

Niels Streekmann

Clustering-Based Support for Software Architecture Restructuring

# VIEWEG+TEUBNER RESEARCH

## **Software Engineering Research**

Herausgeber/Editor:

Prof. Dr. Wilhelm Hasselbring

Im Software Engineering wird traditionell ein Fokus auf den Prozess der Konstruktion von Softwaresystemen gelegt. Der Betrieb von Systemen, die kontinuierlich Dienste mit einer geforderten Qualität bieten müssen, stellt eine ebenso große Herausforderung dar. Ziel der Reihe Software Engineering Research ist es, innovative Techniken und Methoden für die Entwicklung und den Betrieb von nachhaltigen Softwaresystemen vorzustellen.

Traditionally, software engineering focuses on the process of constructing and evolving software systems. The operation of systems that are expected to continuously provide services with required quality properties is another great challenge. It is the goal of the Series Software Engineering Research to present innovative techniques and methods for engineering and operating sustainable software systems.

Niels Streekmann

# Clustering-Based Support for Software Architecture Restructuring

With a foreword by Prof. Dr. Wilhelm Hasselbring

VIEWEG+TEUBNER RESEARCH

Bibliographic information published by the Deutsche Nationalbibliothek  
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie;  
detailed bibliographic data are available in the Internet at <http://dnb.d-nb.de>.

Dissertation University of Kiel, 2011

1st Edition 2012

All rights reserved

© Vieweg+Teubner Verlag | Springer Fachmedien Wiesbaden GmbH 2012

Editorial Office: Ute Wrasmann | Anita Wilke

Vieweg+Teubner Verlag is a brand of Springer Fachmedien.

Springer Fachmedien is part of Springer Science+Business Media.

[www.viewegteubner.de](http://www.viewegteubner.de)



No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the copyright holder.

Registered and/or industrial names, trade names, trade descriptions etc. cited in this publication are part of the law for trade-mark protection and may not be used free in any form or by any means even if this is not specifically marked.

Cover design: KünkelLopka Medienentwicklung, Heidelberg

Printed on acid-free paper

Printed in Germany

ISBN 978-3-8348-1953-6

# Foreword

Successful software systems are long-living systems. During their lifetime the underlying development platforms, programming languages and software architectures eventually become obsolete. When this happens, these software systems are often called *legacy systems*: those engineers responsible for maintaining the systems are usually not the engineers who built the systems. An emerging question is how the resulting requirements for modernization may be achieved. As a first step, it is useful to reconstruct the existing software architecture via reverse engineering techniques. If we aim at a real modernization, just a transformation of legacy code into the syntax of a new programming language will not lead to a maintainable system. Therefore, as a second step, the target architecture for the modernized software system needs to be defined. Then, a migration strategy from the legacy systems architecture toward the chosen target architecture is needed.

Niels Streekmann introduces in this research work a new approach to restructuring software architectures, meant to support engineers with modernizing existing legacy systems. From the above-mentioned modernization tasks, the work presented in this book focuses on finding an appropriate migration strategy on the architectural level. To support this task, a semi-automatic method is introduced which groups elements of the reconstructed legacy architecture via clustering algorithms. These clustered groups of source code elements are then mapped to components of the target architecture.

Besides the presented conceptual research work, this book reports on an extensive experimental evaluation of the restructuring approach. In particular, the reconstruction of the architectural elements of a large industrial system is presented. Thus, this book will be a valuable resource for both researchers and practitioners interested in restructuring (legacy) software systems.

Wilhelm Hasselbring

# Acknowledgements

During the past five years many people influenced this thesis by providing helpful comments, criticism and reviews to my ideas and publications. I would like to express my deepest gratitude to all of them, even though they may not be mentioned explicitly in the following.

First of all I would like to thank Willi Hasselbring for supervising my thesis and giving me useful hints and suggestions from the selection of the topic to the final completion. I also thank the second examiner of this thesis Andreas Winter for his extensive remarks on reengineering and graph clustering.

My former employer, the OFFIS - Institute for Information Technology, had a vital part in the creation of this thesis by giving me the time to develop and try out my ideas in the context of interesting research projects and to write down this thesis. I thank all my former colleagues and especially all current and former members of the Software Engineering for Business Information Systems Group for the pleasant and productive working atmosphere. I very much appreciated working with you.

