PAMPERO: Precise Assistant for the Modeling Process in an Environment with Refinement Orientation

Claudia Pons[(1) (2)], Roxana Giandini[(1)], Gabriela Pérez[(1)], Pablo Pesce[(1)], Valeria Becker[(1)], Jorge Longinotti[(1)], and Javier Cengia[(1)]

[(1)] LIFIA – Facultad de Informática, Universidad Nacional de La Plata, Calle 50 esq. 115. CP 1900. La Plata, Buenos Aires, Argentina
[(2)] CONICET (Consejo Nacional de Investigaciones Científicas y Técnicas)
cpons@info.unlp.edu.ar

1 Introduction

Abstraction [2] facilitates the understanding of complex systems by dealing with the major issues before getting involved in the detail. Apart from enabling for complexity management, the inverse of abstraction, refinement, captures the essential relationship between specification and implementation. Refinement relationship makes it possible to understand how each business goal relates to each system requirement and how each requirement relates to each facet of the design and ultimately to each line of the code. Documenting the refinement relationship between these layers allows developers to verify whether the code meets its specification or not, trace the impact of changes in the business goals and execute test assertions written in terms of abstract model's vocabulary by translating them to the concrete model's vocabulary.

Refinement has been studied in many formal notations such as Z [1] and B[4] and in different contexts, but there is still a lack of formal definitions of refinement in semi-formal languages, such as the UML. The standard modeling language UML [5] provides an artifact named *Abstraction* (a kind of Dependency) to explicitly specify abstraction/refinement relationship between UML model elements. In the UML metamodel an Abstraction is a directed relationship from a *client* (or clients) to a *supplier* (or suppliers) stating that the client (the refinement) is dependent on the supplier (the abstraction). The Abstraction artifact has a meta attribute called *mapping* designated to record the abstraction/implementation mappings, that is an explicit documentation of how the properties of an abstract element are mapped to its refined versions, and on the opposite direction, how concrete elements can be simplified to fit an abstract definition. The more formal the mapping is formulated, the more traceable across refinement steps the requirements are.

Although the Abstraction artifact allows for the explicit documentation of the abstraction/refinement relationship in UML models, an important amount of variations of abstraction/refinement remains unspecified, in general hidden under other notations. For example UML artifacts such as generalization, composite association, use case inclusion, among others, implicitly define abstraction/refinement relationship. The starting point to enable traceability across refinement steps is to discover and precisely capture the various forms of the abstraction/refinement relationship, in particular those forms which are hidden in the model.

N.J. Nunes et al. (Eds.): <<UML>> 2004 Satellite Activities, LNCS 3297, pp. 246 – 249, 2005.
© Springer-Verlag Berlin Heidelberg 2005

2 Tool Support

The task of documenting refinement steps needs to be assisted by tools. To experiment, we created a tool integrated in the Eclipse environment [3], called PAMPERO (**P**recise **A**ssistant for the **M**odeling **P**rocess in an **E**nvironment with **R**efinement **O**rientation), based on the formal definition of refinement [6] [7]. The tool can be downloaded from http://sol.info.unlp.edu.ar/eclipse; it supports the documentation of explicit refinements (i.e. Abstractions artifacts with their corresponding mapping expressions) and the semi-automatic discovering and documentation of hidden refinements.

PAMPERO consists of four components: an editor, an abstraction/refinement translator, an OCL evaluator, and a detective:

The Editor. The editor supports the creation of a number of UML and OCL artifacts, including Abstractions; see figure 1. Additionally, the editor allows developers to specify the abstraction mapping attached to Abstraction artifacts, using OCL expressions.

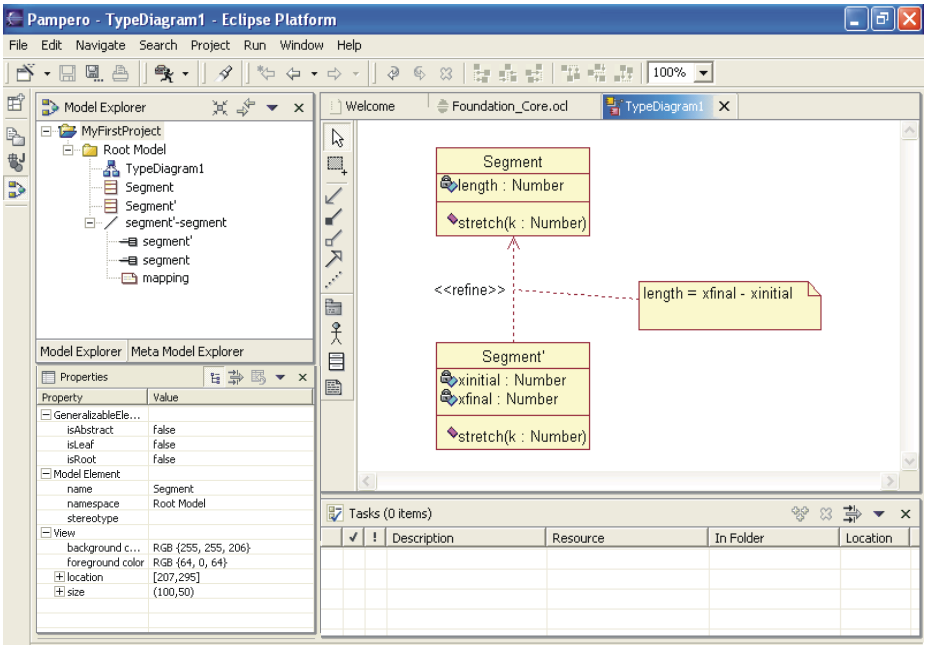

Fig. 1. The PAMPERO tool: Edition of explicit refinement

The abstraction/refinement Translator. The translator takes an OCL expression attached to a Class and translates it to concrete vocabularies, following the refinement steps. The translation of expressions attached to elements other than Class, is not supported yet.

