

## ABSTRACT

RIAZ, MARIA. Inferring Security Requirements from Natural Language Requirements Artifacts. (Under the direction of Dr. Laurie Williams).

Security requirements engineering can provide a foundation for secure software development. However, security requirements are often inadequately understood and improperly specified, mostly due to lack of security expertise and a lack of emphasis on security during early stages of system development. Natural language requirements artifacts, such as requirements specifications, explicitly state the functional requirements for software systems. However, these functional requirements often have security implications, necessitating additional security requirements that developers may overlook. These implied security requirements should be identified and specified to strengthen the overall security of the system.

The goal of this research is *to support secure software development by systematically identifying and specifying implied security requirements of a system using empirically-developed security requirements patterns.*

Our process takes as input existing natural language requirements artifacts and produces a set of candidate security requirements for the system as output. From a user's perspective, an analyst can import the artifacts they want to analyze into our process. Our process will automatically identify security properties implied by sentences in the input and suggest applicable security requirements patterns. The analyst will then be able to select and instantiate appropriate patterns to specify security requirements for the system.

To enable the automatic identification of security properties, we manually developed a labeled corpus of requirements sentences and the security properties implied by those sentences (such as confidentiality, integrity, availability, accountability, and privacy) using a

classification guide. Once the domain corpus is available, it can be used to train machine learning classifiers and predict security properties across other documents in the domain. The labeled corpus can be developed incrementally over time as additional labeled requirement sentences are added into the system. We evaluated the performance of the machine learning classifiers in identifying security properties through a case study of six documents from the electronic healthcare domain. The domain corpus contains ~11,000 sentences, of which 13% explicitly state security requirements and an additional 33% sentences indicate the need for implied security requirements. Our classification approach identified security properties with a precision of .82 and recall of .79. We evaluated the generalizability of our process by analyzing over 350 randomly selected requirements sentences from a networking product requirements specification document. We predicted the security properties with a maximum precision of .77 and maximum recall of .75.

To support the specification of security requirements based on the implied security properties, we have empirically developed a set of 35 security requirements patterns based upon an analysis of NIST security and privacy controls. Each pattern is cataloged in terms of one or more security properties and supports the goals for preventing, detecting or responding to a breach of the corresponding security properties. The solution part of the pattern provides a group of related security requirements templates, with a combined set of 131 different templates in the 35 patterns. The templates can be instantiated to specify security requirements to support the goals related to each security property.

We conducted multiple controlled experiments involving a total of 205 graduate students to evaluate the use of automatically-suggested security requirements templates in identifying and specifying security requirements. Participants in the treatment group, using

the templates, identified almost twice as many relevant security requirements (46% vs. 25%) in the same amount of time as compared to the control group. The requirements identified by the treatment group were also more relevant (89% vs. 62%) and of better quality (3.2 vs. 2.3 on a scale of 1-5). Our results indicate that our process can support an analyst in considering multiple security properties and identifying a larger set of security requirements when compared to a control group.

We conducted an evaluation of our process in identifying security requirements as compared to a proprietary process for a product from the networks domain. Using our process, we identified security requirements comparable to 105 of the 108 security requirements identified using the proprietary process. We missed 2 non-technical security requirements related to personal training, and partially identified requirements related to the use of a specific operating system. We also proposed 6 additional security requirements patterns for detecting and responding to vulnerabilities and failure conditions, and limiting system's exposure to persistent attackers.

We contribute an empirically validated systematic process for identifying a candidate set of implied security requirements from natural language requirements artifacts. To support automatic identification of security properties, we create domain classifiers for healthcare and networks domains. To support specification of security requirements related to the identified properties, we develop a framework for systematically discovering security goals and empirically developing security requirements patterns to meet the goals. Our research can help mitigate errors of omissions in security requirements by identifying the implied security requirements early on that may otherwise be overlooked.

© Copyright 2016 Maria Riaz

All Rights Reserved

Inferring Security Requirements from Natural Language Requirements Artifacts

by  
Maria Riaz

A dissertation submitted to the Graduate Faculty of  
North Carolina State University  
in partial fulfillment of the  
requirements for the degree of  
Doctor of Philosophy

Computer Science

Raleigh, North Carolina

2016

APPROVED BY:

---

Dr. Laurie Williams  
Committee Chair

---

Dr. Julie Earp

---

Dr. Emerson Murphy-Hill

---

Dr. Munindar Singh

---

Dr. Mladen Vouk

## **DEDICATION**

To my children, the love, peace and joy of my life.

To my husband, for his commitment and support in his own unique way.

To my parents, for being my voice of reason, for their unconditional love.

To my siblings, for being my best friends.

To all my mentors among my professors, colleagues and friends for enriching my life.

## **BIOGRAPHY**

Maria Riaz was born and raised in Pakistan in a closed-knit family of six. She completed her Bachelors of Engineering in Software Engineering from National University of Sciences and Technology (NUST) in 2003 with highest distinction and gold medals for best academic performance and best project. She received scholarship to peruse Masters in Computer Engineering from Kyung Hee University, South Korea, which she completed in 2005. Maria joined G. I. K. Institute of Engineering Sciences and Technology (GIKI), Pakistan, as a Research Associate in 2006. She taught undergraduate courses, supervised senior design projects and served as the advisor for Women Engineering Society among other activities during her four years at GIKI. With a strong inclination towards teaching and research, Maria came to US to pursue PhD at North Carolina State University (NCSU) in 2011. Maria received Faculty for the Future fellowship given her academic record, commitment to teaching and supporting women in engineering. Maria has lived in four different countries, gaining invaluable experiences and a deep appreciation for diversity and acceptance. Maria is set to join Google Inc. at the completion of her PhD.

## ACKNOWLEDGMENTS

I owe my sincerest gratitude to all who contributed directly and indirectly towards completion of this dissertation.

I am truly thankful to my adviser Dr. Laurie Williams for her graciousness and guidance. She has been my mentor, my advocate and a constant source of inspiration. She always has my best interest at heart, whether it's advising my research, providing financial assistantship, helping me carve a career path, or supporting me in balancing work and family life. She is a wonderful role model.

I am thankful to my committee members for their time and genuine interest in helping me succeed: Dr. Julie Earp, Dr. Emerson Murphy-Hill, Dr. Munindar Singh, and Dr. Mladen Vouk. Dr. Singh has been especially pivotal during my PhD. I have constantly solicited his wisdom and expertise and benefited from his insights.

I am fortunate to have wonderful colleagues and friends in my research group: Sarah Elder, Pat Francis, Eric Helms, JeeHyun Hwang, DaYoung Lee, Jason King, Pat Morrison, Rahul Pandita, Akond Rahman, John Slankas, Ben Smith, Will Snipes and Chris Theisen among others. In addition to providing feedback on my research, they went out of the way to provide me logistic support while working remotely during my research and always made me feel included. A special thanks to John and Jason for their constant collaboration, Sarah for helping with the final analysis, Chris and Rahul for logistic support, Pat Morrison and his wife Karen for their invaluable friendship and career counseling. I am thankful for two of my closest friends at NCSU, Swathi Subramanian and Jitendra Harlalka, who were my teammates for many course projects and freely shared their friendship and positive vibe. I am

thankful to Savera Tanwir for being a gracious host and for helping me navigate through the PhD.

I am truly thankful to David Wright and the Science of Security Lablet community. I am grateful to have closely interacted with Dr. Annie Anton in the first year of my PhD and have benefitted from her guidance, expertise and generosity. I would also like to thank Aaron Massey, Jeremy Maxwell and Jonathan Stallings for sharing their expertise and insights.

It has been my pleasure and a wonderful learning experience to work with Dr. Travis Breaux, Hanan Hibshi and Jennifer Cowley at Carnegie Mellon University. I am also thankful for the opportunity to collaborate with Dr. Jianwei Niu, Jean-Michel Leher and Rocky Slavin at UTSA, Dr. Daphne Yao and her students at Virginia Tech, Dr. Fabio Massacci at UT in Italy, and Christian Quesada and Dr. Marcelo Jenkins at UCR in Costa Rica, who all helped in advancing my research. A special thanks to Tony, Mike, Karen and Steve at Cisco Systems, Inc.

I am truly grateful to all the teachers, advisors and mentors who have helped me learn and grow in both personal and professional capacity. I am thankful to my family and in-laws for their never-ending support and to my friends for cheering me on, for celebrating my milestones, big and small. I am especially grateful to my children who always make me feel like I am making a difference and that everything will be fine.

This work was supported by the USA National Security Agency (NSA) Science of Security Lablet and partially supported by Laboratory of Analytic Sciences (LAS) at NCSU and Schlumberger Faculty For The Future (FFTF) fellowship. Any opinions expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSA.

## TABLE OF CONTENTS

LIST OF TABLES .....	xiii
LIST OF FIGURES .....	xv
LIST OF ABBREVIATIONS.....	xvi
<b>1 INTRODUCTION</b> .....	1
1.1 Solution Methodology.....	4
1.2 Research Questions .....	9
1.3 Thesis Contributions .....	10
1.4 Dissertation Structure.....	11
<b>2 BACKGROUND</b> .....	13
2.1 Software Requirements .....	13
2.1.1 Software Requirements Models.....	13
2.1.2 Inferring Specifications from Natural Language Text.....	14
2.2 Software Patterns.....	15
2.2.1 Related Concepts .....	16
2.2.2 Adoption of Patterns in Software Engineering .....	16
2.2.3 Empirical Evaluation of Software Patterns.....	17
2.3 Automated Text Classification.....	19
2.3.1 Supervised Machine Learning for Text Classification .....	19
2.3.2 Metrics for Evaluating Text Classification.....	21
2.4 Empirical Software Engineering .....	23
2.4.1 Empirical Studies .....	23
2.4.2 Replication Studies .....	24
2.4.3 Systematic Reviews .....	25
2.4.4 Validity of Empirical Studies.....	26
<b>3 RELATED WORK</b> .....	28
3.1 Security Requirements Engineering.....	28
3.2 Knowledge Reuse in Security Requirements .....	30
3.3 Security Requirements Patterns .....	33
3.4 Empirical Evaluation of Security Requirements Engineering Methodologies .....	34
<b>4 SYSTEMATIC MAPPING STUDY TO UNDERSTAND EMPIRICAL EVALUATION OF SOFTWARE PATTERNS</b> .....	37

