TOWARDS A KNOWLEDGE BASE TO IMPROVE REUSABILITY OF DESIGN PATTERN

Cédric Bouhours, Hervé Leblanc and Christian Percebois
IRIT – MACAO team (Models, Aspects, and Components for Object-oriented Architectures)
Université Paul Sabatier
118 Route de Narbonne
F-31062 Toulouse Cedex 9

Keywords: Software architectures, Design patterns, Design review.

Abstract: In this paper, we propose to take directly into account the knowledge of experts during a design review activity. Such activity requires an ability to analyze and to transform models, in particular to inject design patterns. Our approach consists in identifying model fragments which can be replaced by design patterns. We name these fragments "alternative models" because they solve the same problem as the pattern, but with a more complex or different structure than the pattern. In order to classify and to explain the design defects of this alternative models base, we propose the concept of strong point. A strong point is a key design feature which permits the pattern to resolve a problem most efficiently.

1 CONTEXT

The emergent MDE community, aiming to give a productive character to models, has proposed model-driven process development. However, these processes should be able to reuse the knowledge of experts generally expressed in terms of analysis (Fowler, 1997), design (Gamma et al., 1995) or architectural (Buschmann, 1996) patterns approved by the community. Given the existence of "code review" activities (Dunsmore, 1998) in some development processes, we would like to introduce a "design review" activity, directed by design patterns, to improve object model quality. We limit our approach to design patterns, because we consider that analysis patterns are business domain specific, and the use of architectural patterns must be planned before the design stage.

This activity may be decomposed into four sub-activities. First, model preparation puts the model to review in a minimal quality, for example, to impose that one class implements at least one interface or to impose that all attributes are *private* or *protected*. Then, a research based on structural and behavioral similarities determines the model fragments which may be substitutable with a pattern. Next, a design expert validates the patterns proposed to substitute the fragments found in the previous sub-activity, considering the designers' intentions. Lastly, the designers integrate the validated patterns into their models. This integration is dealt with by automatic parameterized transformations.

Up to now, in spite of the efforts to improve reusability of design patterns, thanks to assistance tools to guide pattern integration in models by precise modeling (Guennec, 2000) (France, 2004), and thanks to pattern wizards dedicated to integrate patterns by code refactorings (Eden, 1997) (O'Cinnéide, 1999), we do not find a model inspecting tool that urges the use of patterns in the most automatic way possible. To do the detection of the substitutable fragments, we use a match method. Rather than using an approximate design-pattern match detection based on a similarity research (Arcelli Fontana, 2004), we do exact pattern matching of models substitutable with a design pattern. Then, we seek a set of substitutable models for each structural pattern proposed by Gamma *et al.* We name these models "alternative models". According to the taxonomy proposed by Chikofsky and Cross (Chikofsky, 1990), the implemented technique can be connected to a redocumentation technique so as to permit model restructuring.

In a first part, we present the concept of alternative model, and how to collect and to use

Bouhours C., Leblanc H. and Percebois C. (2007).
TOWARDS A KNOWLEDGE BASE TO IMPROVE REUSABILITY OF DESIGN PATTERN.
In *Proceedings of the Second International Conference on Software and Data Technologies - SE*, pages 421-424
DOI: 10.5220/0001325104210424

them. Next, we present a problem solvable with the *Composite* design pattern, and its corresponding alternative models. Moreover, to characterize the design defects of these alternative models, we deduce them with some modification to the pattern structure.

2 ALTERNATIVE MODELS

An alternative model is a model which solves the same problem as the pattern, but with a more complex or different structure than the pattern. Therefore, in agreement with the hypotheses on design patterns and class design defects (Guéhéneuc, 2001), it is a candidate model for substitution with a pattern.

Each alternative model is characterized like a pattern. A set of structural features is associated with each alternative model role. For the moment, these features concern inter-class relations only: *i.e.*, associations, generalizations and aggregation-composition links, but neither interfaces nor class semantics. We deduce these features both associating corresponding pattern roles with each class of alternative models, and studying their structure (Bouhours, 2006).

2.1 Discovery

For this study, we choose to collect a set of models which do not use any patterns. If these models solve a problem solvable with a particular pattern, they may be considered as an alternative model.

We have organized an experiment which consists of the design of the seven standard problems solvable with the seven GoF structural patterns, in UML notation. We use the examples presented in the "motivations" section of the GoF catalogue, when they are relevant. Each problem admits a solution using a pattern, but participants have solved these problems without any pattern knowledge. From three hundred models obtained, we have selected fifteen of them which present significant structural variants with patterns under consideration, the others were either incorrect or duplicated design. We consider these models valid because they permit a solution to the problems under study, namely, they implement functionalities required by this problem. Each model obtained (between two and six for each pattern) constitutes a plausible alternative to just one pattern.