Furthermore, I would like to thank all participants of the PhD student seminars (*Doktorandenrunden*) of the software engineering groups of the universities in Oldenburg and Kiel and the former OFFIS division Business Information Management. These seminars have been a source for many helpful remarks and inspiring discussions. I also thank all participants of the talks I gave at other universities, workshops, and conferences and the reviewers of my submitted papers for their expert feedback. Moreover, I would like to thank all people who helped me during the evaluation of my approach by providing assistance, experiment subjects, tools and advice.

However, my biggest thanks go to my family. My parents Ulla and Rolf Streekmann, who made all this possible with their long lasting financial and above all personal support, and my girlfriend Silvia, who endured the temporal constraints in the past five years and who was always there for me with much understanding.

Niels Streekmann

# Abstract

The modernisation of existing software systems is an important topic in software engineering research and practice. A part of the modernisation of software systems is the restructuring of their architecture. This has to be done in numerous contexts, including the evolution to service-oriented architectures, the re-establishment of the maintainability of a system or the smooth migration of a system to a new development environment. Architecture restructurings are coarse-grained changes to the internal structure of the system that are performed in temporally limited projects. The planning of the transfer of an existing implementation to the target architecture of a system is currently a mostly manual task. While the analysis of the existing system is supported by e.g. architecture reconstruction approaches, the actual restructuring process is not supported by current approaches.

The MARE approach, which is introduced in this thesis, was developed to provide support for the stepwise restructuring of the implementation towards a target architecture. MARE supports architecture restructurings by semi-automatically creating a complete mapping of elements of the existing implementation to components of the target architecture. The creation of the mapping bases on explicit knowledge about the target architecture and its decomposition criteria. MARE employs graph clustering to implement the creation of the complete mapping.

The MARE Method describes an iterative process model for the overall architecture restructuring process. It emphasises the target architecture as the basis for the architecture restructuring. The iterations of the process model allow for a stepwise restructuring of the system and the integration of human influence on the result of MARE.

The clustering algorithm employed by MARE to create the complete mapping bases on agglomerative hierarchical clustering. It is adjusted to incorporate knowledge about the target architecture. The decomposition criteria are considered by the definition of weights for the different types of dependencies that relate the elements of the existing implementation.

The MARE approach was evaluated in three case studies. These examined the application of MARE in small and middle-sized open source projects as well as for an industrial system with 3.5 million lines of code. The main goal of the evaluation is to show the quality and stability of the clustering algorithm. It furthermore shows the influence factors for the creation of the complete mapping.

# Contents

- Foreword** **V**
  
- Acknowledgements** **VII**
  
- Abstract** **IX**
  
- List of Figures** **XV**
  
- List of Tables** **XIX**
  
- 1 Introduction** **1**
  - 1.1 Motivation . . . . . 1
  - 1.2 Solution Approach . . . . . 3
  - 1.3 Scientific Contribution . . . . . 4
  - 1.4 Overview . . . . . 5
  
- I Foundations** **7**
  
- 2 Software Architecture** **9**
  - 2.1 Software Architecture Basics . . . . . 9
  - 2.2 Detailed Design . . . . . 14
  - 2.3 Architectural Styles . . . . . 15
  - 2.4 Architecture Metrics . . . . . 19
  - 2.5 Architectural Erosion . . . . . 21
  
- 3 Software Evolution and Modernisation** **23**
  - 3.1 Reengineering . . . . . 25
  - 3.2 Migration . . . . . 26
  - 3.3 Reverse Engineering . . . . . 27
  - 3.4 Architecture Reconstruction . . . . . 30
  - 3.5 Restructuring . . . . . 37