4.1	Methodology .....	39
4.1.1	Exclusion Criteria .....	39
4.1.2	Inclusion Criteria .....	39
4.1.3	Search and Selection Process.....	40
4.1.4	Included Papers.....	44
4.1.5	Quality Assessment of Included Studies.....	46
4.1.6	Data Extraction and Analysis Methodology.....	47
4.2	Results and Analysis .....	49
4.2.1	MS-RQ1: Research Questions and Hypotheses.....	49
4.2.2	MS-RQ2: Empirical Design Context Factors .....	52
4.2.3	MS-RQ3: Constructs and Measures.....	58
4.2.4	MS-RQ4: Threats to Validity of Included Studies .....	60
4.2.5	MS-RQ5: Replicated Studies.....	63
4.3	Synthesis and Recommendations.....	66
4.3.1	Identification of Suitable Participants.....	66
4.3.2	Tasks and Training on Task.....	68
4.3.3	Study Themes and Evaluation Constructs .....	69
4.4	Threats to Validity of Our Mapping Study .....	72
4.5	Discussion .....	73
<b>5</b>	<b>DIGS – A FRAMEWORK FOR DISCOVERING GOALS FOR SECURITY REQUIREMENTS</b>	
	<b>ENGINEERING .....</b>	<b>77</b>
5.1	Elements of DIGS Framework.....	78
5.1.1	Assets .....	78
5.1.2	Actions .....	79
5.1.3	Security Properties .....	80
5.1.4	Security Actions.....	82
5.1.5	Security Mechanisms .....	83
5.2	Security Goal Patterns.....	84
5.3	Implied Security Goal Patterns .....	88
5.4	Steps for Applying DIGS .....	90
5.5	Empirical Evaluation Methodology .....	91
5.5.1	Goals, Hypotheses, and Metrics.....	92

5.5.2	Participants.....	94
5.5.3	Experimental Design.....	95
5.5.4	Experimental Analysis Methodology .....	97
5.5.5	Oracle of Security Goals.....	98
5.6	Results and Analysis .....	99
5.6.1	H <sub>01</sub> : Security Goal Patterns.....	99
5.6.2	H <sub>02</sub> : Implied Security Goals.....	103
5.6.3	Combined Security Goals by Group.....	104
5.6.4	Breakdown of Identified Goal Patterns.....	106
5.6.5	Threats to Validity .....	108
5.7	Discussion and Lessons Learned .....	109
5.8	Summary .....	111
<b>6</b>	<b>PREVENTION, DETECTION AND RESPONSE PATTERNS FOR SECURITY REQUIREMENTS ENGINEERING.....</b>	<b>113</b>
6.1	Understanding the Process for Developing Patterns .....	113
6.1.1	Knowledge Sources .....	114
6.1.2	Knowledge Synthesis and Pattern Identification .....	115
6.1.3	Pattern Representation .....	117
6.1.4	Pattern Refinement.....	117
6.1.5	Pattern Application .....	118
6.2	Considerations for Writing Security Requirements Patterns .....	119
6.2.1	Sources of Security Requirements .....	119
6.2.2	Identifying Reusable Security Requirements.....	121
6.2.3	Decoupling Requirements from Mechanisms.....	121
6.2.4	Managing Conflicts and Ambiguities .....	122
6.2.5	Maintaining Dependencies and Traceability .....	123
6.3	Systematic Process for Developing Security Requirements Patterns .....	123
6.3.1	Identifying and Analyzing the Knowledge Source.....	124
6.3.2	Synthesizing Knowledge for Pattern Identification.....	129
6.3.3	Documenting Security Requirements Patterns .....	132
6.4	Discussion of Identified Security Requirements Patterns .....	134
6.4.1	Supporting Security Goals through Security Requirements Patterns .....	136

6.4.2	Limitations .....	138
6.5	Summary .....	139
<b>7</b>	<b>IDENTIFICATION OF SECURITY PROPERTIES IMPLIED BY SENTENCES IN NATURAL LANGUAGE ARTIFACTS .....</b>	<b>141</b>
7.1	Methodology .....	142
7.1.1	Step 1: Pre-process Input Artifacts .....	142
7.1.2	Step 2: Develop a Domain Corpus.....	143
7.1.3	Step 3: Predict Security Properties .....	146
7.2	A Case Study in Healthcare Domain.....	147
7.2.1	Study Documents .....	147
7.2.2	Domain Corpus .....	149
7.2.3	Breakdown of Security Properties .....	150
7.2.4	Similarities Among Sentences .....	153
7.2.5	Effectiveness of Automatically Identifying Security Properties in Natural Language Text .....	155
7.2.6	Factors Influencing Classifier Performance.....	157
7.3	Generalizability of Approach to Networks Domain .....	161
7.3.1	Domain Corpus .....	161
7.3.2	Breakdown of Security Properties .....	162
7.3.3	Similarities Among Sentences .....	164
7.3.4	Cross-Validation to Analyze Performance of Classifier.....	164
7.3.5	Using Classifier from Healthcare to Predict Security Properties for Networks Domain	168
7.4	Threats to Validity.....	171
7.5	Summary .....	172
<b>8</b>	<b>IDENTIFYING THE IMPLIED SECURITY REQUIREMENTS – CONTROLLED EXPERIMENT AND REPLICATIONS ON THE USE OF SECURITY REQUIREMENTS TEMPLATES .....</b>	<b>174</b>
8.1	Security Requirements Templates used in the Evaluation.....	176
8.2	Research Methodology.....	179
8.2.1	Goals, Hypotheses and Metrics.....	180
8.2.2	Participants.....	182
8.2.3	Study Environment .....	186
8.2.3.1	Shared Aspects of Study Context.....	186

8.2.3.2	Differences in Study Context .....	187
8.2.4	Experiment Artifacts .....	189
8.2.5	Experiment Design.....	192
8.2.6	Evaluation Methodology.....	194
8.2.6.1	Oracle of Security Requirements .....	195
8.2.6.2	Mapping Responses to the Oracle.....	196
8.3	Results Based on Individual Experiments.....	199
8.3.1	NCSU13: Original Experiment at NCSU .....	200
8.3.2	UT14: Replication at UT .....	204
8.3.3	NCSU14: Replication at NCSU.....	207
8.3.4	UCR15: Replication at UCR.....	210
8.3.5	Summary of Findings from Individual Studies.....	212
8.4	Results Based on Analysis across Studies.....	217
8.5	Synthesis based on Differences Introduced among Studies.....	220
8.5.1	Filling the Details in Security Requirements Templates .....	221
8.5.2	Differentiating between Relevant and Extraneous Templates.....	223
8.5.3	Impact of Task Time on Study Outcomes .....	225
8.6	Breakdown of Identified Security Requirements .....	229
8.7	Feedback from Participants.....	233
8.8	Lessons Learned Conducting the Experiments .....	237
8.8.1	Encouraging Participation and Performance .....	237
8.8.2	Communicating with the Original Experimenters .....	238
8.8.3	Minimizing Technical Setup.....	239
8.8.4	Managing and Reporting Emerging Contexts.....	240
8.8.5	Developing Shared Insights .....	240
8.8.6	Working with Diverse Groups of Participants.....	241
8.9	Threats to Validity.....	242
8.9.1	Internal Validity .....	242
8.9.2	External Validity.....	244
8.9.3	Construct Validity.....	245
8.9.4	Conclusion Validity .....	245
8.10	Discussion and Considerations for Future Evaluation .....	248

8.10.1	Differences in Use of Knowledge Sources based on Expertise .....	249
8.10.2	Role of Additional Context and Domain Knowledge.....	250
<b>9</b>	<b>INDUSTRIAL CASE STUDY IN THE NETWORKS DOMAIN.....</b>	<b>252</b>
9.1	Requirements Artifacts.....	252
9.2	Coverage of Security Requirements Patterns in PSB.....	253
9.3	Gap Analysis .....	257
9.3.1	Security Properties Identification .....	257
9.3.2	Suggested Security Requirements Patterns.....	258
9.3.3	Limitations .....	259
9.4	Discussion and Lessons Learned .....	260
<b>10</b>	<b>CONCLUSION .....</b>	<b>261</b>
10.1	Summary.....	261
10.2	Contributions .....	264
10.3	Future Directions .....	265
	REFERENCES .....	267
	APPENDICES .....	280
	APPENDIX A: List of Selected Studies on Software Patterns.....	281
	APPENDIX B: Quality Appraisal of Selected Studies.....	283
	APPENDIX C: Research Questions and Hypotheses of Selected Studies .....	285
	APPENDIX D: Prevention Patterns for Security Requirements Engineering.....	288
	P-C_I_PR-1.....	288
	P-C_I_PR-2.....	288
	P-C_I_PR-3.....	288
	P-C_PR-1 .....	289
	P-C_PR-2 .....	289
	P-AY-1 .....	290
	P-AY-2.....	290
	P-AY-3.....	290
	P-I-1 .....	291
	P-I_A-1 .....	291
	P-ID-1 .....	292
	P-ID-2 .....	292

P-C_I_ID-1 .....	292
P-C_I_ID-2 .....	293
P-C_I_ID-3 .....	293
P-CIA_PR-1 .....	294
P-I_A_ID-1 .....	294
P-ALL-1 .....	295
P-ALL-2 .....	295
P-ALL-3 .....	296
APPENDIX E: Detection Patterns for Security Requirements Engineering .....	297
D-C_PR-1 .....	297
D-ID-1 .....	297
D-I-1 .....	297
D-I-2 .....	298
D-I-3 .....	298
D-ALL-1 .....	299
D-ALL-2 .....	299
APPENDIX F: Response Patterns for Security Requirements Engineering .....	300
R-C_PR-1 .....	300
R-ID-1 .....	300
R-I-1 .....	300
R-A-1 .....	301
R-A-2 .....	301
R-AY-1 .....	301
R-CIA-1 .....	302
R-SEC-1 .....	302

## LIST OF TABLES

Table 1. Search Results by Digital Libraries .....	42
Table 2. Paper Counts at Each Voting Stage .....	43
Table 3. Extracted Data Items.....	48
Table 4. Themes Related to Software Patterns Usage .....	50
Table 5. Details Related to Participants .....	53
Table 6. Factors Used to Group Participants Across Studies .....	55
Table 7. Task Categorization (D: <i>Design</i> ; I: <i>Implementation</i> ; DI: <i>Design and Implementation</i> ) .....	57
Table 8. Details of Software Patterns in Selected Studies .....	58
Table 9. Constructs and Associated Measures Extracted from Included Studies.....	59
Table 10. Threats to Validity of Included Studies .....	61
Table 11. Classification of Replication Studies .....	64
Table 12. Factors Conducive for Replication .....	65
Table 13. Core Set of Security Properties.....	81
Table 14. Security goal patterns.....	86
Table 15. Implied security goals associated with 18 security goal patterns .....	89
Table 16. Metrics used for evaluation of DIGS .....	94
Table 17. Differences between Control and Treatment – DIGS Evaluation .....	96
Table 18. Analysis of variance for Precision and Recall – DIGS Evaluation. ....	101
Table 19. Analysis of variance for standardized responses for the correctly identified Prevention, Detection, and Response security goals. ....	101
Table 20. Analysis of variance for standardized Initial and Implied goals. ....	104
Table 21. Recall of combined security goals – DIGS.....	105
Table 22. Summary of NIST controls by Security Goals .....	126
Table 23. Top 20 ranked attributes for security properties – NIST controls .....	128
Table 24. Top 20 ranked attributes for security actions – NIST controls.....	128
Table 25. Prevention, Detection and Response Patterns for Security Requirements .....	135
Table 26. Classification Guide – Manual Classification .....	144
Table 27. Healthcare Documents and Associated Security Property Counts .....	150
Table 28. Breakdown of Sentences with Explicit or Implicit Security Requirements – Healthcare .....	151
Table 29. Frequently Occurring Property Groups – Healthcare .....	152
Table 30. Top 20 Keywords by Security Properties – Healthcare Documents Study .....	154
Table 31. Ten-Fold Cross Validation – Performance of Classifiers for Healthcare Documents .....	156
Table 32. Classifier Performance by Security Properties for Healthcare Documents – Unbalanced (Original) Training Set.....	158
Table 33. Classifier Performance by Security Properties for Healthcare Documents – Balanced Training Sets .....	159
Table 34. Top 20 Keywords by Security Properties – Networks Documents Study .....	164
Table 35. 10-Fold Cross-Validation of Domain Classifier (Networks).....	165