2.2 Use

This collection method allows us to constitute a catalogue for each pattern and its associated alternative models that we consider as potentially bad design practices. One entry of the catalogue corresponds to one pattern with its alternative models classified by the strong points of the pattern. A strong point is a key design feature which permits the pattern to resolve a problem most efficiently. For example, the Composite pattern resolves the problem: "*How to compose and use object hierarchies as simply as possible for a client in keeping the extensibility possibilities on components?*". So the two strong points for this pattern are "*uniform processing*" and "*decoupling and extensibility*".

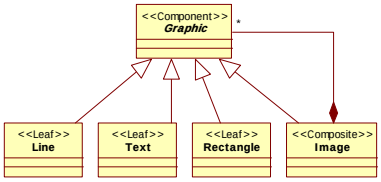
The strong points are the "essence" of the patterns. They are characterized by criteria of object-oriented architecture and software engineering quality, partially deduced from the "consequences" section of the GoF catalogue and from the study of the design defects of alternative models. As pattern injection may alter some object oriented metrics (Huston, 2001), they allow us to compute dedicated pattern metrics to classify the alternative models and to help the estimation of the pertinence of pattern injection in a design model.

In order to characterize the design defects, we deduce the alternative models in perturbing the strong points of each pattern. A perturbation may either delete a strong point or simply damage it. So to specify the degree of damage, we add sub-features for some strong points. Moreover, thanks to these perturbations, we should build new alternative models not taken from experiments. And, if we reverse these perturbations and if we apply them on alternative models, we would deduce a sequence of structural refactoring operations (Sunyé, 2001) that automatically perform the pattern integration in the models to review.

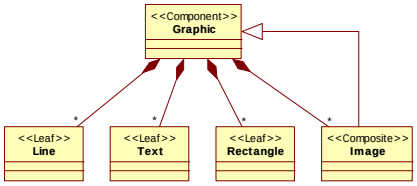
3 COMPOSITE ALTERNATIVE MODELS

The problem "*Design a system enabling to draw a graphic image: a graphic image is composed of lines, rectangles, texts and images. An image may be composed of other images, lines, rectangles and texts.*" may be solvable with the Composite design pattern.

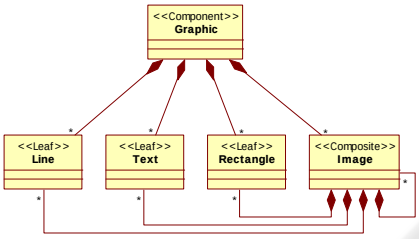
The below figures represent this problem instantiation *(Model 0)* and the five alternative models *(Models 1 to 5)* taken from our experiment.



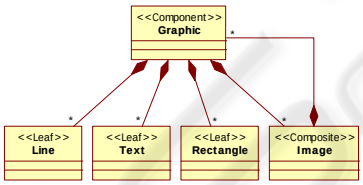
0: Composite problem instantiation



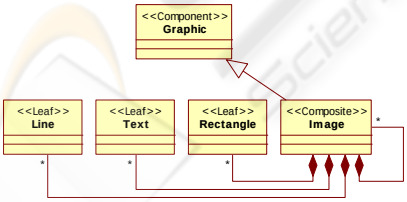
1: Development of the composition on «Component»



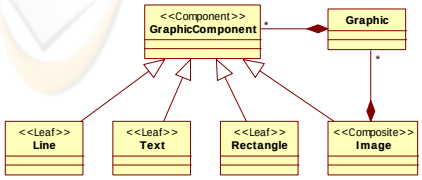
2: Development of the composition on «Component» and «Composite»



3: Recursive composition



4: Development of the composition on «Composite»



5: Indirect composition on «Composite»

Thanks to the analysis of these alternative models, we find two strong points with their sub-features for the *Composite* pattern:

1 Decoupling and extensibility.
  - 1.1 Maximal factorization of the composition.
  - 1.2 Addition or removal of a leaf does not need code modification.
  - 1.3 Addition or removal of a composite does not need code modification.

2 Uniform processing.
  - 2.1 Uniform processing on operations of composed object.
  - 2.2 Uniform processing on composition managing.
  - 2.3 Unique access point for the client.

In order to validate these strong points, we deduce now each alternative model in perturbing the strong points:

First, if we use the instantiated pattern *(Model 0)* and if we replace the inheritance links by inverted composition links and composition links by inheritance links, we obtain a first alternative model *(Model 1)* where the first strong point is deleted and the second is damaged on the first sub-feature. Indeed, without the inheritance link, there is no guarantee that the processing produces conformity between «Composite» and «Leaf» classes.

In the first alternative model *(Model 1)*, if we replace the inheritance link by its composition equivalence, we keep on damaging the second strong point, in deleting the second sub-feature. Indeed, this alternative model *(Model 2)* has only composition relationships that impose to manage the composition in «Component» and «Composite» classes.