<b>4</b>	<b>Graph Clustering</b>	<b>45</b>
4.1	Graphs . . . . .	45
4.2	Graph Clusterings . . . . .	45
4.3	Clustering Algorithms . . . . .	47
4.4	Applications of Graph Clustering . . . . .	51
<b>5</b>	<b>Model-Driven Software Development</b>	<b>55</b>
5.1	Modelling Concepts . . . . .	56
5.2	Model Transformations . . . . .	59
5.3	Architecture in Model-Driven Software Development . . . . .	60
5.4	Interoperability Metamodels for Reengineering . . . . .	62
5.5	Model-Driven Reengineering Approaches . . . . .	67
<b>II</b>	<b>Clustering-Based Support for Software Architecture Re- structuring</b>	<b>71</b>
<b>6</b>	<b>MARE Approach</b>	<b>73</b>
6.1	Problem Description . . . . .	73
6.2	Application Scenarios . . . . .	74
6.3	Goals of MARE . . . . .	77
6.4	Research Questions . . . . .	79
6.5	Definitions . . . . .	81
<b>7</b>	<b>MARE Method</b>	<b>83</b>
7.1	Embedding in the Reengineering Process . . . . .	84
7.2	MARE Clustering Activities . . . . .	101
7.3	MARE Metamodels . . . . .	108
7.4	Summary . . . . .	116
<b>8</b>	<b>MARE Clustering</b>	<b>119</b>
8.1	Goals and Criteria of the Clustering . . . . .	119
8.2	Target-Architecture-Driven Clustering . . . . .	121
8.3	Impact of Dependencies on the Clustering . . . . .	131
8.4	Arbitrary Decisions . . . . .	133
8.5	Result Validation . . . . .	136
8.6	Summary . . . . .	142

**III Evaluation 145**

**9 Evaluation Methods 147**

- 9.1 Goals of the Evaluation . . . . . 147
- 9.2 GQM Plan . . . . . 149

**10 Case Studies 157**

- 10.1 Preliminary Case Study . . . . . 157
- 10.2 Open Source Case Study . . . . . 164
- 10.3 Industrial Case Study . . . . . 186
- 10.4 Summary . . . . . 194

**11 Related Work 197**

- 11.1 Architecture Restructuring . . . . . 197
- 11.2 Architecture Reconstruction . . . . . 200
- 11.3 Alternatives to Hierarchical Clustering . . . . . 206
- 11.4 Summary . . . . . 208

**IV Conclusion 211**

**12 Results 213**

- 12.1 MARE Method . . . . . 213
- 12.2 MARE Clustering Algorithm . . . . . 215
- 12.3 Evaluation . . . . . 216

**13 Future Work 219**

- 13.1 Automated Restructuring of the Implementation . . . . . 219
- 13.2 Definition of Cohesion for Coarse-Grained Modules . . . . . 220
- 13.3 Further Evaluation of MARE . . . . . 220

**14 Concluding Remarks 223**

**Bibliography 225**

# List of Figures

2.1	Mapping Example . . . . .	15
2.2	LCOM Connectivity Example . . . . .	21
3.1	The Horseshoe Model (Source: Kazman et al. (1998)) . . . . .	25
3.2	Architecture Reconstruction Taxonomy (Source: Ducasse and Pollet (2009)) . . . . .	31
3.3	The Reflexion Method (Source: Murphy et al. (2001)) . . . . .	33
3.4	Architectural Reconstruction in Focus (Source: Medvidovic and Jakobac (2006)) . . . . .	35
3.5	Architecture Conformance Checking (Source: Lilienthal (2009)) . . . . .	36
3.6	Restructuring Concept Hierarchy . . . . .	38
4.1	Dendrogram Example . . . . .	48
5.1	MOF Layered Metamodel Architecture . . . . .	58
5.2	MOF Key Concepts (Source: Object Management Group (OMG) (2006)) . . . . .	59
5.3	Basic Concepts of Model Transformation (Source: Czarnecki and Helsen (2006)) . . . . .	60
5.4	Simplified Interrelations of MDA Models . . . . .	61
5.5	Top Level Classes of the DMM (Source: Lethbridge et al. (2004)) . . . . .	64
5.6	Layers and Packages of the KDM (Source: Object Management Group (OMG) (2009)) . . . . .	65
6.1	Structural View of the Dublo Pattern (Source: Teschke et al. (2004)) . . . . .	76
6.2	Complete Mapping Example . . . . .	78
7.1	Complete MARE Method Overview . . . . .	85
7.2	Model-Based Architecture Restructuring Process . . . . .	86
7.3	Mapping of Activities to the Horseshoe Model . . . . .	87
7.4	Hierarchical Refinement Example . . . . .	89
7.5	Vertical Extraction Example . . . . .	90