Table 36. Classifier Performance across Domains – .....	169
Table 37. List of Security Requirements Templates.....	177
Table 38. Example Security Requirements Templates.....	178
Table 39. Metrics used for evaluating participants' responses. [ <i>w.r.t: with respect to</i> ] .....	181
Table 40. Frequency of Participants' Academic and Work Experience across Studies. (CS: <i>Computer Science</i> ; SE: <i>Software Engineering</i> ; Sec: <i>Security</i> ).....	185
Table 41. Summary of Context Factors across Different Experiments. [ <i>*metrics expected to be affected by differences in study context</i> ] .....	188
Table 42. Number of Participants in Each Group – Security Requirements Templates Study. ....	193
Table 43. Security requirements templates associated with use case sentences in the oracle. ....	196
Table 44. Overall mean scores for all metrics across studies. [ <i>*scaled for comparison</i> ] ...	200
Table 45. Results of 2x2 ANOVA for NCSU13.....	203
Table 46. Results of 2x2 ANOVA for UT14.....	205
Table 47. Results of 2x2 ANOVA for NCSU14.....	207
Table 48. Results of 2x2 ANOVA for UCR15.....	210
Table 49. Difference between treatment and control group means across studies. [ <i>*Significant at p-value &lt; 0.1; **Significant at p-value &lt; 0.05</i> ].....	213
Table 50. Results of 3-way ANOVA for combined data from all studies.....	218
Table 51. Group means and variances for the combined data from all the studies. ....	220
Table 52. Requirements Artifacts – Industrial Case Study.....	253
Table 53. Coverage of Security Requirements Patterns for Security Requirements in PSB.....	255
Table 54. Breakdown of Security Requirements Patterns that cover Security Requirements in PSB .....	256
Table 55. Additional Security Requirements Patterns based on PRD .....	259

## LIST OF FIGURES

Figure 1. Overview of SRD Process .....	5
Figure 2. Example of SRD Process.....	7
Figure 3. Quality Score of Studies Over the Years.....	47
Figure 4. Constructs Hierarchy Based on the Research Questions.....	52
Figure 5. Frequencies of Constructs Used in Primary Studies .....	60
Figure 6. Frequencies of constructs used in the primary studies for each theme. ....	71
Figure 7. Elements of DIGS Framework .....	78
Figure 8. Template for Security Goals.....	85
Figure 9. Security goals for ‘patient health record’. ....	87
Figure 10. Implied security goals for ‘audit records’. ....	88
Figure 11. Steps for Applying DIGS .....	90
Figure 12. Means and standard errors of Precision and Recall by Task Group and System – DIGS Evaluation. ....	100
Figure 13. Means and standard errors of standardized responses for the correctly identified (a) Prevention, (b) Detection, and (c) Response security goals.....	102
Figure 14. Means and standard errors of standardized Initial and Implied goals correctly identified. ....	103
Figure 15. Breakdown of Security Goals in the Oracle vs. Identified – iHRIS.....	106
Figure 16. Breakdown of Security Goals in the Oracle vs. Identified – Cyclos.....	107
Figure 17. Classifier Performance for Healthcare Documents – Balanced (B) vs. Unbalanced (U) Training Sets.....	160
Figure 18. Percentage of Sentences Implying each Security Property – Artifacts from Networks and Healthcare Domains .....	163
Figure 19. Classifier Performance for Networks Requirements Sentences – Balanced (B) vs. Unbalanced (U) Training Sets .....	167
Figure 20. Precision and Recall Comparison across Different Training Sets for SMO – Networks Domain .....	168
Figure 21. Task screen for treatment group in UCR15.....	194
Figure 22. Box-plots with results from UT14 across each factor and metric .....	206
Figure 23. Box-plots with results from NCSU14 across each factor and metric.....	209
Figure 24. Box-plots with results from UCR15 across each factor and metric .....	212
Figure 25. Mean scores for treatment and control groups across studies .....	216
Figure 26. Relation among tasktime and metrics of quality, coverage and relevance for all participants.....	228
Figure 27. Requirements in the oracle identified per security property .....	232
Figure 28. Feedback related to the use of security requirements templates .....	235
Figure 29. An Overview of Research Contributions .....	261

## LIST OF ABBREVIATIONS

A	Availability
ACR	Access Control Rules
AY	Accountability
C	Confidentiality
DIGS	Discovering Goals for Security (framework)
DL	Description Logic
ER	Entity Relationship
FOL	First Order Logic
I	Integrity
ID / IA	Identification and Authentication
<i>k</i> -NN	<i>k</i> -Nearest Neighbor
NB	Naïve Bayes
PR	Privacy
RE	Requirements Engineering
RML	Requirements Modeling Language
SIREN	Spiral Model for Requirements Engineering
SMO	Sequential Minimal Optimization
SQUARE	Security Quality Requirements Engineering
SRD	Security Requirements Discoverer (process)
SRE	Security Requirements Engineering
SVM	Support Vector Machine
UCR	University of Costa Rica, Costa Rica
UT	University of Trento, Italy

# 1 INTRODUCTION

Security is an important consideration for any software system that manages valuable data and processes. If a system is not adequately secure, operational, reputational and business stakes rise significantly for both the users and owners of such a system. According to Tondel et al., (I.A. Tondel, M.G. Jaatun, and Meland 2008) an average organization suffers an approximate cost of \$230,000 each year for software security-related losses, after accounting for losses related to physical thefts and viruses. Most of this cost can be attributed to software vulnerabilities and configuration errors. According to the National Vulnerability Database maintained by US National Institute for Standards and Technology (NIST)<sup>1</sup>, 92% of the reported vulnerabilities are in the applications and not due to insecure networks. These facts and figures underscore the significance of incorporating secure software development practices as part of the system development.

Security requirements engineering can provide a foundation for developing secure systems. Despite the availability of methods and processes for security requirements engineering (Fabian et al. 2010), teams often do not focus on security during early stages of software development (D. Mellado et al. 2010). Existing approaches outline the various steps involved in identifying security requirements, but leave the task of executing these steps for the requirements engineer, who may not be an expert in security. Security standards and certifications provide lists of security requirements grouped by category, but lack the conceptual scaffolding to enable software developers to identify when a particular

---

<sup>1</sup> <http://nvd.nist.gov/>

requirement applies to their system. Success of current security assurance techniques like penetration testing, that occur late in the product's lifecycle, vary based on skill, knowledge, and experience of testers and provide little support earlier in the lifecycle (B. Arkin, S. Stender, and McGraw 2005).

Errors in the identification of security requirements can lead to serious security concerns for the software system that impact core functionality, leading to loss of reputation, financial penalties, and even legal prosecution. According to Walia and Carver's classification of requirements errors (Walia and Carver 2009), lack of adequate knowledge or expertise is one of the most commonly identified reasons for requirements errors. Moreover, omissions in requirements is the most commonly occurring requirements error (Alshazly, Elfatraty, and Abougabal 2014). Given that security expertise is limited and minimal resources are available for eliciting security requirements, security requirements are even more likely than other types of requirements to be left unspecified or inadequately specified (Riaz and Williams 2012). Feedback from practitioners across various organizations emphasize that a tool-assisted process to aid in security requirements elicitation may facilitate secure software development (Salini and Kanmani 2012). By automating parts of the security requirements engineering process and providing guidance on applicable security requirements, we can enhance the ability of security non-experts to actively consider security concerns.

Our research on identifying applicable security requirements for software systems is guided by the following primary motivations:

(I) Software systems that share security properties, such as confidentiality or integrity, also have similar sets of security requirements (Firesmith 2003);

(2) Security requirements, if specified at the right level of abstraction, can be reusable across multiple systems, even as a set to meet the same security property (Firesmith 2004)(Mellado, Fernández-Medina, and Piattini 2007);

Natural language requirements artifacts, such as requirements documents, often explicitly state the security requirements for software systems. However, functional requirements of the system not related to security may also have additional security implications (Slankas and Williams 2013) resulting in implied security requirements. For example, consider the natural language requirements sentence "The system shall provide a means to edit discharge instructions for a particular patient."<sup>2</sup> This sentence does not explicitly state a security requirement but implies security requirements for confidentiality (*of patient's discharge instructions*), integrity (*when editing*) and accountability (*who performed the edits*). Security requirements that can be generated include:

- "The system shall enforce access privileges that enable authorized users to edit discharge instructions for a particular patient." (confidentiality)
- "The system shall ensure that the edits do not compromise the integrity of the discharge instructions." (integrity)
- "The system shall log every time discharge instructions for a particular patient are edited." (accountability)

Such implied security requirements should be identified and specified to create a comprehensive set of security requirements for the system.

---

<sup>2</sup> <http://www.hl7.org/>

The goal of this thesis is *to support secure software development by systematically identifying and specifying implied security requirements of a system using empirically developed security requirements patterns.*

## **1.1 Solution Methodology**

To meet the overall goal, we have developed a semi-automated process<sup>3</sup>, Security Requirements Discoverer (SRD), for identifying candidate security requirements for a system based on natural language requirements artifacts. Our process takes natural language requirements artifacts (requirement specifications, feature requests, etc.) and a labeled corpus of sentences for the current problem domain as input. At the end of the process, we provide a set of candidate security requirements for the system as output. We provide an outline of the process in Figure 1.

The inputs to SRD process include natural language requirements artifacts for which we are interested in identifying security requirements. A domain corpus of requirements sentences labeled with implied security properties (such as confidentiality, integrity, accountability) and a security requirements patterns repository that supports specification of requirements related to the identified properties are also provided as input. The domain corpus and patterns were developed as part of this research to support the various steps in the SRD process. The domain corpus is specific to a particular domain however the security requirements patterns are reusable across multiple domains.

