experience of expert designers. They help to transfer knowledge between developers, give them a common vocabulary and document software [3].

This work shows the way for defining patterns with profiles, defining a profile for each pattern. As profiles extend UML vocabulary and patterns define for designers a common vocabulary, so it is possible using profile for defining a pattern vocabulary in UML. Profile for patterns can be used not only to specify particular domain but it can be used for solve particular problems in different domains (using profile for patterns transversally on dissimilar domains).

In the following section, UML profiles are introduced. Later approaches for the specification of design pattern and tools to support profile for patterns are presented. Afterward the way for defining pattern profiles is described. Finally, the conclusions are showed.

2. UML profiles

The profiles idea is described in [13] (Chapter 4, Terms and definitions): *"A stereotyped package that contains model elements that have been customized for a specific domain or purpose using extension mechanisms, such as stereotypes, tagged definitions and constraints. A profile may also specify model libraries on which it depends and the metamodel subset that it extends."*

For extending and adapting UML to a platform or domain, standard UML define a package "Profile". This package includes three mechanisms for refining: stereotypes, tag values and constrains. These elements allow adapting the UML semantic to particular requirements, without changing the UML metamodel.

Stereotypes extend the UML vocabulary and it is possible to associate to it tag values and constrains. Tag values are attributes associated to elements extended by the profile. Constrains are semantic restrictions added to elements of the model. Many times natural language is used for define constrains however Object Constraint Language (OCL) is a best option because it is most precise.

One of reasons for defining profiles is that it allows defining a new vocabulary for a specific domain, defining a particular notation for existent elements in accordance with these domain, adding semantic for metamodel elements what have an imprecise semantic, adding new semantic to metamodel and adding constrains to existent metamodel elements [13]. These advantages can be used for MDA (Model Driven Architecture) which approaches the software development by definition and transformation of models [14].

3. Other approaches

There are different ways for describe patterns, a formal approach and other more informal. The formal specification is the best way to achieve a precise notation in accordance with [4], [5], [6], [7]; however, can be sophisticate and difficult understand. In [6] and [7], mathematics background is required to comprehend the specification. Smith and Stotts in [6] show how the language termed ρ-calculus can express design patterns. Flores in [7] proposes a formal specification of pattern using a formal language (RSL).

Numerous works have studied the use of UML to define and document patterns. For example,. Le Guennec, Sunyé and Jézéquel in [5] modificate the UML 1.3 metamodel. They allow meta-level collaborations and OCL constraints instead of parameterized collaborations. The OMG in [12] introduce notions for defining patterns and for applying patterns to models. Fontoura and Lucena in [8] extend UML for representing a class of design pattern called "configuration patterns". France, Kim and Song in [11]. define a metamodeling language for specifying different perspectives of design patterns.

Some works as [9] and [10] are less oriented to specification of patterns for documentation but they show other important features. Sanada and Adams in [9] extend UML for supporting design patterns in design class diagrams. Dong and Yang in [10] present a UML profile for a best visualization of design pattern in UML diagrams.

Such as, it is observed in [11] and [4], if the pattern specification is precise then pattern-based development techniques and supporting tools can be used for verification of the presence of pattern in design models, building of solutions from patterns, incorporation of pattern into design model, and generation of new models and code from patterns.

4. Tool to support profile for patterns

An appropriate CASE tool must allow define and incorporate profiles in the models. After a definition of a profile for a pattern and its incorporation in the repository, the instantiation of predefined patterns to design elements in a particular model it is necessary. The UML tool should get the possibility of the definition of stereotypes, tag values and OCL constrains; later allow to incorporate them into a model and to verify that these model's elements are in accordance with the OCL constrains.

A form to define profile for patterns is to choose a CASE tool (see [16]) that offer the opportunity of introduce new stereotypes and tags in the metamodel and incorporate them in the models. The tool must have a way for exporting models to documents written in a standard specification (as XMI [17]), therefore the resulting file can be analyzed in order for checking if the model is consistent with respect to the profile constrains.

Instead of implementing a delay consistency check, in [15] a different approach is showed. The incorrect use of the profile is prevented extending a CASE tool with a set of profile operations. These operations are the interface for using elements of a profile and for ensuring the consistency of its use. Also it is possible automatically compute derived stereotypes and tags. The operations have associated constraints, which are checked in the moment of the operation execution. With this strategy, the CASE tool must allow the definition of stereotypes and tags, and provides scripting facilities for implementing the profiles operations.

5. Defining pattern profiles

Profiles give a general structure for defining patterns. However, it isn't possible to define a semantic for all patterns in a single profile, it is necessary to do a profile for each pattern; in each profile the semantic of a particular pattern is described.