From the second alternative model *(Model 2)*, if we factor «Composite» composition links on «Component», we obtain an alternative model *(Model 3)* without a single strong point. Indeed, this factorization adds a cycle between «Composite» and «Component» that produces two access points for the client.

From the instantiated pattern *(Model 0)*, by developing composition from «Composite» to «Component» over every sub-class of «Component», we obtain a new alternative model *(Model 4)*. The first strong point is deleted, but the second is only damaged on the first sub-feature. Although the access point is not in a good place, namely on «Composite», we consider that the third sub-feature is present.

Lastly, if we add an intermediary class between «Composite» and «Component» in the problem

instantiation *(Model 0)*, we damage the two strong points. In the first, the damage is due to the first sub-feature: the factorization is not maximal. For the second strong point, the composition management is done in «Composite» and "Graphic" classes, and there are two redundant access points *(Model 5)*.

Table 1 resumes the state of every strong point for each Composite alternative model. For each strong point, we represent sub-features in this table with a "+" if it is present and with a "-" if it is deleted. In a first approach, we define a "quality score" simply based on strong points. We consider the strong points qualitatively equivalent, and compute the metrics with their degree of perturbation.

4 CONCLUSION

In order to validate our approach, we have applied OCL rules on UML models. So, it has been necessary to implement detection of each alternative model by several rules. When a rule is not validated in the model, the NEPTUNE (Neptune, 2003) platform returns the context of the error, which is the model fragment substitutable by a pattern. In a first attempt, we have applied these rules on industrial models and then on OMG meta-models.

To increase the range of our catalog, which is, for now, only constituted of alternative models taken by our experiments, we are currently developing a collaborative web site allowing to share knowledge about object misconception.

REFERENCES

Arcelli Fontana F., Raibulet C., Tisato F., "Design Pattern Recognition", in *Proceedings of the ISCA 13th IASSE*, Nice, France, July 1-3, 2004, pages 290-295.

Bouhours C., Leblanc H., Percebois C., "Structural variants detection for design pattern instantiation", in *1st International Workshop on DPD4RE*, Benevento, Italy, October 2006.

Buschmann F., Meunier R., Rohnert H., Sommerlad P., Stal M., "Pattern-Oriented Software Architecture", John Wiley \& Sons, August 1996.

Chikofsky E. J., Cross J. H., "Reverse engineering and design recovery: A taxonomy", in *IEEE Software*, 7(1), page 13-17, January 1990.

O'Cinnéide M., Nixon P., "A Methodology for the Automated Introduction of Design Patterns", in *ICSM '99: Proceedings of the IEEE International Conference on Software Maintenance*, IEEE Computer Society, 1999, pages 463.

Dunsmore A.P., Comprehension and Visualisation of Object-Oriented code for Inspections, Technical Report, EFoCS-33-98, Computer Science Department, University of Strathclyde, 1998.

Eden A. H., Yehudai A., Gil J., "Precise specification and automatic application of design patterns", in *ASE '97: Proceedings of the 12th international conference on Automated software engineering (formerly: KBSE)*, IEEE Computer Society, 1997, pages 143.

Fowler M., "Analysis patterns: reusable objects models", Addison Wesley Longman Publishing Co, Inc., 1997

France R. B., Kim D., Ghosh S., Song E., "A UML-Based Pattern Specification Technique", in *IEEE Trans. Softw. Eng.*, IEEE Press, 2004, 30, pages 193-206.

Gamma E., Helm R., Johnson R., Vlissides J., "Design Patterns: Elements of Reusable Object-Oriented Software", Addison Wesley Professional, 1995.

Guennec A. L., Sunyé G., Jézéquel, J., "Precise Modeling of Design Patterns", in *UML*, 2000, pages 482-496.

Guéhéneuc Y. G., Albin-Amiot. H., "Using Design Patterns and Constraints to Automate the Detection and Correction of Inter-Class Design Defects", in *Proceedings conference TOOLS*, July 2001, pages 296-305.

Huston B., "The effects of design pattern application on metric scores", in *Journal of Systems and Software*, 58(3), Elsevier Science, September 15, 2001, pages 261-269.

Neptune, [w] http://neptune.irit.fr, 2003.

Sunyé G., Pollet D., Le Traon Y., Jézéquel J.M., "Refactoring UML Models", in *Proceedings of UML 2001*, pages 134-148.

Table 1: State of the Strong Points of the Composite Alternative Models.

		Alternative model number									
Strong point	Sub-features	5		1		4		2		3	
1	1	-	2/3	-	0	-	0	-	0	-	0
	2	+		-		-		-		-	
	3	+		-		-		-		-	
2	1	+	1/3	-	2/3	-	2/3	-	1/3	-	0
	2	-		+		+		-		-	
	3	-		+		+		+		-	
Quality score:		50%		33.3%		33.3%		16.7%		0%	