---

<sup>3</sup> Source code available at: <http://go.ncsu.edu/securitydiscoverer/>

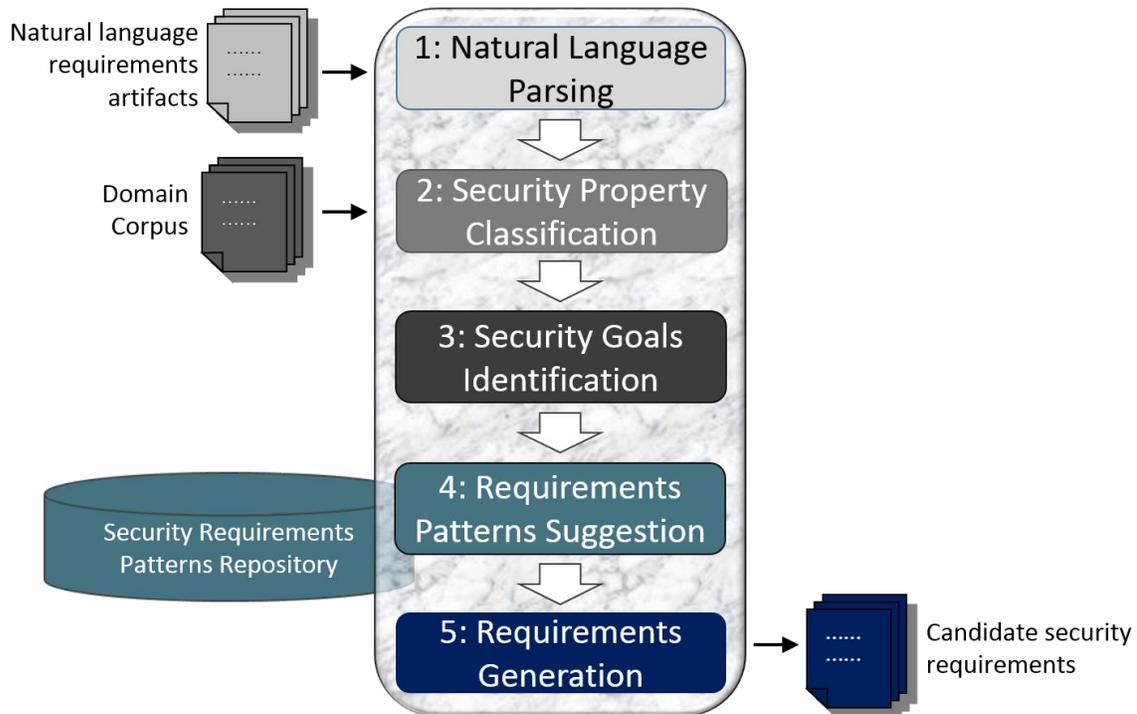


Figure 1. Overview of SRD Process

*Step 1:* As a first step of the SRD process, we automatically parse the natural language requirements artifacts provided as input to identify individual sentences in the input artifacts (Section 7.1.1).

*Step 2:* In the second step, we identify security properties implied by each sentence in the input, such as confidentiality or integrity (Section 5.1.3). To automate the process for identifying security properties, we leverage supervised machine learning to train a classifier that learns the features in natural language sentences that correspond with various security properties in the domain corpus (If a domain corpus is not available, we outline steps for creating the domain corpus in Section 7.1.2). We elaborate on the activities involved in the automatic identification of security properties in Chapter 7. We have applied our process for identifying security properties for the healthcare domain (Section 7.2), networks domain

(Section 7.3 and Section 9) and also classified NIST security and privacy controls (Section 6.3).

*Step 3:* In the third step of the SRD process, we identify security goals for assets based on the classified security properties. For instance, if a sentence implies a need for confidentiality, we create goals for preventing, detecting and responding to the breach of confidentiality of the asset that is indicated in the sentence. We also identify implied goals based on the initial goals such as need for availability of authentication or cryptographic services. Multiple sentences might imply the same goals if they have the same assets and related actions. We have developed a framework for systematically discovering the initial and implied goals, DIGS, and provide details of DIGS in Chapter 5.

*Step 4:* In the fourth step of the SRD process, we suggest applicable security requirements patterns from a patterns repository based on the security goals identified from the input artifacts. Since we are suggesting security requirements that are implied, rather than explicitly stated in the input requirements artifacts, the security requirements patterns repository provides a reusable solution for phrasing the implied security requirements. The repository is empirically developed (Chapter 6) and catalogued in terms of the security goals for preventing, detecting and responding to breaches of various security properties to provide mapping between security goals and security requirements patterns supporting those goals.

*Step 5:* In the fifth and last step of the SRD process, an analyst can select the patterns that are applicable and instantiate the patterns in the context of the system to specify security requirements. Parts of the requirements templates, provided in the solution section of the suggested patterns, can be filled in from the words that appear in the input sentence (actor,

action, asset etc.). We evaluate the use of security requirements patterns in identifying and specifying security requirements in Chapter 8 and Chapter 9 through empirical studies.

We provide an example to illustrate how the SRD process works in Figure 2.

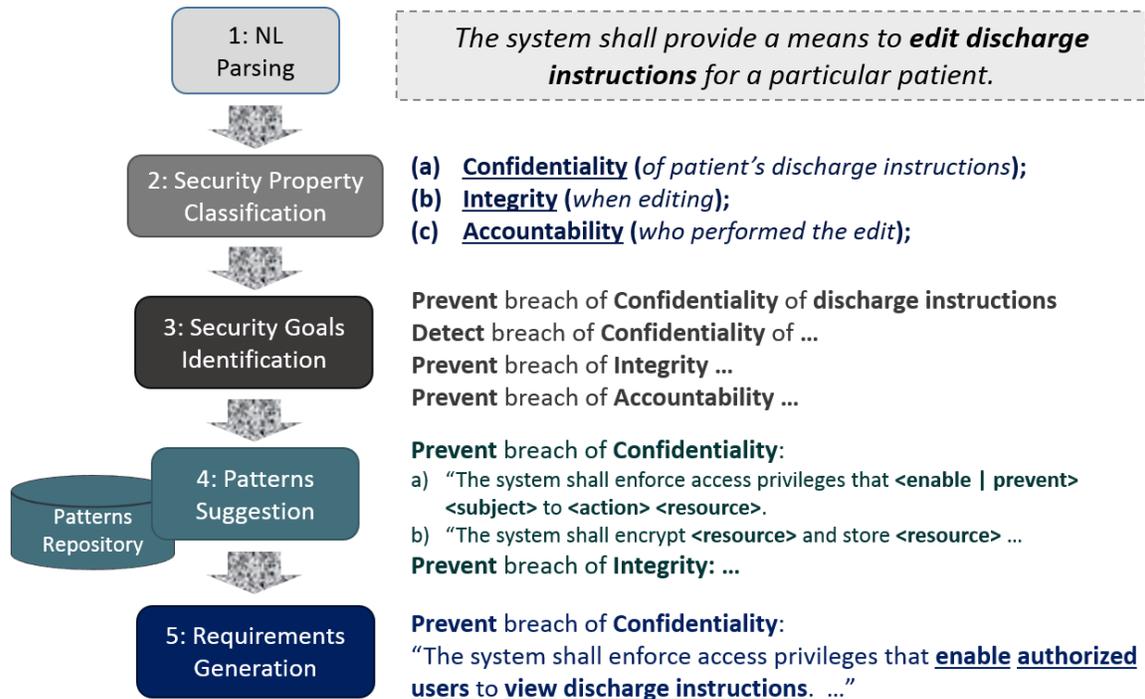


Figure 2. Example of SRD Process

For the example input sentence in the figure, SRD process identifies the security properties of confidentiality, integrity and accountability in the classification step. In the next step we create security goals for preventing, detecting and responding to the breach of confidentiality, integrity and accountability. Security requirements patterns, developed by the researchers, are then suggested by the process. Each pattern is associated with security goals indicating the security properties as well as the conditions under which the patterns becomes applicable (e.g., preventing or responding to breaches, other contextual information such as type of resource to be protected). In the figure, we also show a snippet of the suggested

template for ‘authorized access’ to prevent a breach of confidentiality. The requirements templates suggested as part of the pattern are then instantiated by filling in subject (actor), resource (asset) and action elements from words in the input sentence to generate security requirements.

We solve the following challenges to support our solution methodology.

**Challenge 1:** *Development of a conceptual framework for identifying a core set of security properties.* The framework should identify a comprehensive set of categories of security properties and outline when each security property is indicated. We develop a framework, DIGS (Discovering Goals for Security), that identifies a core set of six security properties as well as goals related to preventing, detecting and responding to the breach of each property. The security properties identified as part of the DIGS framework are used for categorizing sentences in the input artifacts as well as for structuring the security requirements patterns in the repository. (Chapter 5)

**Challenge 2:** *Automatic identification of security properties implied by natural language requirements artifacts.* We develop a validated domain corpus (a labeled corpus of sentences indicating the security properties that are implied by each sentence) to support automatic identification of security properties in natural language artifacts. We train classifiers for making predictions based on the features learned from sentences in the domain corpus. We evaluate multiple classification algorithm and identify factors that may influence classifiers’ performance. (Chapter 7)

**Challenge 3:** *Specification of implied security requirements for the identified security properties.* The security requirements we are identifying as part of SRD are implied, rather

than explicitly stated in the input artifacts. For each security property, we need to identify security requirements that support the property and provide a reusable means for phrasing the corresponding requirements. We empirically develop a repository of security requirements patterns that helps translate security properties implied by individual sentences into security requirements that meet specific goals. (Chapter 6)

*Challenge 4: Empirical evaluation, replication and synthesis of findings.* We conduct multiple experiments, replications and case studies to evaluate parts of our process. We identify considerations related to design, execution and analysis of experiments to support replication and synthesis across multiple studies. (Chapter 4, 8)

## 1.2 Research Questions

To meet our research goal, we investigate these corresponding research questions:

- **RQ1:** What is the core set of security properties that can be used to classify security goals and requirements of a system? (Chapter 5)
- **RQ2:** What features in the natural language requirements artifacts can help in the identification of security properties that are implied by individual sentences? (Chapter 7)
- **RQ3:** How effectively can we automate the process of identifying security properties that are implied by individual sentences in the natural language requirements artifacts? (Chapter 7)
- **RQ4:** What is the effectiveness of automatically-suggesting security requirements patterns based on the implied security properties in supporting the identification and specification of security requirements for a system? (Chapter 8)

- **RQ5:** How effectively can our process be used to identify and specify security requirements for a software system? (Chapter 9)

### 1.3 Thesis Contributions

We contribute the following with this thesis:

- An empirically evaluated framework, DIGS, for systematically discovering security goals for a software system.
- A systematic process for identifying security requirements patterns from existing security requirements knowledge sources.
- A catalog of 35 security requirements patterns based on NIST Special Publication 800-53<sup>4</sup> to generate security requirements operationalizing security goals.
- A semi-automated process, Security Requirements Discoverer (SRD), for identifying and classifying security properties implied by sentences in natural language requirements artifacts to identify a set of candidate implied security requirements for the system.
- Empirical evaluation of SRD in classifying sentences in natural language requirements artifacts in terms of the implied security properties through multiple case studies of documents from different domains.
- Empirical evaluation of SRD related to the use of security requirements templates for identifying and specifying security requirements implied by natural language requirements artifacts through multiple controlled experiments.

