

Contents

Part I Opening Chapters

1	Introduction	3
1.1	Problem Identification	6
1.2	Outline of the Approach	8
1.3	Research Questions	10
1.4	Research Methodology	17
1.5	Outline	21
2	Running Example: iTrust	25
2.1	Development of a Medical Management System	26
2.2	The iTrust Electronics Health Records System	27
2.3	Suitability of iTrust Concerning the Research Questions	34
3	State of the Art in Secure Software Systems Development	37
3.1	Object-Oriented Programming	37
3.2	Restructuring and Adaption	39
3.3	Model-driven Software Development	39
3.3.1	Domain Model	41
3.3.2	Design Model	41
3.3.3	Implementation Model	42
3.4	Development Processes	44
3.4.1	Sequential Software Development	44
3.4.2	Agile Software Development	46
3.5	(Security-)Compliance & Certifications	47
3.5.1	Architecture Compliance Checking	47
3.5.2	Software Reviews and Audits	48
3.5.3	Standards and Certifications	49

3.6	Security Checks	53
3.6.1	UMLsec Security Checks	53
3.6.2	SecDFD Security Checks	58
3.6.3	Implementation-Level Security Checks	61
3.7	Conclusion on the State of the Art	63
4	A Walkthrough of the Proposed Development Approach	65
4.1	Key Ideas of the GRaViTY Approach	66
4.2	The GRaViTY Development Approach	68
4.3	Developer Perspective on Using GRaViTY	70
 Part II Tracing		
5	Program Model for Object-oriented Languages	75
5.1	Background on Program Representations	77
5.2	Program Model for Object-oriented Programs	78
5.2.1	Namespaces	79
5.2.2	Types	81
5.2.3	Inheritance	82
5.2.4	Methods & Fields	82
5.2.5	Member Access	84
5.2.6	Overloading, Overwriting and Hiding	85
5.2.7	Modifiers & Visibilities	86
5.2.8	Annotation Mechanism	86
5.3	Tool Support	87
5.4	Evaluation of the Program Model	88
5.5	Threats to Validity	89
5.6	Conclusion on the proposed Program Representation	90
6	Model-Synchronization and Tracing	91
6.1	Background on Tracing	93
6.2	Inter-Artifact Tracing and Model-Synchronization	95
6.2.1	Background on Bidirectional Graph Transformations	97
6.2.2	Model-Synchronization with Triple Graph Grammars	98
6.2.3	Tool Support for the Model Synchronization	103
6.2.4	Evaluation of the Model Synchronization	106

6.2.5	Threats to Validity	113
6.2.6	Conclusion on the Inter Artifact Model-Synchronization	114
6.3	Tracing within UML Models of Different Abstraction	115
6.3.1	Background on Refinements in UML Models	115
6.3.2	Refinement Relationship Types	116
6.3.3	Polymorphism in UML	117
6.3.4	UMLsec Secure Dependency in the Context of Inheritance	117
6.3.5	Refinements of UML Models	119
6.3.6	Tool Support for Model Refinements	124
6.3.7	Conclusion on Tracing within UML Models	126
6.4	Tracing and Propagation of Security Requirements	126
6.4.1	Persistence of Security Requirements in the Implementation	127
6.4.2	Dynamic Tracing between UML Models and the Implementation	134
6.4.3	Conclusion on the Propagation of Security Requirements	137
7	Application to Legacy Projects using Reverse-Engineering	139
7.1	Reverse-Engineering UML Models Using TGGs	141
7.2	Mapping Early Design-Models to Code	143
7.2.1	Background on Early Design Models	144
7.2.2	Semi-Automated Mapping Approach	145
7.2.3	Tool Support for Semi-Automated Mappings	154
7.2.4	Evaluation	155
7.2.5	Threats to Validity	160
7.2.6	Conclusion on the Semi-Automated Mappings	161
7.3	Conclusion on the Application to Legacy Projects	162
 Part III Security		
8	Static Security Compliance Checks	165
8.1	Background on Static Security Analysis	168
8.1.1	Design Model-based Security Checks	168
8.1.2	Static Code Analysis	169
8.2	Structural Compliance between Models and Code	170