The evaluator. The evaluator takes OCL expressions and evaluates them on a given model. Expressions might be either originally written in the model's vocabulary or translated by the translator from another abstraction level. The evaluator was implemented following the design of the USE evaluator [8]. Figure 2 shows the evaluation of OCL well-formedness rules on the model.

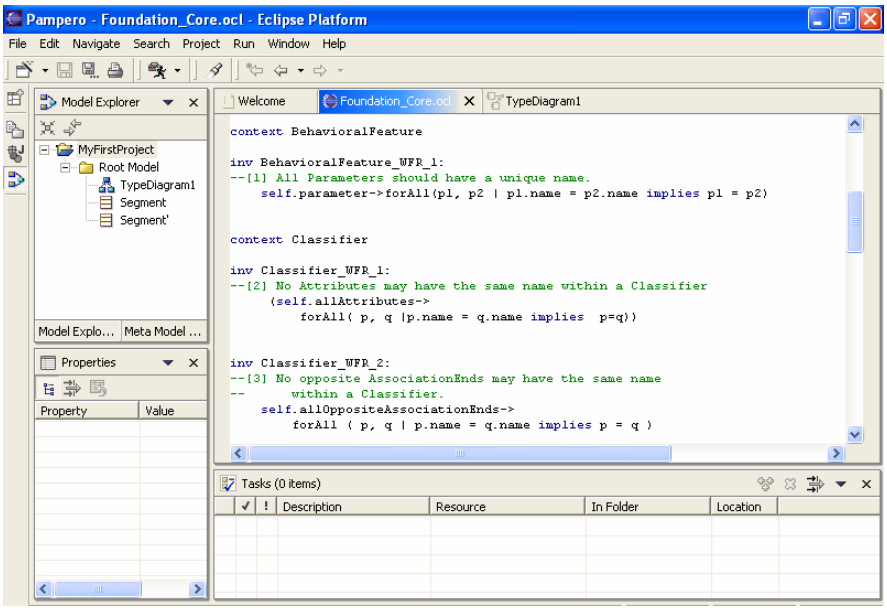

Fig. 2. The PAMPERO tool: Evaluation of OCL constraints

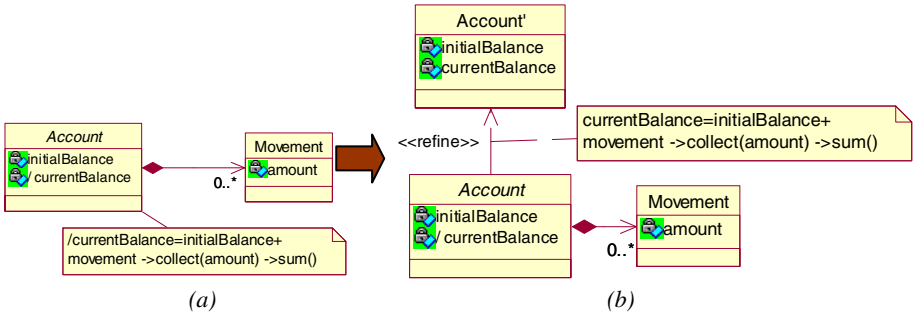

Fig. 3. Refinement hidden under decomposition: (a) Composite Association relationship. (b) Refinement relationship derived from the Composite

The Detective. This component looks into the model to discover and reveal cases of hidden refinement. The abstraction mappings automatically generated by the detective are generally in an immature state and should be completed by the developer. Figure 3

displays and example where a refinement relationship hidden under composite association is discovered and revealed by the tool. In the example the specification of the derived attribute currentBalance is suggested as mapping making it possible to translate OCL invariants such as (**Context** Account' **inv:** currentBalance>0) to a refined version such as:

Context Account inv: (initialBalance + movement->collect(amount)->sum()) > 0.

3 Conclusions

To enable traceability of requirements the presence of "undercover refinement" should be discovered and precisely documented. When the mapping between the abstract and the concrete models is explicitly (and formally) documented, assertions written in the abstract model's vocabulary can be translated, following the representation mapping, in order to analyze if they hold in the implementation. Alternatively, instances of concrete models can be abstracted according to the abstraction mapping so that abstract properties can be tested on them.

The contribution of this article is to clarify the abstraction/refinement relationship in UML models, providing basis for tools supporting the refinement driven modeling process. PAMPERO is an evidence of the feasibility of the proposal.

References

1. Derrick, J. and Boiten,E. Refinement in Z and Object-Z. Foundation and Advanced Applications. FACIT, Springer, 2001
2. Dijkstra, E.W., A Discipline of Programming. Prentice-Hall, 1976.
3. IBM, The Eclipse Project. Home Page. Copyright IBM Corp. and others, 2000-2004. http://www.eclipse.org/.
4. Lano,K. The B Language and Method. FACIT. Springer, 1996.
5. OMG. The Unified Modeling Language Specification – Version 1.5, UML Specification, revised by the OMG, http://www.omg.org, March 2003.
6. Pons, C., Pérez,G., Giandini, R., Kutsche, Ralf-D. Understanding Refinement and Specialization in the UML. 2nd International Workshop on MAnaging SPEcialization/Generalization Hierarchies (MASPEGHI). In IEEE ASE 2003, Canada.
7. Pons, C., Pérez., G. and Kutsche, R-D. Traceability across refinement steps in UML Modeling. Workshop in Software Model Engineering, 7th International Conference on the UML, October 11, 2004, Lisbon, Portugal.
8. Richters Mark and Gogolla Martin. Validating UML Models and OCL Constraints. Springer-Verlag, 2000. http://www.db.informatik.uni-remen.de/projects/USE.