---

<sup>4</sup><http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf>

- An empirical evaluation of SRD through an industrial case study.
- A systematic map of empirical literature that draws together research evaluating software pattern application in problem solving.
  - A classification of measures used to evaluate success when applying software patterns and approaches to minimize subjectivity during evaluation.
  - Compilation of reported threats to validity of studies evaluating software pattern application.
  - Considerations to support replication and comparability of study findings.

## 1.4 Dissertation Structure

The rest of this dissertation is organized as follows:

- *Background and Related Work:* We present the background topics in Chapter 2 followed by related work in Chapter 3.
- *Systematic Reviews:* In Chapter 4, we document our systematic mapping study on the empirical evaluation of software patterns involving human participants.
- *Frameworks and Patterns:* In Chapter 5, we present our framework, DIGS, for discovering security goals. We also provide findings from empirical evaluation of DIGS through a controlled experiment. In Chapter 6, we present our systematic methodology for developing security requirements patterns as well as resultant patterns that we have identified to help in operationalizing security goals and generating security requirements. We also provide the 35 security requirements

patterns we have developed to support security goals related to preventing, detecting and responding to security breaches in the Appendices D-F.

- *Methodologies for Automation:* In Chapter 7, we present our methodology for automatically identifying security properties implied by sentences in natural language requirements artifacts that are indicative of the security goals for the software system through a case study of multiple documents from healthcare domain.
- *Empirical Evaluation:* In Chapter 8, we present findings from a series of four empirical studies on the use of security requirements templates. In Chapter 9, we provide an evaluation of our process through an industrial case study in the networks domain.

We conclude this thesis in Chapter 10.

## 2 BACKGROUND

We present the background concepts in this chapter.

### 2.1 Software Requirements

Software requirements capture the description of a software system in terms of its intended purpose (Nuseibeh and Easterbrook 2000). We summarize background concepts related to software requirements that are relevant to our research in this section.

#### 2.1.1 Software Requirements Models

Software requirements models provide a basis for eliciting and capturing software requirements as well as analyzing whether the requirements have been incorporated in the software system. Lamsweerde (Van Lamsweerde 2001) has categorized existing requirements modeling approaches in terms of semi-formal and formal models. Standard techniques used for semi-formal requirements specifications models include entity-relationship (ER) diagrams, state transition diagrams, and data flow diagrams. Semi-formal models conceptualize the comprising units of the model and their links. However, the reasoning and assertions around these units is carried out informally and is open to multiple interpretations. Formal modeling techniques include history-based specification, such as temporal-logic; state-based specifications, such as Z notation; and transition-based specifications such as transition functions that map input state to output state based on triggering events. In terms of formal models, requirements modeling language (RML) (Greenspan, Mylopoulos, and Borgida 1982) was the main effort at using knowledge representation techniques in RE. RML is based on First Order Logic (FOL) for modeling and

reasoning about requirements. Description logic (DL), a subset of FOL, has been used in recent years for reasoning about conflicting privacy requirements (Travis D Breaux, Hibshi, and Rao 2014).

Formal specifications allow reasoning and analysis of the system in terms of the specified properties. However, formal specifications are often less accessible to practitioners, in terms of usability, as compared to specifications based on natural language or standard notations. Balancing the concerns of analyzability and usability is a prevailing challenge in requirements engineering (Van Lamsweerde 2001). Gervasi and Zowghi (Gervasi and Zowghi 2005) have employed natural language parsing techniques to reason about inconsistencies in natural language requirements. They address some of the usability challenges by automatically extracting propositional logic formulae from controlled natural language and supporting conversion of the logic formulae back to natural language.

Reasoning about completeness of requirements is difficult in the absence of a completion criteria. Goal-oriented requirements engineering addresses this challenge to some extent by providing an intuitive framework for reasoning about completeness (Yue 1987). Goals can provide a criterion for completeness of requirements where all the specified goals should be operationalized by a requirements specification. Goals are also more stable than requirements and provide rationale for the requirements, addressing the 'why' aspect of requirements engineering (Anton and Potts 1998).

### **2.1.2 Inferring Specifications from Natural Language Text**

A number of approaches have leveraged natural language processing, information extraction and machine learning based approaches to extract formal specification from

natural language artifacts in recent years. Slankas et al. (Slankas et al. 2014) have proposed an approach for automatic extraction of access control rules (ACR) from policy statements in natural language artifacts. The approach uses frequently-occurring noun phrases in the documents as seeds to identify ACR patterns using an iterative algorithm. Gao and Singh (Gao and Singh 2014) have proposed an approach for extracting normative contracts from natural language business contracts for multiagent systems. Rahul et al. (Pandita et al. 2012) have extracted code contracts from API documents to support tool-based verification of these contracts. Massey et al. (Massey et al. 2013) have leveraged automated text mining approaches for analyzing the policy documents to identify requirements expressed as privacy protections or vulnerabilities.

In our research, we leverage natural language requirements artifacts to automatically identify security implications of functional requirements. These security implications lead to the identification of security goals and specification of implied security requirements that support the goals.

## **2.2 Software Patterns**

While the availability of frameworks and tools enable developers to build more complex systems by reusing software components, software development remains a knowledge-intensive problem solving activity, largely relying on available skills and expertise (Robillard 1999). Experts in any field are rare; capturing and sharing expert knowledge helps to raise the baseline performance of a novice (or non-expert) (Charness and Tuffiash 2008). Software patterns are a popular approach for capturing and sharing expert knowledge for constructing successful solutions to recurring problems (D. C. Schmidt, M. Fayad, and Johnson 1996).

### **2.2.1 Related Concepts**

Patterns are closely related to the concept of problem schemas. A problem schema is a mental structural representation that facilitates problem solving by grouping problems into categories that require similar solutions (Cooper and Sweller 1987). Similarly, software patterns represent a structural unit that captures the knowledge of an expert in solving common software problems that require similar solutions. Software patterns, like problem schemas, allow structuring of information such that the essential commonalities of the solution are abstracted, while leaving out specific details. Kohls et al. (Kohls and Scheiter 2008) have discussed this relation between design patterns and schema theory in detail, noting that patterns can be used to transfer knowledge and expertise by facilitating the creation of mental schemas in the recipient.

Jackson (Jackson 2005) argues that once the requirements, design and solution for a problem have been successfully formulated, it may be counterproductive to rework the problem from scratch. Software patterns essentially capture reusable solutions to common software problems that have proven to be successful over multiple instances. In this sense, software patterns are closely related to Jackson's problem frames. A problem frame is a recognized class of problems specified primarily in terms of problem context, problem and solution domains, phenomena and requirements. The class of problems in a problem frame have a recognized solution. Problem frames can thus be considered problem patterns.

### **2.2.2 Adoption of Patterns in Software Engineering**

Software patterns have been used during various phases of a software development lifecycle. Patterns have been developed for requirements (Withall 2007), analysis (Fowler

1997), design (Gamma et al. 1995), architecture (Fowler 2002), testing (Smith and Williams 2012), security (N. Yoshioka, H. Washizaki, and Maruyama 2008) and configuration management (Berczuk 2003), among other phases of a software development lifecycle. In requirements engineering, Jackson introduced problem frames as a means to characterize requirement patterns (Jackson 2005). Yang et al. (J. Yang and Liu 2008) have integrated problem frames with goals to model requirement patterns. Breaux et al. show how grounded analysis can be applied to a set of requirements to identify several patterns that can be used to refine regulatory codes into functional requirements (T D Breaux et al. 2008). Security patterns have been documented by Schumacher et al. (Schumacher et al. 2006) including patterns for secure design and architecture as well as security requirements. In a recent systematic review of security patterns research, Yurina Ito et al. (Ito et al. 2015) have identified a need to investigate the use of security patterns in early phases of software development including analysis and requirements.

### **2.2.3 Empirical Evaluation of Software Patterns**

The software patterns community has attributed several benefits to the use of patterns during problem solving, such as: facilitation and ease of reuse; identification and capture of abstract concepts; aid in defining interfaces and interactions; means of shared documentation; construction of software with defined properties; and provision of common vocabulary (Budgen et al. 2008). Despite the popular adoption of patterns in software engineering, how do we know that software patterns meet their goal of successfully capturing and sharing expert knowledge across the software development community to support problem solving? Applying software patterns during problem solving, much like most of software engineering,

is a human-centered activity. The merit of a pattern can be assessed in terms of how successfully a practitioner employs a pattern to solve a particular commonly occurring problem. Empirical research evaluating the use of software patterns by software engineers (i.e., studies that involve human participants) can help in establishing whether a given pattern helps in problem solving. A number of empirical approaches have been used for evaluating software patterns including studies involving human participants (Yskout, Scandariato, and Joosen 2012).

A number of secondary studies have been conducted in recent years that assess the impact of design and architecture patterns on various software quality attributes. Zhang and Budgen (C. Zhang and Budgen 2012) conducted a mapping study to identify the usefulness and usability of Gang of Four (GoF) design pattern application in an empirical setting. With the exception of the role of design patterns in maintenance, Zhang and Budgen did not identify any firm support for the benefits attributed to design patterns based on their analysis of 11 selected papers. They suggest that future studies on empirical evaluation of design patterns should focus on the role of design patterns in maintenance.

Ampatzoglou et al. (Ampatzoglou, Charalampidou, and Stamelos 2013) have conducted a mapping study to investigate the effect of GoF design patterns usage on software quality attributes. They classify the studies into primary areas of research including formalization, detection, and application of design patterns. Based on the findings reported in primary studies, researchers found contradictory evidence related to the effect of design patterns on software quality. Researchers noted that design pattern detection and effect of design patterns on software quality are the most active areas of research related to GoF patterns. Ali and

Elish (M. Ali and Elish 2013) have also conducted a survey on the impact of GoF design patterns on software quality. They note that only a subset of Gof design patterns have been empirically evaluated and only four quality attributes have been explored related to GoF design patterns.

Galster and Avgeriou (Galster and Avgeriou 2012) have analyzed the impact of patterns related to the domain of service-oriented architecture (SOA) on various quality attributes. Researchers found a mismatch between patterns for SOA and important quality attributes for the domain of SOA.

Findings of the aforementioned secondary studies indicate an apparent lack of consensus regarding the impact of various patterns on software quality attributes. The findings also highlight a need for further empirical evaluation of software patterns while considering the context in which these patterns are applied to establish applicability and generalizability of the results. We have conducted a systematic mapping study of existing literature on software pattern application in an empirical setting to aggregate and characterize existing research design practices (Riaz, Breaux, and Williams 2015) to support further empirical evaluation, replication and synthesis across the studies.

## **2.3 Automated Text Classification**

This section summarizes background concepts related to using machine learning for text classification relevant to our research.