7.6	Quasar Enterprise: As-Is, To-Be and Ideal (Source: Engels et al. (2008)) . . . . .	92
7.7	Tolerated Dependencies . . . . .	93
7.8	Multiple Component Instance Example . . . . .	100
7.9	Dependencies to Multiple Component Instances . . . . .	101
7.10	Process with Main MARE Activities . . . . .	102
7.11	MARE Model Transformation . . . . .	102
7.12	MARE Configuration Activities . . . . .	103
7.13	MARE Clustering Activities . . . . .	108
7.14	Data Flow in MARE . . . . .	109
7.15	Metamodel of the Source System Model . . . . .	111
7.16	Concrete Source Elements for Java . . . . .	111
7.17	Target Architecture Metamodel . . . . .	112
7.18	Metamodel of the Initial Mapping . . . . .	114
7.19	Static Dependency Types in Java . . . . .	115
8.1	Graph Metamodel . . . . .	124
8.2	Adjusted Hierarchical Clustering Algorithm . . . . .	126
8.3	Example for Equation 8.5 . . . . .	131
8.4	Example for the Impact of Dependencies . . . . .	132
8.5	Example for Arbitrary Decisions . . . . .	134
8.6	Example for Several Clustering Runs . . . . .	136
8.7	Example for the Change of the Initial Mapping . . . . .	141
9.1	GQM Levels (Based on Basili et al. (1994)) . . . . .	150
10.1	Package Structure of the JPetStore . . . . .	158
10.2	Target Architecture of the JPetStore . . . . .	159
10.3	Reference Mapping of the JPetStore . . . . .	160
10.4	Package Structure of oAW . . . . .	165
10.5	Target Architecture of oAW . . . . .	166
10.6	Experiment 1: Results without Consolidating Iterations for Updating Rules . . . . .	171
10.7	Experiment 1: Results for 10 Consolidating Iterations for Updating Rules . . . . .	172
10.8	Experiment 1: Results without Consolidating Iterations for Composite Cluster Similarity Functions . . . . .	173
10.9	Experiment 1: Results for 10 Consolidating Iterations for Composite Cluster Similarity Functions . . . . .	173

- 10.10 Experiment 2: Results for Best Weights and Equal Weights . . . . . 176
- 10.11 Experiment 2: Comparison of PCMSE and MoJoFM . . . . . 176
- 10.12 Experiment 3: Results for Different Initial Mapping Strategies . . . . . 179
- 10.13 Experiment 4: Exemplary Sensitivity Analysis Results for the De-  
tailed Target Architecture . . . . . 181
- 10.14 Experiment 4: Exemplary Sensitivity Analysis Results for the Coarse  
Target Architecture . . . . . 183
- 10.15 Experiment 1: Results of the MARE Clustering Iterations . . . . . 190
- 10.16 Experiment 2: Result for the Extended Target Architecture . . . . . 192

# List of Tables

- 3.1 Maintenance Categories (Source: Abran et al. (2004)) . . . . . 24
- 6.1 Detailed Research Questions . . . . . 80
- 7.1 Dependency Type Weights (Source: Rayside et al. (2000)) . . . . . 106
- 10.1 JPetStore Dependency Weights . . . . . 161
- 10.2 NCMSE Results Without DaoConfig and BaseSqlMapDao . . . . . 162
- 10.3 NCMSE Results With DaoConfig and BaseSqlMapDao . . . . . 163
- 10.4 oAW Dependency Weights . . . . . 168
- 10.5 Weights for the Sensitivity Analysis . . . . . 170
- 10.6 Experiment 1: Quality for Updating Rules . . . . . 174
- 10.7 Experiment 1: Quality for Composite Cluster Similarity Functions . . 174
- 10.8 Experiment 1: Performance for Different Configurations . . . . . 175
- 10.9 Internal and External Dependencies . . . . . 178
- 10.10 Internal and External Dependencies for Different Initial Mappings . . 178
- 10.11 Results for MoJo and NCMSE for Best Weights and Different Initial Mappings . . . . . 179
- 10.12 Number of Different Source Elements in the Different Initial Mappings 180
- 10.13 Weights for the Industrial Case Study . . . . . 188
- 10.14 Similarity of Complete Mappings with and without Consolidating Iterations . . . . . 193
- 11.1 Comparison Between the Extended Reflexion Method and MARE . . 203