



Universiteit
Leiden
The Netherlands

Interactive scalable condensation of reverse engineered UML class diagrams for software comprehension

Osman, M.H.B.

Citation

Osman, M. H. B. (2015, March 10). *Interactive scalable condensation of reverse engineered UML class diagrams for software comprehension*. Retrieved from <https://hdl.handle.net/1887/32210>

Version: Not Applicable (or Unknown)

License: [Leiden University Non-exclusive license](#)

Downloaded from: <https://hdl.handle.net/1887/32210>

Note: To cite this publication please use the final published version (if applicable).

Cover Page



Universiteit Leiden



The handle <http://hdl.handle.net/1887/32210> holds various files of this Leiden University dissertation.

Author: Osman, Mohd Hafeez Bin

Title: Interactive scalable condensation of reverse engineered UML class diagrams for software comprehension

Issue Date: 2015-03-10

Interactive Scalable Condensation of Reverse Engineered UML Class Diagrams For Software Comprehension

Mohd Hafeez Osman

March 2015

Interactive Scalable Condensation of Reverse Engineered UML Class Diagrams For Software Comprehension

Proefschrift

ter verkrijging van
de graad van Doctor aan de Universiteit Leiden,
op gezag van Rector Magnificus prof.mr. C.J.J.M. Stolker,
volgens besluit van het College voor Promoties
te verdedigen op dinsdag 10 maart 2015
klokke 11:15 uur

door

Mohd Hafeez Osman
geboren te Perak, Maleisië
in 1979

Promotiecommissie

Promotores	:	Prof. dr. J.N. Kok Prof. dr. M.R.V. Chaudron	Universiteit Leiden Chalmers and Gotëborgs Universitet
Copromotor	:	Dr. P. van der Putten	Universiteit Leiden
Commissieleden	:	Prof. dr. T.H. Bäck Prof. dr. Y.-G. Guéhéneuc Prof. dr. J.J. Vinju Dr. Ir. F. Verbeek	Universiteit Leiden École Polytechnique de Montréal Technische Universiteit Eindhoven Universiteit Leiden



This research was financed by the government of Malaysia under the Federal Training Award (HLP).

Images used on the cover :

1. Victory Boogie-Woogie (1942-1944), by *Piet Mondrian*
2. Gray Tree (1911), by *Piet Mondrian*

Copyright ©2015 by Mohd Hafeez Osman
Typeset by L^AT_EX, printed by Ipskamp Drukkers
ISBN: 978-94-6259-588-0

“to my family...”

Contents

I	Introduction and Background	1
1	Introduction	3
1.1	Research Context	3
1.2	Problem Statement	4
1.3	Research Objective	5
1.4	Research Methods	5
1.5	Roadmap	6
2	Definitions	11
2.1	Software Comprehension	11
2.1.1	Program Comprehension Model	12
2.1.2	Cognitive Design Elements for Software Exploration	13
2.2	Forward and Reverse Engineering	14
2.2.1	Forward Engineering	16
2.2.2	Reverse Engineering	17
2.2.3	Static and Dynamic Analysis	17
2.3	The Unified Modeling Language	18
2.3.1	UML Class Diagram	19
2.3.2	UML Class Diagram for Software Comprehension	21
2.3.3	XML Metadata Interchange	21
2.4	Machine Learning	22
2.4.1	Definition of Machine Learning	23
2.4.2	Types of Machine Learning	23
2.4.3	Machine Learning Classification Algorithms	24
2.4.4	Performance Measure For Classification Algorithms	25
2.5	Summary	28

3	UML Usage in Open Source Software Development	31
3.1	Introduction	31
3.2	Related Work	32
3.3	Case Study	33
3.4	Approach	34
3.5	Results and Findings	36
3.5.1	Usage of UML Diagrams	36
3.5.2	Ratio between Design and Implementation	39
3.5.3	Level of Detail (LoD)	39
3.5.4	Frequency of Updating UML Models	42
3.5.5	Key Classes	43
3.5.6	Threats to Validity	45
3.6	Conclusion and Future Work	45
4	Assessing the Correctness and Completeness of UML CASE tools in Reverse Engineering	47
4.1	Introduction	47
4.2	Related Work	49
4.3	Examined Tools and Properties	50
4.3.1	Examined Tools	50
4.3.2	Examined Properties	50
4.4	Sample Cases	52
4.4.1	Movie Catalog System (MovieCat)	52
4.4.2	Automatic Teller Machine (ATM) Simulation System	52
4.5	Approach	52
4.5.1	Round-trip Capability	52
4.5.2	Reconstruction of UML Diagram Types (package/class/sequence)	53
4.6	Result and Findings	54
4.6.1	Reverse Engineering Capability	54
4.6.2	Class Diagram Properties	57
4.7	Discussion	62
4.8	Conclusion and Future Work	64
II	UML Class Diagram Simplification	67
5	Eliciting Developer's Views on Simplifying Class Diagrams	69
5.1	Introduction	70
5.2	Related Work	70
5.2.1	Eye Tracking	70
5.2.2	Software Visualization	71
5.3	Survey Methodology	71

