



Isela Macía Bertrán

On the Detection of Architecturally-Relevant Code Anomalies in Software Systems

Tese de Doutorado

Thesis presented to the Programa de Pós-Graduação em Informática of the Departamento de Informática, PUC-Rio as partial fulfillment of the requirements for the degree of Doutor em Informática.

Advisor: Prof. Arndt von Staa
Co-advisor: Prof. Alessandro Fabricio Garcia

Rio de Janeiro
March, 2013



Isela Macía Bertrán

**On the Detection of Architecturally-Relevant
Code Anomalies in Software Systems**

Thesis presented to the Programa de Pós-Graduação em
Informática of the Departamento de Informática, PUC-Rio as
partial fulfillment of the requirements for the degree of Doutor
em Informática.

Prof. Arndt von Staa

Advisor

Departamento de Informática – PUC-Rio

Prof. Alessandro Fabricio Garcia

Co-advisor

Departamento de Informática – PUC-Rio

Profa. Yuanfang Cai

Drexel University

Prof. Paulo Henrique Monteiro Borba

Universidade Federal de Pernambuco (UFPE)

Prof. Carlos José Pereira de Lucena

Departamento de Informática – PUC-Rio

Prof. Renato Fontoura de Gusmão Cerqueira

Departamento de Informática – PUC-Rio

Prof. José Eugenio Leal

Coordinator of the Centro Técnico Científico da PUC-Rio

Rio de Janeiro, March 20th, 2013

All rights reserved. It is not allowed the total or partial reproduction of this work without the university, author and supervisor authorization.

Isela Macía Bertrán

She completed her undergraduate studies in Computer Science at the University of Havana, Cuba in 2006. She received her Master degree in Informatics from the Pontifical Catholic University of Rio de Janeiro (PUC-Rio) in 2009.

Bibliographic data

Macía, Isela

On the Detection of Architecturally-Relevant Code Anomalies in Software Systems / Isela Macía Bertrán; orientador: Arndt von Staa; co-orientador: Alessandro Fabricio Garcia. — Rio de Janeiro : PUC-Rio, Departamento de Informática, 2013.

260 f: il.(color.) ; 30 cm

1. Tese (Doutorado em Informática) — Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2013.

Inclui Bibliografia.

1. Informática – Tese. 2. Arquitetura de Software. 3. Degradação Arquitetural. 4. Anomalia de Código. 5. Anomalia de Código Arquiteturalmente Relevante. I. Staa, Arndt. II. Garcia, Alessandro. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. IV. Título.

CDD: 004

To my parents and husband

Acknowledgements

First of all, I would like to thank my supervisors, Arndt von Staa and Alessandro Garcia. In particular, I want to express my deepest appreciation to Arndt for his amazing wisdom and for giving me freedom to shape my research. Arndt, thanks a lot for always keeping me on the right track and making me think. I must also thank Alessandro Garcia for teaching me how to do research, guiding me in all the process with his extensive knowledge, enthusiasm and motivation.

Special thanks to Nenad Medvidovic for his honest criticism, insightful discussions, questions and for helping me during my stay in Los Angeles.

I am also thankful to the members of my examining committee, who generously contributed with their time and expertise: Carlos Lucena, Renato Cerqueira, Yuanfang Cai and Paulo Borba.

Very special thanks to Julio for the unique way in which he loves, understands, and supports me day by day, to my parents and to my family who have believed in me and encouraged me since the first day .

Special thanks to Chico and Camila for the long talks and for supporting me in the bureaucratic tasks, and to Roberta for the hard work together and for reviewing the English in this thesis.

I thank Rosi for supporting me and helping me.

Thanks to my colleagues from the Laboratory of Software Engineering (LES) at PUC-Rio for the wonderful environment, to my colleagues from the Opus Research Group at PUC-Rio for the technical discussions and to my colleagues and staff members from the Department of Computer Science at USC for making my stay there an extraordinary experience.

Thanks to Vera for her constant smile and for lending me her phone booth.

Many thanks to the faculty and staff members from the Informatics Department at PUC-Rio, who helped me during the last six years.

My doctoral studies were financially supported by CNPq and CAPES. These institutions have all my gratitude.