This work proposes architecture for patterns using UML profiles, showing that the definition of a profile for each pattern is an option for doing the specification of design patterns. The architecture proposed is structured in levels; a hierarchy between levels of profiles (see Figure 1) is imposed allowing the reuse of definitions. In the bottom level is the "Profile Package" of OMG standard, following with a "Design Pattern Framework Profile" (DPFP), a level of class of pattern and finally, on the top, the definition of particular patterns.

DPFP has common specification features to all pattern profiles. This framework will add particular elements for pattern; also the redefinition of profile elements (stereotypes, tag values and constrains) necessary for attaching a semantic specific of patterns.

The level of class of pattern is inspired in one of the most popular catalogs of design pattern ([3]), where the patterns are divided in Creational, Structural and Behavioural. They reuse definitions of DPFP, but each class of pattern add a particular semantic in it definition. Finally, when a new pattern want be included then a new profile is defined; each profile of added patterns will have a particular semantic, but all patterns will respect the same generic structure of some of class of pattern.

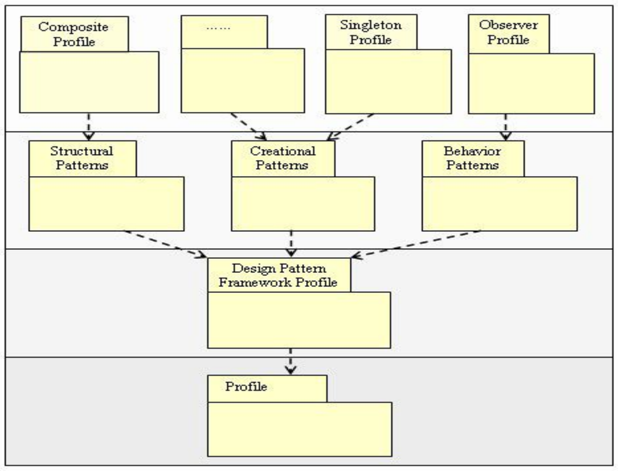

Figure 1: Architecture for patterns using UML profiles

One of the advantages for using UML profiles for patterns is that if UML tools know profiles features, it is not necessary define a specific tool for patterns. This feature is an important advantage in relation with others approaches.

This work is some different to [5] because not use the traditional approach proposed for UML as is the collaboration model and templates. Also it differs of [9] and [10] since it not defines a profile for each pattern. Both are more oriented to easy visualization of pattern when they are applied in UML models. Neither is same to [8] and [11] because it is not introducing a new notation (only use an UML artefact: 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.

5.1 A Case Study

Suppose that the composite pattern [3] is used in a design problem. Different elements should be draw at the screen, therefore an abstract method “draw()” is defined in the Graphic class and it is implemented by all its subclasses. The “Picture” class is a composite of graphics, then for drawing it should send the message “draw” to all its “Children”. We can see that “Picture” class is working as composite, “Graphic” as component; and the “Line” and “Rectangle” classes as leaf components.

Consider that the developer wants to visualize in the design the pattern that it is been used to communicate knowledge with other developers, to give them a common vocabulary, and to have more clear documentation of the system design. Having UML elements for this last propose will help for identifying the pattern used.

Before adding stereotypes in the design diagram, it is necessary to specify a profile for the composite pattern (see Figure 2). This profile defines three class stereotypes (Component, LeafComponent and Composite) and one association stereotype (Children).

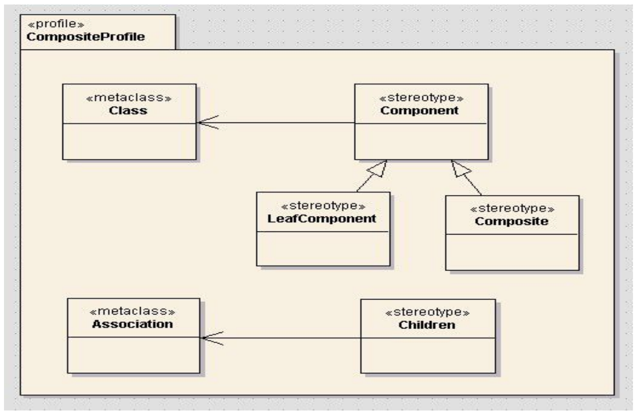

Figure 2: Composite Profile

In this case, tag values are not defined but constrains are established using OCL. As example, some constrains of the Composite Profile appear in Figure 3.

```
Children:(Core::Association)
inv:
  self.connection->exists
     (participant.isSterotyped(“Composite”) and
       multiplicity.min=1 and
       multiplicity.max=1) and
       self.connection->forAll(c1,c2|
         c1.participant.isSterotyped(“Composite”) and
         c2.participant.isSterotyped(“Component”)
            implies c1.aggregation=#composite and
                   c2.aggregation=#none )

inv:
 Class.allInstances->forAll(c1,c2|
     c1.isSterotyped(“Composite”) and
     c2.isSterotyped(“Component”) and
     c1.oclType.superTypes->includes(c2.oclType)
         implies
            Class.allInstances->exists(c3|
                  c3.isSterotyped(“LeafComponent”) and
                  c3.oclType.superTypes->includes(c2.oclType)))
```

Figure 3: Constrains of Composite Profile

After adding constrains on UML metamodel elements, it is possible to check this constrains on model elements that use stereotypes of the profile. Therefore, the model is correct according the syntactic and semantic rules that the profile has established.