### **2.3.1 Supervised Machine Learning for Text Classification**

We use supervised machine learning for classifying security properties in natural language requirements artifacts. Supervised machine learning techniques use a labeled set of

data i.e., sentences for which security properties have been identified to learn similarities among the sentences that belong to the same class. We use three different supervised machine learning algorithms including Naïve Bayes (NB), support vector machines (SVM), and  $k$ -nearest neighbor ( $k$ -NN) classifiers. We use a sentence as a unit of analysis for which we want to identify the classification. The set of sentences for which classification is already known and are used to train the machine learning classifiers are called the ‘training set’. The set of sentence for which we want to predict the classification using the trained classifier are called the ‘test set’. The set of all the words appearing in the sentences is the vocabulary or feature set.

Naïve Bayes is a probabilistic classifier. For text sentences, words appearing in the sentences are considered occurring independently of each other (i.e., bag of words assumption). Naïve Bayes classifier computes probability values for a sentence to belong to all the possible classes and assigns the classification with the maximum probability. Naïve Bayes is often used as a baseline method for text categorization as the classification is fast and needs a relatively small training set.

SVM is a non-probabilistic binary classifier. SVM constructs hyperplanes with maximum separation between the two classes such that sentences belong to one class (i.e., sentences that imply a security property) are at greater distance from sentences belonging to the other class (i.e., sentences that do not imply the property). For SVM, we use Sequential Minimal Optimization (SMO) classifier to train the SVM in recognizing patterns in the input. SVM classifiers have higher training times but testing is relatively fast.

For Naïve Bayes and SVM classifiers, we use the Weka (Hall et al. 2009) implementation of the algorithms. We convert the sentence in the training and test sets to word vectors (Joachims 1998) which serve as the features for classification. During the conversion to word vectors, we can apply standard transformations such as stemming, removing stop words and assigning weights to words based on relevance to the sentence and different classifications (Baharudin, Lee, and Khan 2010). Once the features (i.e., word vectors) have been created, we can select a subset of the features that are most useful for predicting a specific security property based on the associated information gain values (Quinlan 1986).

$k$ -NN classifiers work by computing the distance between the current sentence that needs to be classified (test item) and all other sentences that have already been classified. The algorithm then selects the classification for the current sentence to be the same as the majority of the  $k$  sentences with the smallest distance values i.e.,  $k$  nearest neighbors of the sentence that we want to classify.  $k$ -NN classifiers have higher testing times as each sentence has to be compared with all other sentences in the training set. To determine the closest sentence(s), we apply a custom distance function based upon a modified version of Levenshtein distance (Levenshtein 1966) developed by Slankas et al. (Slankas et al. 2014). The distance value acts as the feature used for classification.

### **2.3.2 Metrics for Evaluating Text Classification**

We evaluate the performance of machine learning algorithms for text classification using a stratified  $n$ -fold cross-validation and compute the metrics for precision, recall, and  $F_1$  measure. To compute these metrics, we first need to categorize the classifier's predictions into following four categories.

- True positives (TP) are correct predictions when a classification under evaluation (e.g., security property) is implied by the sentence and also predicted.
- True negatives (TN) are correct predictions when a classification under evaluation (e.g., security property) is not implied by the sentence and neither predicted.
- False positives (FP) are predictions in which the sentence of another classification is incorrectly classified as the one under evaluation (e.g., a security property is predicted but not implied by the sentence).
- False negatives (FN) are predictions in which a sentence of the same classification under evaluation is incorrectly placed into another classification (e.g., a security property is implied by the sentence but not predicted).

We compute the following metrics for evaluating the classifier performance as follows:

- *Precision* (P) is the proportion of correctly predicted classifications against all predictions for the classification under test:  $P = TP/(TP+FP)$ .
- *Recall* (R) is the proportion of classifications found for the current classification under test:  $R = TP/(TP+FN)$ .
- *F<sub>1</sub> measure* (F<sub>1</sub>) is the harmonic mean of precision and recall, giving equal weight to both:  $F_1 = 2 \times (P \times R) / (P + R)$ .

We can also use the metric for accuracy to see the overall correctness of predicting both when a security property is implied and when it is not implied. However, accuracy is a useful measure when the classes are balanced (i.e., equal number of examples with and without a particular security property in the training set).

- *Accuracy* is the proportion of correct predictions against all the sentences:  
 $(TP+TN)/(TP+TN+FP+FN)$ .

With the  $n$ -fold cross-validation, data is randomly partitioned into  $n$  folds based upon each fold of approximately equal size and equal response classification. For each fold, the classifiers are trained on the remaining folds and then the contents of the fold are used to test the classifier. The  $n$  results are then averaged to produce a single result. We follow Han et al.'s recommendation (Han, Kamber, and Pei 2011) and use 10 folds as this produces relatively low bias and variance. The cross-validation ensures that all sentences are used for training and that each sentence is tested just once. We directly utilized Weka classifiers through the available Java APIs utilizing their default options. Since the Weka classifiers do not natively support multiple classifications for an item, we created individual classifiers for each algorithm and classification (security property).

## **2.4 Empirical Software Engineering**

This section summarizes background concepts related to empirical software engineering that are relevant to our research.

### **2.4.1 Empirical Studies**

Empirical studies provide a systematic and scientific approach for collecting evidence to test various hypotheses or address specific research questions. In software engineering, evidence based software engineering has gained focus in recent years (BA A. Kitchenham, Dybå, and Jørgensen 2004). Case studies and experiments are the two types of empirical studies we have used in our research. Case studies involve an up-close or in-depth analysis of one or more 'cases' (such as a person, group, system or software artifact) within the context

(Yin 2009). Case studies are useful when addressing descriptive and explanatory research questions. In contrast, experiments are useful to establish causality or effect of a particular treatment (independent variables) on the outcome (dependent variables) in a controlled setting (Wohlin et al. 2000). The main concern in experiments is to establish validity during design, execution and analysis. Reliability of the findings and support for replication are also important considerations.

#### **2.4.2 Replication Studies**

Replication studies are beneficial to evaluate the validity of prior study findings, either by reproducing results or by isolating factors that can influence results and that lead to variations. According to Lindsay et al., (Lindsay and Ehrenberg 1993) a 'close' replication study attempts to recreate the known conditions of original study and is very similar to the original study. Close replications are often used to establish whether the original outcomes are repeatable at all i.e., initial outcomes were not unduly influenced by confounding factors. A 'differentiated' replication study has a known or deliberate variation in terms of a major effect of the original study conditions. Differentiated replications allow researchers to explore the impact of variations in treatments on the study outcomes (Lindsay and Ehrenberg 1993). If the replication study involves only a subset of tasks or studies a subset of factors, it will be considered a 'partial replication'. In addition to replication, Gomez et al. (Gómez, Juristo, and Vegas 2010) argue that there are other ways of verifying experimental findings, including reproduction and re-analysis of original studies. Where a replication study uses the same method as the original study, a reproduction study (also called a 'conceptual replication') uses a different method to reproduce the findings of the original study. A

reproduction study can establish if the results of the original study are due to the study design or do the findings of the original study hold independently. A re-analysis study (also called an 'internal replication') uses existing data and re-analyzes it using similar or different analysis procedures to see if the results of the original study hold or to develop new insights based on the original data. We have included six replication studies in our analysis and use above definitions from literature to classify these replication studies.

### **2.4.3 Systematic Reviews**

Systematic reviews provide a means for identifying all relevant research related to a topic of interest or to address a research question using a defined and systematic methodology (B Kitchenham and Charters 2007). The systematic review process minimizes potential biases when aggregating evidence from multiple primary sources, such as subconsciously looking for sources that support a particular hypothesis rather than taking into account all the relevant evidence. Systematic mapping studies (or *scoping study*) provide a defined methodology to identify the scope and coverage of research results available in the literature on a software engineering topic of interest (B A Kitchenham, Budgen, and Brereton 2011). Mapping studies are differentiated from Systematic Literature Reviews (SLR) (B Kitchenham and Charters 2007) based on the nature of research questions and the synthesis of collected evidence. Research questions in a mapping study are exploratory in nature and focus on providing insights into the current state of research and practice about a specific topic. Whereas, an SLR focuses on synthesizing results to address a well-defined research question (Silva et al. 2010). The outcome of a mapping study is an identification of relevant studies and structuring of data from these studies based on the research questions. Additionally,

mapping studies can be useful in identifying appropriate methodology for subsequent analysis of the data once the form and extent of dataset has been determined (Budgen et al. 2008).

#### **2.4.4 Validity of Empirical Studies**

Validity of empirical studies is an important consideration when designing studies so that the study findings are reliable and generalizable. A number of threats can arise during the design and execution of the studies. Some threats to validity of empirical studies can be mitigated by case study or experiment design. In situations where design and execution methodology cannot mitigate threats, those threats must be reported as potential limitations. Four broad categories of threats to validity of empirical studies are presented below:

*Internal validity* is the degree to which an investigator can correctly draw causal inferences from empirical data. Issues related to experimental procedures, treatments, and participant experiences may threaten internal validity (Creswell 2003).

*Construct validity* is the degree to which the experiment is measuring what it is purported to measure (Creswell 2003). The use of ambiguous definitions that lead to multiple, even conflicting, interpretations of the experimental construct is a threat to construct validity. Quantitative constructs, such as code size, are easier to measure, whereas qualitative constructs, such as quality and efficiency, are prone to variable interpretations and require the investigator to provide an explicit account of the assumptions that link the measures to the construct. In qualitative case study research, strategies to mitigate threats to construct validity include using multiple sources of evidence, establishing chains of evidence and using key informants to review the study findings (Yin 2009).

*External validity* is the degree to which the results are generalizable to other people, situations or past, present and future events (Creswell 2003). In many cases, these threats cannot be directly addressed as experimental setting is often a scaled-down version of real-world problems in a controlled environment. However, establishing baselines and availability of a repository of standard problem sets can help minimize potential biases.

*Conclusion validity* is the degree to which investigators draw statistically accurate conclusions from the data, for instance by having sufficient statistical power and by satisfying all statistical assumptions (Creswell 2003). The power of an experiment represents the probability of rejecting a null hypothesis when it is false during statistical analysis of the results (Cohen 1988).

### 3 RELATED WORK

We present the related work in this chapter.

#### 3.1 Security Requirements Engineering

Security requirements are often identified from analyzing assets, threats and vulnerabilities while considering multiple points of views such as of users or attackers. Methods for eliciting and documenting security requirements include both frameworks and requirements models.

SQUARE method (Mead, Houg, and Stehney 2005) provides a 9-step process for coordinating various technical activities and artifacts related to security requirements engineering (SRE). Modifications to the SQUARE methodology have been introduced over the years to make the process more light-weight (SQUARE-lite) after observing that the original SQUARE process can take up several months for large projects. Moreover, reusable security requirements have been introduced (SQUARE-lite) on top of SQUARE-lite to have better-quality specifications at lower cost. We share similar motivations and incorporate automation and reuse to reduce the resources needed for security requirements engineering. Our process can be used as a stand-alone or as support during the phases related to identifying assets and goals, eliciting security requirements as well as categorizing requirements.

