Objectifying a Merise analysis using transformation rules

Annya Romanczuk, Isabelle Borne

Ecole des Mines de Nantes
La chantrerie, 4 rue Alfred Kastler
44070 Nantes Cedex 03, France
Email: Annya.Romanczuk, Isabelle.Borne@emn.fr

Abstract

The main goal of this project was to design a semi-automatic tool which, from a Merise analysis schema, produces an OMT analysis schema. This work was based on a real-world case provided by a company wishing to migrate to object-oriented techniques in order to improve its maintenance process and software quality. We extracted translation and transformation rules, in a systematic way, from the existing Merise analysis of the case study. A comparison of both methods led to define a set of automatic rules and a set of concepts for which there is no correspondence between the two analysis representations.

Keywords: Migration, legacy systems, restructuring, objectifying.

1 Introduction

Legacy management information systems, implemented in COBOL, need to be modernized in order to gain control of maintenance costs, and to be adaptable to various platforms. We report here a study carried out for a French company wanting to improve its maintenance process and software quality by using object-oriented techniques. This company produces softwares for large-scale distribution, i.e stock management, purchase and sale management, etc.

Our study consisted in designing a semi-automatic tool which produces an object-oriented analysis, based on OMT [Rum91], from a classical analysis, based on a Merise analysis schema. Merise [Tar83] is a French *systemic* method of the same kind of Axial [Pel86], and OMT is one of the most used methods in France. More precisely the data of our study are the analysis and design of the stock management system in the Merise formalism.

Our project, in a larger context, addresses the area of reengineering, because a complete reengineering cycle is necessary for a real migration of the legacy system. The reengineering cycle we envisaged [DoD92] encompasses the following phases: reverse-engineering, restructuring, redocumentation, forward engineering.

At the moment our contribution is located in a first step of the restructuring phase [Chi90], that is a systematic transformation of the Merise representation towards an OMT representation. A further restructuring step will consist in improving the obtained object-oriented representation. Indeed, the result of the first step is not optimal, and will need to be improved in order to produce a flexible architecture for the reengineered system.

2 The restructuring process

This work tackles the design level of the legacy application and not the code level such as most of the COBOL migration experiences [Rug93]. The reverse-engineering phase is not necessary in our case since the analysis of the legacy system is provided. The restructuring method we proposed at the analysis level involves three steps:

- The selection of the candidate components for the transformation. This means the selection of concepts extracted from the Merise analysis of the legacy system, and which correspond with Merise concepts (entity, association, action, event, operation,...).
- The choice of the translation or transformation rules allowing to go from the Merise level to the OMT level, according to the candidate components found in the first step.
- The transformation of the basic Merise structures to the suitable OMT structures, preserving the original functionalities.

A systematic study of the Merise models (the diagrams and their semantics) of the legacy application analysis and of their possible correspondences in the OMT method allowed to bring out the transformation process between the two kinds of analysis schemas. Although this study were based

Merise	**OMT**
Data Conceptual Model (MCD) entity relationships diagram rule example: a type of entity becomes a class	Object Model object/class diagram
Flow graph actor-flow graph rule example: an actor becomes a class	Scenario event trace diagram
Process Conceptual Model (MCT) MCT diagram rule example: the initial event becomes a particular state	Dynamic Model state diagram
Process Organizational Model (MOT) MOT diagram rule example: an operation becomes a process	Functional Model data flow diagram

Table 1: Correspondences between the methods' models

on a classical stock management system analysis, the deduced transformation rules are not domain dependent.

The set of the formal translation rules is composed of automatic rules and semi-automatic rules for which interactions with the user are required. Indeed, as shown in the table, the transformation of a Merise data conceptual model to an OMT object model, or of a flow graph to an OMT event trace diagram, can be automatic. There is a direct correspondence between the involved concepts.

On the other hand, the transformation of a Merise process conceptual model (MCT) to an OMT state diagram cannot be automatic. A state diagram concerns only one specific sub-system, whereas a MCT contains various treatments from various sub-systems. Moreover, a MCT describes actions that cannot be transformed into a state diagram.

In the same way, the translation of a Merise MOT to the OMT functional model is difficult to automate. For example, an actor from a MOT will have a suitable structure in a data flow diagram, only if the user adds a new storage for this actor. One of the main point was to not lose information during the translations, that is not possible with an automatic transformation of the last two models.

3 Conclusions

The restructuring of a Merise analysis towards an OMT analysis is possible with a semi-automatic translation tool. The user will have to interact with the system to take decisions between different transformations or to add or verify information in specific cases.

Now, the next step of this study is to implement the transformation rules in a meta-tool, which will generate software engineering tools instantiated with the two methods, Merise and OMT.

The development of such semi-automatic tools based on translation rules is one solution for the migration of the legacy systems at the design level. Working at the design level allows us to be independent of a specific platform, and to be compatible with other existing tools.

References

[Chi90] Chikofsky E.J., Cross II J.H. *Reverse Engineering and Design Recovery: A Taxonomy,.* IEEE Software 7(1), 13-17, 1990.

[DoD92] DoD workshop. *Reengineering Definitions.* Santa Barbara 1992. http://www.stsc.hill.af.mil/RENG/DEFIN.HTML

[Pel86] Pellaumail P. *la méthode Axial.* Editions d'Organisation, Paris 1986.

[Rug93] Rugaber S., Doddapaneni S. *The Transition of Application Programs from COBOL to a Fourth Generation Language.* Conference on Software Maintenance, Montreal, Canada, September 27-30, 1993.

[Rum91] Rumbaugh J., Blaha M., Premerlami W., Eddy F. and Lorensen W. *Object-Oriented Modeling and Design.* Englewood Cliffs, NJ: Prentice-hall, 1991.

[Tar83] Tardieu H., Rochfeld A., Colleti R. *La méthode Merise : Principes et Outils.* Les Editions d'Organisation, Paris, 1983.