5.3.1	Questionnaire Design	72
5.3.2	Experiment Description	74
5.4	Results and Findings	74
5.4.1	Part A: Personal Background	75
5.4.2	Part B: Selected Cases	80
5.4.3	Part C: Class Diagram Indicators for Class Inclusion/Exclusion	83
5.5	Discussion	87
5.5.1	Class Properties	88
5.5.2	Class Role and Responsibility (RnR)	88
5.5.3	Class Diagram Simplification Tool Features	89
5.5.4	Threats to Validity	89
5.6	Conclusion	89
5.7	Future Work	90
6	Exploring the Suitability of Object-Oriented Design Metrics as Features for Class Diagram Simplification	91
6.1	Introduction	92
6.2	Related Work	93
6.2.1	Usage of design metrics	93
6.2.2	Automated Abstraction of Class Models	93
6.3	Examined Properties and Tools	94
6.3.1	Examined Properties	94
6.3.2	Tools	94
6.4	Survey Methodology	94
6.4.1	Questionnaire Design	94
6.4.2	Experiment Description	98
6.5	Results and Findings	98
6.5.1	Background of the Respondents (Part A)	98
6.5.2	Indicator for Class Inclusion	100
6.5.3	Practical Simplification Problems (Part C)	104
6.6	Discussion	112
6.6.1	Respondents' Background	113
6.6.2	Software Design Metrics	113
6.6.3	Class Names and Coupling	115
6.6.4	Class Diagram Preferences	115
6.6.5	Threats to Validity	115
6.7	Conclusion	116
6.8	Future Work	116

7	Condensing Reverse Engineered Class Diagram using Object-Oriented Design Metrics	117
7.1	Introduction	117
7.2	Related Work	119
7.3	Research Questions	120
7.4	Approach	121
7.4.1	Examined Predictors and Tools	121
7.4.2	Case Studies	122
7.4.3	Process	123
7.5	Evaluation of Results	127
7.5.1	Predictor Evaluation	127
7.5.2	Benchmark Scoring Results	129
7.6	Discussion	131
7.6.1	Threats to Validity	131
7.7	Conclusion and Future Work	132
7.7.1	Future Work	133
8	Condensing Reverse Engineered Class Diagrams through Class Name Based Abstraction	135
8.1	Introduction	135
8.2	Related Work	136
8.2.1	Code Summarization	136
8.2.2	Analysis of Execution Trace	137
8.3	Research Questions	138
8.4	Approach	138
8.4.1	System Document	139
8.4.2	Document Preprocessing	139
8.4.3	Text Processing	141
8.4.4	Text Classification	146
8.4.5	Analyze Result	146
8.5	Experiment Description	146
8.5.1	Dataset	146
8.5.2	Evaluation Measures	146
8.5.3	Experiment	147
8.6	Analysis of Results	147
8.6.1	RQ1 : Influence of Predictors	147
8.6.2	RQ2 : Most Influential Predictors	148
8.6.3	RQ3 : Classification Algorithms Performance	148
8.6.4	RQ4 : Set of Predictors Performance	148
8.7	Discussion	152
8.7.1	Text Metrics Predictors Performance	152
8.7.2	Classification Algorithms	152

8.7.3	Application of Classification Method	152
8.7.4	Threats to Validity	154
8.8	Conclusion and Future Work	154
9	Interactive Scalable Abstraction of Reverse Engineered UML Class Diagrams	155
9.1	Introduction	156
9.2	Related Work	157
9.2.1	The Usage of Network Metrics	157
9.2.2	The Usage of Software Version History	157
9.2.3	Other Related Work	158
9.3	SAAbs Overview	158
9.3.1	Input: XMI	159
9.3.2	Process: XMI Parser	159
9.3.3	Process: Feature Extraction	160
9.3.4	Process: Classification	163
9.3.5	Output: Class Ranking	163
9.3.6	Output: Visualization	164
9.3.7	Implementation	165
9.4	Discussion	168
9.4.1	E3: Provide abstraction mechanism	168
9.4.2	E4: Support goal-directed, hypothesis-driven comprehension . .	168
9.4.3	E5: Provide overviews of the system architecture at various levels of abstraction	168
9.4.4	E6: Support the construction of multiple mental models & E11: Show the path that led to the current focus	168
9.4.5	E15: Provide effective presentation style	169
9.5	Conclusion and Future Work	169
III	Validation and Conclusion	171
10	Validation	173
10.1	Introduction	173
10.2	Research Question	174
10.3	Experiment Design	174
10.3.1	Questionnaire Design	174
10.3.2	Experiment Description	176
10.4	Results	177
10.4.1	RQ1: The Understandability of Condensed Class Diagrams . . .	178
10.4.2	RQ2: Choices of Class Diagram	181
10.4.3	RQ3: Software Architecture Abtractor Framework	182
10.4.4	RQ4: Usefulness of the SAAbs Tool	182

10.5 Discussion	186
10.5.1 Choosing a Class Diagram	186
10.5.2 Limitation of SAAbs	187
10.5.3 Threats to Validity	187
10.6 Conclusion and Future Work	188
11 Conclusions	189
11.1 Summary of Findings	189
11.1.1 RQ1: Which information in class diagrams do developers find important for understanding software designs?	190
11.1.2 RQ2: Which object-oriented design metrics do developers find most indicative for class importance?	191
11.1.3 RQ3: How to automatically condense class diagrams using object-oriented design metrics?	191
11.1.4 RQ4: Can the automatic condensation of class diagrams be enhanced by using class names?	192
11.1.5 RQ5: Does our automated framework for condensing of class diagrams help developers to understand the design of software systems?	193
11.2 Contributions	194
11.3 Discussion	194
11.3.1 Software Comprehension	195
11.3.2 Condensation of Class Diagrams	195
11.4 Future Work	197
11.4.1 Enriching the Ground Truth	197
11.4.2 Exploring Features	198
11.4.3 Task-oriented Validation	198
11.4.4 Class Segmentation	199
11.4.5 Visualization of Result	199
A Case Study Candidates	201
List of Figures	205
List of Tables	209
Bibliography	211
Samenvatting	227
List of Publications	229
Acknowledgments	231
About the Author	233