Franqueira et al. have proposed an approach for security requirements based on assurance arguments (Franqueira et al. 2011). Haley et al. (Haley et al. 2008) have developed a framework that support formal and informal argumentation about security requirements by

constructing a context for the system. The framework supports definition of security goals similar to the prevention goals identified as part of our DIGS framework. With the help of a domain expert, the researchers identify actions that violate the goals and corresponding harm that should be prevented. Security requirements are then specified as specific constraints on specific system functionality to prevent harm. We consider goals related to detecting and responding to potential security breaches in addition to prevention goals. Moreover, we specify security requirements in terms of functions and practices.

Approaches for modeling security requirements include misuse cases (Alexander 2003) (Sindre and Opdahl 2005) and abuse cases (McDermott and Fox 1999) that document an attacker's perspective and support identification of security requirements that mitigate the attack scenarios. Basin and Doser (Basin and Doser 2006) present Model Driven Security approach where security requirements are specified alongside system models for constructing access control infrastructures. A number of researchers have used goal modeling for eliciting security requirements including Secure Tropos (Giorgini, Mouratidis, and Zannone 2006) and extensions such as incorporating notions of delegation and trust (Asnar et al. 2007) and contextual information (R. Ali, Dalpiaz, and Giorgini 2009). Massacci et al. (Massacci, Mylopoulos, and Zannone 2010) have revised and refined the Secure Tropos and i\* modeling languages based on experience through multiple case studies and proposed a consolidated version as an SI\* modeling language. The existing goal modeling approaches can leverage the security goal patterns and implied goals that we have identified as part of DIGS framework to discover an initial set of goals and model additional relations among the goals.

Mellado et al. (D. Mellado et al. 2010) have conducted a systematic review of SRE approaches to summarize existing methodologies. Based on their findings, they provide a set of recommendations include reuse of security requirements elements such as assets, threats, goals and requirements. The researchers also identify the need for automatic integration of security requirements with other requirements of the system. Our process supports the aforementioned recommendations. Fabian et al. also provide a comparison of existing SRE methods (Fabian et al. 2010). Various approaches for identifying security requirements may be used complementary to each other for a comprehensive analysis.

The success of existing SRE approaches varies based on the skill, knowledge, and experience of the analysts. Our approach for identifying security requirements differ from existing modeling-based approaches in that we provide support to an analyst at multiple steps: by automatically identifying implied security properties from existing requirements artifacts, providing a systematic framework for discovering security goals as well as provision of security requirements templates and patterns to identify and specify security requirements.

### **3.2 Knowledge Reuse in Security Requirements**

A comprehensive analysis of security requirements, starting from scratch, is time and resource consuming. A recent case study, documenting the use of SQUARE methodology in SRE, reported an effort of around 12 person-weeks for applying the methodology, with 3 person-days for identifying security goals (Suleiman and Svetinovic 2013).

Security requirements are potentially reusable across systems that share same security properties. Firesmith (Firesmith 2004) proposes the use of parameterized templates to model

reusable security requirements. Whereas patterns also prescribe when a pattern is applicable and how to instantiate it, Firesmith's parameterized templates do not include this information as part of the template.

Toval et al. (A. Toval et al. 2002) present hierarchically structured parameterized and non-parameterized templates for reusable security requirements that adhere to IEEE standards for specifying quality requirements. Some of the quality attributes are: identification (unique), priority, criticality, viability, risk, source, traceability. Toval's requirements elicitation process model is based on spiral model for requirements engineering (SIREN) and explicitly incorporates requirements reuse. A repository for reusable requirements is maintained annotating the domain (e.g., accounting or finance) and profile (e.g., information systems security) indicating when a requirement is applicable for reuse. Toval's reusable requirements are closely related to security requirement patterns. In addition to providing a repository of security requirements patterns, we also automate the process of identifying security implications from natural language artifacts.

Mellado et al. (Mellado, Fernández-Medina, and Piattini 2007) have integrated concepts from SIREN, security modeling approaches such as misuse cases, and common criteria concepts of assurance to propose a reuse-based conceptual framework for systematically identifying security requirements. They outline various steps, similar to SQUARE framework, and discuss where other existing frameworks or models fit at each step in the overall process. The researchers have acknowledged the need for tool support for the proposed process as well as empirical evaluation of the process through case studies to refine and fill-in the details of various steps in the conceptual framework.

Souag et al. (Souag et al. 2015) have recently conducted a systematic mapping study on reusable knowledge in security requirements engineering. They have identified different forms and representations of knowledge reuse. The reusable knowledge is mostly captured in the form of ontologies or taxonomies of threats, vulnerabilities, assets and security requirements. A small percentage of existing approaches also reuse patterns of threats and countermeasures and templates of security requirements. The researchers conclude that existing methods should incorporate more reusable knowledge to support security requirements engineering.

In addition to focus on reusing the knowledge of security requirements, recent efforts have focused on automating parts of the SRE process. Daramola et al. (Daramola, Sindre, and Stalhane 2012) have used information retrieval techniques for identifying nouns and verbs in requirements artifacts to identify themes of threats or vulnerability based on a security ontology. The retrieved information is used to fill requirements boilerplates to generate threat and vulnerability descriptions. Kurt et al. have employed an organizational learning approach (Kurt Schneider et al. 2012) to automatically identify security requirements in existing requirements artifacts for reuse in similar projects. These approaches focus on identifying explicitly stated concepts related to threats and security requirements. Our work differs from these approaches as we identify implied security requirements using supervised machine learning and leverage security requirements patterns to specify the implied security requirements.

### 3.3 Security Requirements Patterns

Security standards and certifications provide lists of requirements grouped by category, but lack the conceptual scaffolding to enable software developers to identify when a particular requirement applies to their system. Security requirement patterns can support the analysis of the security requirements for a system by capturing the context and applicability (i.e., when and how to reuse security requirements) when incorporating particular security requirements. Security requirement patterns available in literature cover only a small subset of security requirements landscape. Withall's (Withall 2007) requirement patterns related to security include access control (registration, authentication, authorization), audit (chronicle), and some aspects of privacy (archiving, comply-with-standard). Schumacher et al. (Schumacher et al. 2006) security patterns catalog includes security requirements patterns on access control, auditing, intrusion detection, non-repudiation and accounting. Both catalogs use natural language pattern representation. Wen et al. (Wen, Zhao, and Liu 2011) have proposed security requirements patterns of ownership, authorization, attack and protection based on problem frames and  $i^*$  for modeling and analysis of assets, threats and attacks.

Security requirements patterns available in literature focus mostly on attack patterns according to a survey by Yoshioka et al. (N. Yoshioka, H. Washizaki, and Maruyama 2008). In a recent systematic review of security patterns research, Yurina Ito et al. (Ito et al. 2015) have identified a need to investigate the use of security patterns in early phases of software development including analysis and requirements. Slavin et al. have developed an approach using inquiry-cycle model to select appropriate security requirements patterns (Slavin et al. 2014). Hibshi et al. (Hibshi et al. 2014) have developed a framework based on situation

awareness for measuring security expertise when analyzing various scenarios to identify threats and vulnerabilities. The researchers have identified various analysis patterns that experts use to identify security threats which are distinct from the analysis patterns of a non-experts. Understanding how an expert approaches a scenario can be useful to train security non-experts and develop patterns of security requirements.

Security requirement patterns available in literature cover only a small subset of security requirements, specifically related to access control, audit and some aspects of privacy (Riaz and Williams 2012). Moreover, the existing patterns mostly cover prevention and detection related requirements. Security requirements engineering can be supported by broadening the scope of security requirements patterns to include countermeasures and response patterns as well. Pattern representation and information content of security requirements patterns also need to be streamlined to facilitate pattern applicability and reuse. We develop a systematic methodology for developing security requirements patterns from existing security requirements knowledge sources to address the above concerns.

### **3.4 Empirical Evaluation of Security Requirements Engineering Methodologies**

A comprehensive analysis of security requirements, starting from scratch, is time and resource consuming. A recent case study, documenting the use of SQUARE methodology, reported an effort of around 12 person-weeks for applying the methodology, with 3 person-days for identifying security goals (Suleiman and Svetinovic 2013). Consequently, empirical evaluation of security requirements engineering approaches within a controlled setting is challenging and not many approaches have been empirically evaluated.

Massacci et al. have conducted a number of case studies using Secure Tropos and i\* modeling notations to identify risks and conflicts between security requirements and system functionality (Massacci and Zannone 2006). The studies led to refinement and consolidation of the modeling languages to support further analysis indicating that empirical evaluation is not only beneficial for assessing a current methodology but also for improving existing methodologies.

Giacalone et al. (Giacalone et al. 2014) have conducted an industrial case study on the use of a lean methodology for assessment of security requirements as change requests are received. The methodology is integrated with the existing production cycle and uses security survey of the architecture and security triage of change requests to evaluate change requests based on the need for security. The case study, conducted over a year, indicates that the survey and triage process significantly reduced the time to identify security requirements.

Taubenberger et al. have developed a security requirements-based risk assessment approach, learning from their experience of performing security risk assessment of two systems (Taubenberger et al. 2011). Taubenberger et al. have also evaluated their approach in resolving errors related to the identification of vulnerabilities in comparison to an existing approach. Their findings indicate that explicitly evaluating security requirements during the course of business can help in resolving vulnerability identification errors (Taubenberger et al. 2013). Karpati et al. have recently reported empirical evaluation of misuse case maps in identifying security threats through two controlled experiments (Karpati, Opdahl, and Sindre 2015). Their findings indicate the usefulness of misuse case maps in comparison to an approach based on the combination of misuse cases and system architecture diagrams.

Misuse case maps helped the students, involved in the experiment, to suggest better mitigations and were also viewed more favorable as compared to the alternate approach. The authors have emphasized the need for replicating the experiments to evaluate the generalizability of findings beyond their current empirical setup.

The recent studies indicate a trend towards increasing empirical evaluation of SRE approaches. Security requirements engineering methodologies are most commonly evaluated through case studies over a period of time. However, parts of the process can be evaluated in a controlled setting. We have conducted a number of controlled experiments to evaluate individual steps in our process related to the identification of security goals and use of security requirements patterns in requirements identification and specification. We have also evaluated the overall process through an industrial case study in networks domain.

## **4 SYSTEMATIC MAPPING STUDY TO UNDERSTAND EMPIRICAL EVALUATION OF SOFTWARE PATTERNS**

As part of the SRD process, we use security requirements patterns to identify and specify security requirements. We have conducted multiple empirical studies and replications to evaluate the use of the patterns as part of the SRD process. To help with the design and execution of our studies, and to support the analysis and synthesis of our findings across the studies, we systematically reviewed existing literature evaluating the use of software patterns by human participants in an empirical setting. Systematic reviews, such as a mapping studies (B A Kitchenham, Budgen, and Brereton 2011), can help gather and categorize existing empirical research design practices, identify common research protocols, context factors and measures used to evaluate software patterns. Such information can be of use to researchers designing new studies or replicating existing studies.