Abstract

Macía, Isela; Staa, Arndt; Garcia, Alessandro. **On the Detection of Architecturally-Relevant Code Anomalies in Software Systems**. Rio de Janeiro, 2013. 260p. DSc. Thesis – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Code anomalies can signal software architecture degradation. However, the identification of architecturally-relevant code anomalies (i.e. code anomalies that strongly imply architectural deficiencies) is particularly challenging due to: (i) lack of understanding about the relationship between code anomalies and architectural degradation, (ii) the focus on source code anomaly detection without considering how it relates to the software architecture, and (iii) lack of knowledge about how reliable these detection techniques are when revealing architecturally-relevant code anomalies. This thesis presents techniques for identifying architecturally-relevant code anomalies. Architecture-sensitive metrics and detection strategies were defined to overcome the limitations of conventional detection strategies. These metrics and strategies leverage traces that can be established between architectural views and system implementation. The thesis also documents code anomaly patterns (i.e. recurring anomaly relationships) that are strongly related to architectural problems. A tool, called SCOOP, was developed to collect the architecture-sensitive metrics, apply the new detection strategies, and identify the documented code anomaly patterns. Using this tool, we evaluated our technique in a series of empirical studies, comparing its accuracy with that of conventional detection techniques when identifying architecturally-relevant code anomalies.

Keywords

Software Architecture; Architectural Degradation; Code Anomaly; Architecturally-Relevant Code Anomaly.

Resumo

Macia, Isela; Staa, Arndt; Garcia, Alessandro. **Detecção de Anomalias de Código Arquiteturalmente Relevantes em Sistemas de Software**. Rio de Janeiro, 2013. 260p. Tese de Doutorado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Anomalias de código podem sinalizar a degradação da arquitetura de software. No entanto, a identificação de anomalias de código arquiteturalmente relevantes (ou seja, aquelas que implicam em deficiências arquiteturais) é particularmente difícil devido: (i) a falta de compreensão sobre a relação existente entre anomalias de código e degradação arquitetural, (ii) ao fato do processo de detecção de anomalias ter como foco somente o código fonte, sem considerar como ele se relaciona com sua arquitetura, e (iii) a falta de conhecimento sobre a confiabilidade das técnicas de detecção em revelar anomalias de código que são arquiteturalmente relevantes. Esta tese apresenta técnicas para identificar anomalias de código que são arquiteturalmente relevantes. Métricas sensíveis à arquitetura e estratégias de detecção foram definidas para superar as limitações das técnicas de detecção convencionais. Estas métricas e estratégias aproveitam rastros que podem ser estabelecidos entre as visões arquiteturais e a implementação dos sistemas. A tese também documenta padrões de anomalias de código (ou seja, relações recorrentes de anomalias) que estão relacionados com problemas arquiteturais. Uma ferramenta, chamada de SCOOP, foi desenvolvida para coletar as métricas sensíveis à arquitetura, aplicar as novas estratégias de detecção, e identificar os padrões de anomalias de código. Usando esta ferramenta, a técnica proposta foi avaliada em uma série de estudos empíricos, comparando sua acurácia com técnicas convencionais de detecção durante o processo de identificação de anomalias de código que são arquiteturalmente relevantes.

Palavras-chave

Arquitetura de Software; Degradação Arquitetural; Anomalia de Código; Anomalia de Código Arquiteturalmente Relevante.

Summary

1 Introduction	17
1.1. Motivation	18
1.1.1. Motivating Example	19
1.2. Problem Statement	22
1.3. The State of Art on Code Anomalies and their Empirical Evaluation	24
1.4. Research Questions	26
1.5. Outline of the Thesis Structure	29
2 Background and Related Work	32
2.1. Software Architecture	32
2.2. Architectural Degradation	34
2.2.1. Causes of Architectural Degradation	35
2.2.2. Prevention of Architectural Violations	36
2.2.3. Architectural Anomalies	38
2.2.4. Analysis of Existing Techniques to Prevent Architectural Degradation	40
2.2.4.1. Lack of Prescribed Design Decisions	40
2.2.4.2. Lack of Mechanisms to Detect Architectural Anomalies	40
2.3. Code Anomalies	41
2.3.1. Detection of Code Anomalies	42
2.3.2. Removal of Code Anomalies by means of Refactorings	45
2.3.3. Empirical Studies about Code Anomaly Side Effects	46
2.3.4. Analysis of Existing Research on Code Anomalies	47
2.3.4.1. Lack of Extensive Catalogs of Aspect-Oriented Code Anomalies	48
2.3.4.2. Lack of Knowledge about the Code Anomaly Influence on Architectural Design	48