We propose a way for doing the verification of OCL constrains with a UML tool (see section 4). It is combining a delay consistency check (off line check) and the check of incorrect use of the profile (on line check). Features in the DPFP are defined in order for combination of an off and on line checking. In particular, OCL constrains can add a specific semantic that indicate if it can be checked within on-line way or off-line way.

The pattern UML profile should be enough general to describe the essential spirit of the pattern. If the pattern definition is much restricted, the benefit of pattern using in different situations is loss.

In the same way, it has been defined the composite profile for the composite pattern, others profiles can be defined for other patterns. A resumed technique for defining a pattern profile can be as following. First, it is necessary to identify the main participants of the pattern and its responsibilities in order to have cleared the metamodel to describe. Later, when the profile is defined, a stereotype is included for each participant of the pattern. If tag values are required they can be include in the profile associated to a stereotype. Finally the rules that impose the pattern behaviour is represented through constrains in OCL.

Conclusions

The advantages of defining profiles for particular domains are many; such as to add information to the model to transform it to other models, and to extend semantic of the UML metamodel. If a profile is used for a design pattern, all these advantages are gained. Although profiles were often used for defining specific domains, we show that they can be used for general domains as the pattern definition is. For other way, if profiles are used to represent patterns then it is not necessary to define a special notation.

This work proposes architecture structured in levels for defining design patterns with profiles. A hierarchy between levels of profiles is imposed allowing the reuse of definitions. The profile is used as a tool to document and define design patterns. Developers can introduce stereotypes, tag values and OCL constrains in their models. This allows them to see clearly the pattern that they used, to improve communication with their colleagues and to establish a common vocabulary.

In addition, it analyzes how a UML CASE tool should define, incorporate profiles in the models and how it should check if the model is consistent with respect to the profile constrains. Finally it proposes that OCL constrains can add in DPFP a particular semantic indicating if it can be checked within on-line way or off-line way.

References

[1] ISO/IEC. Unified Modeling Language (UML), Version 1.5, International Standard ISO/IEC 19501.

[2] L. Fuentes, A. Vallecillo, “Una Introducción a los Perfiles UML”, Novática, 2004, vol.168, pp. 6-11.

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

[4] G. Florijn, M. Meijers, P. Van Winsen, “Tool Support for Object-Oriented Patterns”, In *ECOOP’97, 11th European Conference on Object-Oriented Programming*, 1997, vol. 1241 LNCS, pp. 472-495.

[5] A. Le Guennec, G. Sunyé, J. Jézéquel, “Precise Modeling of Design Patterns”, In *UML 2000*, 2000, vol. 1939 LNCS, pp. 482-496.

[6] J. Smith, D. Stotts, “Elemental Design Patterns: A Formal Semantics for Composition of OO Software Architecture”. In *27th Annual IEEE/NASA Software Engineering Workshop*, 2002.

[7] A. Flores, “Soporte Formal para Diseño basado en Patrones y Formalización de Patrones Estructurales Gof”, Tesis de Magíster en Ciencias de la Computación, 2004.

[8] M. Fontoura, C. Lucena, “Extending UML to Improve the Representation of Design Patterns”, *Journal of Object Technology*, 2000.

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

[10] J. Dong, S. Yang, “Visualizing Design Patterns With A UML Profile”, *IEEE Symposium on Human Centric Computing Languages and Environments (HCC 2003)*, 2003, pp.123-125.

[11] R. France, D. Kim, E. Song, “A UML-Based Pattern Specification Technique”, *IEEE Transactions on Software Engineering*, 2004, vol.30(3), pp.193-206.

[12] OMG.“UML Profile for Patterns Specification”, http://www.omg.org/technology/ documents, 2004.

[13] UML Superstructure 2.0 Draft Adpted Specification, http://www.omg.org/technology/ documents, 2004.

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

[15] J. Cabot, C. Gómez, “A simple yet useful approach to implementing UML Profiles in current CASE tools”, *Workshop in Software Model Engineering*, 2003.

[16] M. Jeckle, “UML Tools (Case & Drawing)”, http://www.jeckle.de/umltools.htm

[17] “OMG-XML- Metadata Interchange Specification”, Version 1.2, http://www.omg.org, 2002.