8.2.1	Automation of Structural Compliance Checks	171
8.2.2	Tool Support for Structural Compliance Checks	173
8.2.3	Conclusion on the Structural Compliance Checks ...	173
8.3	Leveraging Correspondence Models for the Calculation of Security Metrics	174
8.3.1	Background on Security Metrics	175
8.3.2	Leveraging Traces for Security Metric Calculation	177
8.3.3	Tool Support for the Calculation of Security Metrics	182
8.3.4	Conclusion on Security Metrics	184
8.4	Security Compliance Checks between Models & Code	186
8.4.1	Verification of SecDFD Contracts	186
8.4.2	Tool Support for the Verification of Contract Implementations	191
8.4.3	Evaluation of the Contract Verification	193
8.4.4	Threats to Validity	197
8.4.5	Conclusion on the SecDFD Contract Verification	198
8.5	Optimized Data Flow Analysis	199
8.5.1	Optimizing Data Flow Analysis based on Security Requirements	199
8.5.2	Tool Support for Optimized Data Flow Analysis	201
8.5.3	Evaluation of the Optimized Data Flow Analysis	202
8.5.4	Threats to Validity	206
8.5.5	Conclusion on the Optimized Data Flow Analysis	207
8.6	Specification of Incremental Security Checks	208
8.6.1	Background on Henshin Model Transformations	208
8.6.2	Incremental Security Violation Patterns	208
8.6.3	Tool Support for Security Violation Patterns	211
8.6.4	Evaluation of Incremental Security Violation Patterns	212
8.6.5	Threats to Validity	218
8.6.6	Conclusion on Security Violation Patterns	219
9	Verification and Enforcement of Security at Run-time	221
9.1	Background on Security Compliance at Run-time	224
9.2	Example Security Violation	225
9.3	Verification at Run-time and Model Adoption	228

9.3.1	Security Monitoring at Run-time	229
9.3.2	Countermeasures	232
9.3.3	Automated Software System Evolution	234
9.4	Tool Support for Monitoring and Adaption	241
9.4.1	Java Annotations and IDE Support	242
9.4.2	Validation at Run-time and Countermeasures	242
9.4.3	Automated Adaption of Design-Time Models	243
9.5	Evaluation of the Security Monitor	243
9.5.1	O1–Effectiveness of the Run-time Monitoring	244
9.5.2	O2–Applicability of the Run-time Monitoring	248
9.5.3	O3–Usability	250
9.6	Threats to Validity	253
9.6.1	Internal Validity	253
9.6.2	External Validity	253
9.7	Conclusion on the Run-time Security Monitoring	254

Part IV Maintenance

10	Security-aware Refactoring of Software Systems	259
10.1	Background on Object-Oriented Refactorings	261
10.2	Formalization of Object-Oriented Refactorings	262
10.2.1	Refactoring of Java Programs	263
10.2.2	Program Refactoring based on Graph Transformation	266
10.2.3	Co-Evolution due to Refactoring Application	287
10.2.4	Tool Support for the Application of Formalized Refactorings	290
10.2.5	Evaluation of the Refactoring Technique	291
10.2.6	Threats to Validity	293
10.2.7	Conclusion on Formalizing Refactorings	294
10.3	Security-aware Refactorings	294
10.3.1	Controlling the Attack Surface of Object-Oriented Refactorings	295
10.3.2	Security Preserving Refactorings	296
10.3.3	Conclusion on the Security Preserving Refactorings	300
10.4	Conclusion on the Refactoring of Security-Critical Software Systems	300