2.3.4.3. Lack of Documentation on Code Anomaly Patterns	49
2.4. Summary	50
 3 Code Anomalies in Aspect-Oriented Programming	 52
3.1. Aspect-Oriented Programming	53
3.2. Identification of Code Anomalies	55
3.2.1. A Catalog of Already Documented Code Anomalies	55
3.2.2. A Catalog of New Code Anomalies	57
3.2.2.1. Anomalous Pointcut Definition	57
3.2.2.2. Undesirable Interdependencies	61
3.3. Experimental Evaluation	66
3.3.1. Target Systems	67
3.3.2. Study Phases and Assessment Procedures	68
3.3.3. Findings on Code Anomaly Occurrences	69
3.3.4. Threats to Validity	74
3.4. Summary	76
 4 Impact of Code Anomalies on Architectural Degradation	 78
4.1. Study Definition and Design	79
4.1.1. Hypotheses	81
4.1.2. Variable Selection	81
4.1.3. Selection Criteria and Target Systems	82
4.1.4. Procedures for Data Collection	83
4.2. Findings on the Impact of Code Anomalies	90
4.2.1. Are Anomalous Code Elements Architecturally-Relevant?	90
4.2.2. Are Particular Characteristics of Code Anomalies Indicators of Architectural Degradation Symptoms?	93
4.2.2.1. Type of Code Anomalies	94
4.2.2.2. Earliness of Code Anomalies	96
4.2.3. Are Architecturally-Relevant Code Anomalies often Refactored?	100
4.3. Threats to Validity	103
4.4. Summary	104

5 Analysis of Conventional Detection Strategies	107
5.1. Study Definition and Design	108
5.1.1. Hypotheses	109
5.1.2. Variable Selection	110
5.1.3. Selection Criteria and Target Systems	110
5.1.4. Procedures for Data Collection and Analysis	111
5.2. Study Results	113
5.2.1. Accuracy of Detecting Architectural Violations	115
5.2.2. Accuracy of Detecting Architectural Anomalies	116
5.2.3. Hypotheses and Overall Accuracy Results	116
5.2.4. Analysis of Overlooked Code Anomalies	117
5.2.4.1. Inability to Analyze Properties of Architectural Concerns in the Source Code	118
5.2.4.2. Inability to Identify Architectural Information in the Source Code	119
5.2.4.3. Patterns of Code Anomalies	120
5.2.4.4. Architectural Design and Strategy Accuracy	122
5.3. Threats to Validity	123
5.4. Summary	125
6 Detection of Architecturally-Relevant Code Anomalies with Architectural-Sensitive Information	127
6.1. Basic Formalism	129
6.1.1. System Meta-Model	129
6.1.2. Meta-Model Instantiation	133
6.2. Architecture-Sensitive Metrics	134
6.2.1. Metrics for Architectural Components	135
6.2.2. Metrics for Architectural Concerns	138
6.3. Architecture-Sensitive Detection Strategies	140
6.3.1. Detection Strategies for Element Anomalies	141
6.3.1.1. Feature Envy (FE)	141
6.3.1.2. Misplaced Class (MC)	142
6.3.1.3. Long Method (LC)	143

