Round-Trip Engineering with Design Patterns, UML, Java and C++

Wilhelm Schäfer
Dep. of CS, University of Paderborn
Warburger Str. 100
33098 Paderborn, Germany
++49+5251 60 3312
wilhelm@uni-paderborn.de

Albert Zündorf
Dep. of CS,University of Paderborn
Warburger Str. 100
33098 Paderborn, Germany
++49+5251 60 3310
zuendorf@uni-paderborn.de

ABSTRACT
The tutorial presents the state-of-the-art in methodologies and tools for round-trip-engineering of object-oriented software systems. This covers not only (UML) class diagrams but also behavior diagrams like message sequence charts, collaboration diagrams, state charts, and activity diagrams.
In addition, round-trip engineering with standard design patterns is addressed. This covers design by combining design patterns, implementation of design patterns, including code generation, and the recognition of standard design patterns in code fragments.

Keywords
Round-Trip Engineering, UML, Design Patterns

1 OVERVIEW

The tutorial presents the state-of-the-art in methodologies and tools for round-trip-engineering of object-oriented software systems. This covers not only (UML) class diagrams but also behavior diagrams like message sequence charts, collaboration diagrams, state charts, and activity diagrams.

In addition, round-trip engineering with standard design patterns is addressed. This covers design by combining design patterns, implementation of design patterns (including code generation), and the recognition of standard design patterns in code fragments.

First, we present the state-of-the-art in deriving semi-automatically implementations (in Java, C++ or the like) from design documents. For class diagrams this is widely known and supported by various OO CASE tools (e.g. [Rose96]). But even for class diagrams a correct implementation of associations versus aggregations versus compositions considering additional aspects like qualifications and constraints (e.g. {sorted} or {ordered}) is not trivial and not sufficiently supported by current CASE tools. The tutorial will summarize state-of-the-art solutions for this problem based on libraries of uniform collection classes ([Rose96], [SL95], [JGL98], [FNTZ98]). This covers, in particular, techniques dealing with dangling references.

Second, the tutorial will show the state-of-the-art in the translation of UML behavior diagrams to Java or C++ code. Generally, this translation is not well understood and very difficult since the UML behavior diagrams may only describe sample scenarios, may be quite incomplete, or may use pseudo code or natural language descriptions. But, there are several recent approaches based on a restricted use of behavior diagrams for dedicated purposes like system structure, process coordination, object consistency constraints, complex computations, complex object structure changes, or just method behavior. These approaches enable the generation of large amounts of code from behavioural design diagrams.

Third, design patterns provide additional information and means for the derivation of a valid implementation. Actually, design patterns assign a specific semantics to a group of classes, attributes, associations, and last but not least methods. In many cases this semantics correspond to certain standard implementations. We will show the state-of-the-art of pattern based design by combining and adapting design patterns and by generating implementations from design patterns. Finally, the tutorial adresses recognition of occurrences of design patterns in code fragments for re-engineering and round-trip purposes ([FMW97], [KS98], [SW98], [JSZ97]).

2 KEY LEARNING OBJECTIVES

Participants will get an overview of standard correspondences between design and implementation. They will learn a certain style in using different kinds of design diagrams for dedicated purposes. This style will enable round-trip engineering from design to code generation including manual modifications of the code which are propagated back automatically into the design. The knowledge how design and code correspond to each other will increase the understanding of design techniques and of the relationships between the different kinds of design diagrams and their usage. The tutorial will enhance the abilities to produce and to understand designs and to implement them.

3 TARGET AUDIENCE

The Tutorial targets industrial practitioners which are interested in methods and tools that facilitate the

ICSE '99 Los Angeles CA

implementation and maintenance phases of developing Java or C++ applications. It also adresses CASE tool builders and especially researchers with an interest in precise semantics definitions of UML and design patterns.

4 CONTENTS

1) Round Trip Engineering Introduction

2) From UML to Java and Back Again

- class diagrams <==> Java
- collaboration diagrams <==> Java
- message-sequence charts <==> Java
- state charts <==> Java
- activity diagrams <==> Java
- story diagrams <==> Java

3) Design Pattern <==> Java

4) Tools demonstrations

5) Conclusions

REFERENCES

[FMW97] G. Florijn, M. Meijers, P. van Winsen: Tool Support for Object-Oriented Patterns. in proceedings ECOOP'97 1997, pp. 472-495, 1997.

[FNTZ98] T. Fischer, J. Niere, L. Torunski, A. Zündorf: Story Diagrams: A new Graph Grammar Language based on the Unified Modelling Language and Java; accepted for proceedings of the 6th International Workshop on Theory and Application of Graph Transformations (TAGT'98), Paderborn, Germany, 1998.

[GHJV95] E. Gamma, R. Helm, R. Johnson, J.Vlissides. Design Pattern (Elements of Reusable Object-Oriented Software). Addison Wesley. ISBN 0-201-63361-2, 1995.

[JGL98] Technical reference of the generic collection library for Java http://www.objectspace.com/jgl/

[JSZ96] J. Jahnke, W. Schäfer, A. Zündorf: A Design Environment for Migrating Relational to Object-Oriented Database Systems. Proc. of the Int. Conf. on Software Maintenance, ICSM'96, Monterey, CA November 1996, USA.

[JSZ97] J.-H. Jahnke, W. Schäfer, and A. Zündorf: Generic Fuzzy Reasoning Nets as a basis for reverse engineering relational database applications; Proc. of European Software Engineering Conference (ESEC/FSE), September 1997, Zuerich. Springer (LNCS 1301), 1997.

[JZ98] J.-H. Jahnke and A. Zündorf: Specification and Implementation of a Distributed Planning and Information System for Courses based on Story Driven Modelling. In Proceedings of the Ninth International Workshop on Software Specification and Design, IWSSD9, April 16-18, Ise-Shima, Japan, IEEE Computer Society, pp. 77-86, ISBN 0-8186-8439-9, 1998

[KS98] R. Keller, R. Schauer: Design Components: Towards Software Composition at the Design Level; in proceedings 20th International Conference on Software Engineering (ICSE'98), Kyoto, pp. 302-311, 1998.

[NSS96] O. Neumann, S. Sachweh, W. Schäfer: A High-Level Object-Oriented Specification Language for Configuration Management and Tool Integration. Proc. of the 5th Europ. Workshop on Software Process Technology, EWSPT'96, Nancy, France, October 1996.

[Rose96] Rational Rose: Round-Trip Engineering with Rational Rose/C++; Techncial Documentation to Rational Rose 4.0, Rational Software Corporation, Santa Clara, 1996.

[SL95] A. Stepanov, M. Lee: The Standard Template Library; Technical Report, Hewlett Packard Laboritries, Palo Alto, cf. http://www-leland.stanford.edu/~iburrell/cpp/stl.html, 1995.

[SW98] J. Seemann, J. Wolff von Gudenberg: Pattern-Based Design Recovery of Java Software; proceeding of acm Foundations of Software Engineering (FSE'98), 1998.

[UML97] UML Notation Guide vers. 1.1. Rational Software, Microsoft, Hewlett-Packard, Oracle, Sterling Software, MCI Systemhouse, Unisys, ICON Computing, IntelliCorp, i-Logix, IBM, ObjecTime, Platinum Technology, Ptech, Taskon, Reich Technologies, Softeam, 1997.