Part V Variants

11 Specification of Variability throughout Variant-rich Software Systems	305
11.1 Background on Variability Engineering	307
11.1.1 Feature Identification and Specification	308
11.1.2 Implementation of Variability	310
11.1.3 Product Deployment	311
11.2 UML and PM Variability Extension	313
11.2.1 Variability Notations in GRaViTY	313
11.2.2 Parsing of Antenna Annotations and Mapping to Models	318
11.3 Tool Support for the Synchronization of Variability Annotations	320
11.4 Evaluation of the Variability Extension	321
11.5 Threats to Validity	322
11.5.1 Construct Validity	322
11.5.2 Internal Validity	322
11.5.3 External Validity	322
11.6 Conclusion on GRaViTY's Variability Extension	323
12 Security in UML Product Lines	325
12.1 Security and Variability Profile	328
12.1.1 «ConditionalCritical»	329
12.1.2 «ConditionalSecrecy», «ConditionalIntegrity», etc.	329
12.1.3 «ConditionalEncrypted», «ConditionalLAN», etc.	331
12.2 Deriving Products	332
12.3 Family-based Security Analysis	334
12.3.1 UMLsec Checks as OCL Constraints	335
12.3.2 Template Interpretation	338
12.3.3 Discussion of Correctness and Performance	339
12.3.4 Extensibility of the Approach	340
12.4 Tool Support for Family-based Security Checks of UML Product Lines	341
12.5 Evaluation of SecPL	344
12.5.1 O1—Efficiency of the Security Checks	344
12.5.2 O3—Usefulness of the Tool Support and Security Checks	348

12.6	Threats to Validity	351
12.6.1	External Validity	351
12.6.2	Internal Validity	351
12.6.3	Conclusion Validity	352
12.6.4	Construct Validity	352
12.7	Conclusion on Security in UML Product Lines	352
13	Security Compliance and Restructuring in Variant-rich Software Systems	355
13.1	Application Scenario	359
13.1.1	iTrust example SPL	359
13.1.2	Rule Variants	360
13.1.3	Variability-based Model Transformation	362
13.2	Multi-Variant Model Transformation	364
13.2.1	Solution Overview	364
13.2.2	Multi-Variant Transformation Algorithm	367
13.3	Tool Support for Multi-Variant Model Transformation	371
13.4	Evaluation of the Multi-Variant Model Transformation	371
13.4.1	Detection of Edit Operations	372
13.4.2	Move Method Refactorings	374
13.5	Threats to Validity	377
13.5.1	External Validity	377
13.5.2	Construct Validity	377
13.6	Conclusion on Multi-Variant Model Transformation	377
 Part VI Tool Support and Application		
14	The GRaViTY Framework	383
14.1	Structuring into Eclipse Plugins	383
14.2	GRaViTY as Software Product Line	389
14.3	Conclusion on the Implementation of GRaViTY	391
15	Case Studies	393
15.1	Case Study 1: iTrust	394
15.1.1	Description of the Case Study Execution	394
15.1.2	Discussion of the Observations	401
15.2	Case Study 2: Eclipse Secure Storage	402
15.2.1	Discussion of the Case Study Execution	402
15.2.2	Discussion of the Observations	411
15.3	Threats to Validity	412
15.4	Conclusion on the Case Studies	412

Part VII Closing Chapters

16 Related Work	417
16.1 Tracing between Models and Code	417
16.2 Security Compliance of Models and Code	419
16.2.1 Model-Based Security Analysis	420
16.2.2 Security Compliance	421
16.2.3 Run-time Security Monitoring	422
16.3 (Security-aware) Refactorings	423
16.4 Software Product Lines	425
16.4.1 Product Line Transformations	425
16.4.2 Security of Software Product Lines	426
17 Conclusion	429
17.1 Research Outcomes	431
17.2 Assumptions and Limitations	433
17.2.1 Required Artifacts	434
17.2.2 Tracing and Synchronization	434
17.2.3 Security Requirements and Checks	435
17.2.4 Security Preservation and Re-Certification	436
17.2.5 Software Product Lines	436
17.3 Outlook	437
17.3.1 Automated Trace Creation	437
17.3.2 Continuous Integration	438
17.3.3 Multi-Language Software Systems	438
17.3.4 Security Requirements and Checks	438
17.3.5 Customization	439
17.3.6 Expressiveness of Languages	439
17.3.7 Distributed System Analysis	440
17.3.8 Code Generation	440
17.3.9 Software Product Lines	440
17.4 Summary	441
Bibliography	443