6.3.1.4. God Class (GC)	144
6.3.2. Detection Strategies for Collaborative Anomalies	145
6.3.2.1. Shotgun Surgery (SS)	145
6.3.2.2. Intensive Coupling (IC)	146
6.3.2.3. Dispersed Coupling	147
6.4. Assessment of Architecture-Sensitive Strategies	149
6.4.1. Hypotheses	150
6.4.2. Variable Selection	151
6.4.3. Selection Criteria and Target Systems	152
6.4.4. Procedures for Data Collection and Analysis	153
6.4.5. Findings on Architecture-Sensitive Detection Strategies	156
6.4.5.1. Accurate Detection of Architecturally-Relevant Code Anomalies	156
6.4.5.2. Impact of Architecture-Sensitive Information is Manifold	159
6.4.5.3. Impact of Concern Granularity Level	164
6.4.6. Imperfections in the Detection of Architecturally-Relevant Anomalies	166
6.4.7. Threats to Validity	168
6.5. Summary	169
 7 Patterns of Code Anomalies	 171
7.1. Defining Code Anomaly Patterns	173
7.2. Intra-Component Patterns	173
7.2.1. Multiple-Anomaly Syndrome	174
7.2.2. Similar Anomalous Neighbors	177
7.3. Inter-Component Patterns	180
7.3.1. External Attractor	180
7.3.2. External Addictor	184
7.3.3. Replicated External Network	187
7.4. Inheritance-based Patterns	190
7.4.1. Hereditary Anomaly	191
7.4.2. Mutant Anomaly	194
7.5. Concern-based Patterns	197

7.5.1. Concern Overload	198
7.5.2. Misplaced Concern	200
7.6. Correlating Code Anomaly Patterns	203
7.7. SCOOP: Detecting Architecturally-Relevant Code Anomalies	205
7.7.1. The SCOOP Inputs	206
7.7.2. The SCOOP Engine	207
7.7.3. The SCOOP User Interface	208
7.8. Assessment Code Anomaly Patterns	211
7.8.1. Hypotheses	212
7.8.2. Variable Selection	212
7.8.3. Selection Criteria and Target Systems	213
7.8.4. Procedures for Data Collection	213
7.8.5. Study Results	214
7.8.5.1. Frequency of Patterns in the Target Systems	214
7.8.5.2. Code Anomaly Patterns and Architectural Degradation	217
7.8.5.3. Specific Code Anomaly Patterns and Architectural Degradation	223
7.9. Threats to Validity	225
7.10. Summary	227
 8 Conclusions and Future Work	 230
8.1. Revisiting the Thesis Contributions	231
8.2. Future Work	235
 References	 239
 Appendix A: System Characteristics	 253
 Appendix B: Detection Strategies and Thresholds	 255
 Appendix C: BNF SCOOP Grammar	 259
 Appendix D: SCOOP Rules File	 260

List of Figures

Figure 1.1: A design slice of the Health Watcher system architecture.	20
Figure 3.1: Example of an aspect in AspectJ.	55
Figure 3.2: Example of <i>God Pointcut</i> .	58
Figure 3.3: Example of <i>Idle Pointcut</i> .	60
Figure 3.4: Example of <i>Redundant Pointcut</i> .	61
Figure 3.5: Example of <i>Forced Join Point</i> .	62
Figure 3.6: Example of <i>God Aspect</i> .	64
Figure 3.7: Example of <i>Composition Bloat</i> .	66
Figure 4.1: Impact of an early code anomaly through the system evolution.	98
Figure 4.2: Refactorings and architecturally-relevant code anomalies.	101
Figure 5.1: Accuracy of detection strategies.	113
Figure 5.2: Example of neglected <i>God Class</i> .	119
Figure 6.1: System meta-model.	130
Figure 6.2: A slice of the logistic system design.	131
Figure 6.3: A slice of the logistic system design.	136
Figure 6.4: Precision rates (%) when components do not match packages.	160
Figure 6.5: Recall rates (%) when components do not match packages.	161
Figure 6.6: Accuracy rates (%) at different granularity levels of concerns.	165
Figure 7.1: Abstract representation of <i>Multiple-Anomaly Syndrome</i> .	175
Figure 7.2: Occurrence of <i>Multiple-Anomaly Syndrome</i> in MobileMedia.	177
Figure 7.3: Abstract representation of <i>Similar Anomalous Neighbors</i> .	178
Figure 7.4: Occurrence of <i>Similar Anomalous Neighbors</i> in PDP	180
Figure 7.5: Abstract representation of <i>External Attractor</i> .	181
Figure 7.6: Occurrence of <i>External Attractor</i> in HW.	184
Figure 7.7: Abstract representation of <i>External Addictor</i> .	185
Figure 7.8: Abstract representation of <i>Replicated External Network</i> .	187
Figure 7.9: Occurrence of <i>External Addictor</i> in HW	188
Figure 7.10: Occurrence of <i>Replicated External Network</i> in HW.	190
Figure 7.11: Abstract representation of <i>Hereditary Anomaly</i> .	192
Figure 7.12: Occurrence of <i>Hereditary Anomaly</i> in PDP.	194

Figure 7.13: Abstract representation of <i>Mutant Anomaly</i> .	194
Figure 7.14: Occurrence of the <i>Mutant Anomaly</i> in MIDAS.	197
Figure 7.15: Abstract representation of <i>Concern Overload</i> .	198
Figure 7.16: Occurrence of <i>Concern Overload</i> pattern in the MIDAS system.	200
Figure 7.17: Abstract representation of <i>Misplaced Concern</i> .	201
Figure 7.18: Occurrence of <i>Misplaced Concern</i> in the MIDAS.	203
Figure 7.19: Relationships between code anomaly patterns.	204
Figure 7.20: Abstract representation of the SCOOP architecture.	205
Figure 7.21: Configuration of code anomaly patterns in SCOOP.	205
Figure 7.22: Code anomalies view.	210
Figure 7.23: Code anomaly patterns view.	210
Figure 7.24: Anomalous code elements taking part in the patterns.	217
Figure 7.25: Precision of patterns vs. detection strategies in the identification of architecturally-relevant code anomalies.	221
Figure 7.26: Recall of patterns vs. detection strategies in the identification of architecturally-relevant code anomalies.	222

List of Tables

Table 1.1: Publications directly related to this thesis.	28
Table 1.2: Indirect publications.	29
Table 2.1. Tools for code anomaly detection and source code analysis	44
Table 3.1: Criteria used for the selection of target systems	67
Table 3.2: Analyzed concerns in target systems	68
Table 3.3: Code anomaly occurrences in iBATIS.	70
Table 3.4: Code anomaly occurrences in Aspectual Watcher.	70
Table 3.5: Code anomaly occurrences in Aspectual Media.	70
Table 3.6: Simultaneous occurrences of code anomalies.	74
Table 4.1: Research questions and hypotheses of the study.	81
Table 4.2: Criteria used for the selection of target systems.	82
Table 4.3: Architectural concerns considered in the study.	85
Table 4.4: Architectural anomalies analyzed in the study.	87
Table 4.5: Code anomalies analyzed in the study.	88
Table 4.6: Fisher's test results for architectural violations.	91
Table 4.7: Fisher's test results for architectural anomalies.	91
Table 4.8: Significant p-values for architectural violations.	95
Table 4.9: Significant p-values for architectural anomalies.	95
Table 4.10: Proportions of effective and non-effective refactorings.	102
Table 5.1: Research questions and hypotheses of the study.	109
Table 5.2: Criteria used for the selection of target systems.	110
Table 5.3: Results for the analyzed detection strategies.	114
Table 6.1: Meta-Model instantiation.	134
Table 6.2: Summary of the suite of architecture-sensitive metrics.	134
Table 6.3: Research questions and hypotheses of the study.	150
Table 6.4: Criteria used for the selection of target systems.	152
Table 6.5: Architectural concerns considered in this study for S1, S2 and S3.	153
Table 6.6: Results for the analyzed architecture-sensitive detection strategies.	157

Table 6.7: Results for the analyzed conventional detection strategies.	157
Table 7.1: Research questions and hypotheses of the study.	212
Table 7.2: Code anomaly patterns per target system.	218
Table 7.3: Contingency table and Fisher's test results.	218
Table 7.4: Mann-Whitney's test results and Cohen's d effort.	223
Table 7.5: Spearman's rank correlation results.	223
Table B-1: Thresholds used in the conventional strategies.	256
Table B-2: Thresholds used for the conventional metrics in the architecture-sensitive strategies.	257
Table B-3: Thresholds used for the architecture-sensitive metrics in the architecture-sensitive strategies.	258