We conducted a systematic mapping study of existing literature on software pattern application in an empirical setting to aggregate and characterize existing research design practices (Riaz, Breaux, and Williams 2015). We characterize the research design in terms of the questions researchers have explored and the context of empirical research efforts including participants, patterns, training and tasks. We also classify the studies by what investigators' measured to evaluate pattern applications, and by threats to validity considered during study design and execution. We have established the following research questions to focus our analysis during the mapping study. We address each of these questions in the section referenced in parentheses.

**MS-RQ1.**What research questions and hypothesis have been empirically investigated in reference to the application of software patterns? (Section 4.2.1)

**MS-RQ2.**What is the context of empirical research efforts, such as participant demographics, patterns and task details, which investigate the application of software patterns? (Section 4.2.2)

**MS-RQ3.**What constructs and associated measures have been used to evaluate the application of software patterns? (Section 4.2.3)

**MS-RQ4.**What threats to validity related to the study design and execution have been considered by the researchers? (Section 0)

**MS-RQ5.** How do replicated studies of software pattern application compare to original studies in term of study design? (Section 4.2.5)

Empirical research evaluating the use of software patterns by software engineers (i.e., studies that involve human participants) can help in establishing whether a given pattern helps in problem solving. However, the empirical evidence on how well various patterns help in problem solving is limited, often inconclusive, and merits further investigation. The context of empirical findings is also not well understood which limits applicability and generalizability of the findings. Furthermore, to lend validity to study findings and isolate confounding factors, empirical studies need to be replicated and synthesized. A clear description of the empirical protocol and a clear description of the context of the study is pre-requisite in replicating and synthesizing studies (Sjøberg, Dyba, and Jørgensen 2007). Our analysis leads to considerations for mitigating some of the limitations of existing research to support replication and synthesis.

## **4.1 Methodology**

We have conducted a systematic mapping study to select and analyze existing literature (Petersen et al. 2008) on the empirical evaluation of software pattern application by students and professionals. We present the details of our methodology below.

### **4.1.1 Exclusion Criteria**

Exclusion criteria are used to narrow down the initial set of search results by the process of elimination. Based on our research questions, we exclude publications that meet the following criteria:

- 1) Publications that are unrelated to the software engineering domain;
- 2) Books and non-peer reviewed publications on software patterns; and
- 3) A peer reviewed publication, in the field of software engineering, not related to empirical evaluation of software pattern application.

Our criteria will exclude publications that are not related to empirical evaluation of software patterns or do not involve application of software patterns.

### **4.1.2 Inclusion Criteria**

Inclusion criteria defines the attributes that are essential for a study to be selected in our analysis. Initially, we considered a stricter inclusion criteria indicating that studies should provide adequate details for various data elements related to empirical design. However no study was solely excluded on the basis of level of details provided. Publications that were not excluded per the exclusion criteria and that meet the following criteria are included in our analysis:

- a) Study describes an actual evaluation of software pattern application in empirical settings, not just a proposal, opinion or pilot study;
- b) Study provides some details related to the empirical protocol and context factors;
- c) Study provides some details related to the criteria for evaluating the outcome;
- d) Study provides some results and discussion based on the effect of using patterns.

Our inclusion criteria would exclude surveys where participants provide their opinion about software patterns (Khomh and Guéhéneuc. 2008) without using patterns in problem solving in an empirical setting.

#### **4.1.3 Search and Selection Process**

We conducted an exploratory search to identify appropriate search terms in accordance with the objective of our mapping study. For this purpose, we conducted search on Google Scholar using various combinations of terms 'software', 'pattern' and synonyms for 'study'. Given the broad meaning of term 'pattern' and the prevalent use of 'software' in diverse domains, we noticed a large number of results that were out of scope for our study. In accordance with our exclusion criteria, we deemed two main types of results as out of scope: (1) literature covering domains other than software engineering e.g., patterns in biology, behavioral sciences, network traffic, etc.; and (2) literature on software patterns covering automated techniques for pattern application, mining, or evolution without involving human participants. Restricting the search to specific subject areas eliminated some of these results. The search results still included extraneous publications that were filtered out based on the inclusion / exclusion criteria.

During the exploratory search, we also noticed that many authors used more specific terms (e.g., 'architectural pattern', 'security pattern', 'requirement pattern') instead of the general term 'software pattern'. However such papers invariably mentioned 'design patterns' as a background concept. Many papers also discussed the introduction of 'patterns' concept from the field of 'architecture'. We also noticed that limiting the search to title, abstract, and keywords returned a very small number of results. Based on the results and insights from the exploratory search phase, we selected the following search term for our study:

- [[software AND pattern] OR [design AND pattern] OR [software AND architecture AND pattern]] AND [evaluation OR experiment OR empirical OR study]

With the selected term, we conducted search on four digital libraries for relevant papers up to Sep 2014. The search term could appear anywhere in the paper. The digital libraries are listed below:

- Google Scholar (Subjects: Engineering & Computer Science, Social Sciences)
- ACM Digital Library
- IEEE Xplore
- Science Direct (Type: Journals; Subjects: Computer Science, Engineering, Social Sciences)

We included 'Social Sciences' in the Science Direct subject because we were interested in the use of software patterns in education, among other areas, and this research can appear in Social Sciences publications (e.g., Computers & Education, Educational Psychology, etc).

Table 1 provides a breakdown of our search results, removing duplicates and only considering papers that seemed potentially relevant given the scope of our study. We use two

types of stopping criteria during our search: (1) we reviewed all results until we counted a fixed number of consecutive results that were out of scope (e.g., ACM and IEEE); and (2) we reviewed all results within a fixed-limit threshold (e.g., Google Scholar and Science Direct). We initially planned on using only the first stopping criteria. However we observed a large number of out of scope results early on in Google Scholar (the first digital library searched) and decided to continue looking at the results beyond the initial stopping criteria. We did not use any automated scripts or tools while searching the first three digital libraries. We walked through the search results in the order of decreasing relevance, as sorted by the digital library, marking potentially relevant papers in the process. In the case of Science Direct, we used EndNote to import citations for first 1000 results returned based on relevance to our search string. These results were sorted in alphabetical order (on the name of each paper's first author) and examined in that order to identify potentially relevant papers.

Table 1. Search Results by Digital Libraries

<b>Digital Library</b>	<b>Search Stopping Criteria</b>	<b>Potentially Relevant Papers*</b>
Google Scholar	First 1000 Results	409
ACM	40 consecutive results out of scope	83
IEEE Xplore	40 consecutive results out of scope	37
Science Direct	First 1000 Results	109
		<b>Total: 638</b>

\* excluding duplicates

Our search queries produced an initial set of 638 potentially relevant papers, excluding duplicates. Four researchers separately voted on the paper titles during the first voting stage. The first author voted on all papers, as indicated in Table 2. The second author and two other researchers voted on subsets of 282, 295 and 61 paper titles, respectively. After applying the

exclusion criteria, 207 papers were selected for voting on abstracts. The first author voted on all abstracts; the second author and two other researcher voted on a subset of 58, 44 and 39 abstracts respectively. We further eliminated 150 of these papers by evaluating the abstracts against the given inclusion and exclusion criteria, yielding a set of 57 papers to be considered in the next phase in which the whole paper is considered. At each stage, a paper with a positive vote from any of the voters was considered for the next stage.

Table 2. Paper Counts at Each Voting Stage

Voting Stage	<i># of papers voted on by:</i>				Selected for Next Stage
	Author 1	Author 2	Researcher 1	Researcher 2	
Paper Titles	577	282	295	61	207
Abstracts	207	58	44	39	57

Next, we performed a snowball search, looking at references of the 57 papers selected after voting on abstracts to identify any further potentially relevant papers. Three additional papers were found as a result, providing a total of 60 papers for the next step in the selection process. The first author applied inclusion criteria on the full papers in this set to yield the final set of 27 papers for our analysis.

To expand the scope of our search, we performed manual search on two journals, the Empirical Software Engineering Journal and Journal of Software: Evolution and Process. These journals are not indexed by the digital libraries used in our analysis but publish similar research. We found some papers on software patterns related to automatic pattern detection, pattern evolution in an artifact, or discussion of various structural or quality aspects of patterns without an empirical evaluation involving software pattern application. However, the search did not result in the inclusion of additional studies based on our criteria.

We used 11 papers on design patterns evaluation, identified in Zhang and Budgen's mapping study (C. Zhang and Budgen 2012), as a quasi-gold standard (H. Zhang and Babar 2010) for our search process. Based on our search string, ideally the search results should include all of the 11 papers as we explore the broader concept of software patterns evaluation. We were able to identify 8 of the 11 papers based on our search string corresponding to a quasi-sensitivity of 72.72%. This is similar to the findings of Dieste et al. (Dieste, Grimán, and N. Juristo 2009) for the digital libraries used in our mapping study and considered acceptable for Software Engineering reviews. Our overall search process was able to identify all 11 papers after the snowball search on papers selected after voting on abstracts.

#### **4.1.4 Included Papers**

Our analysis set includes a total of 27 papers covering software patterns from the domains including object-oriented design, software architecture, software security to collaborative learning and ubiquitous computing. Each paper is assigned a unique reference code for the purpose of our analysis. Twenty-one of the 27 papers document original studies whereas the remaining six report replication studies of one of the original studies. The 21 papers reporting original studies are assigned two-digit numerical codes 01 through 21 prefixed by the letter "S". The papers documenting replication studies are numbered differently, starting with the original study number followed by "-R" and a replication number (e.g. S04-R1 is replication one of study S04). We use these codes throughout the remainder of this paper to refer to individual papers. See Appendix A for the reference codes and bibliographical references of the included studies.

Paper S01 discusses the results of three distinct pattern-related empirical studies. We refer to these as S01-A, S01-B, and S01-C. Paper S03 describes two replications of a controlled experiment, which we refer to as S03-A and S03-B. In total, we examined 30 distinct primary studies (24 original studies; 6 replication studies) of software patterns reported in 27 papers. Five of these 30 studies are case studies and the remaining are experiments.

Studies S03 and S04 have been replicated one and five times, respectively. S03 was replicated by an independent set of researchers whereas a subset of researchers from the original study S04 were involved in all its five subsequent replications. Four of the replications of S04 have been conducted as part of a 'Joint Replication Project' [30] and are numbered S04-R2 to S04-R5. These replications were carried out using a replication web portal. Participants in all four of these replication studies were graduate and undergraduate students, whereas the original study was conducted on professionals. We discuss replication studies in Section 4.2.5 when we address MS-RQ5.

Moreover, studies S03-A and S03-B can be considered close replications of the same experiment. Both experiments are conducted in the same timeframe, by the same researchers, and the conclusions are drawn based on the results of both studies. Studies S14 and S16 are based on the same experiment, but report findings on different factors that affect program maintenance in the presence of deployed design patterns. S16 can be considered a re-analysis study based on the same experiment as reported in S14.

With regard to the different types of replication studies, we are not making any assumptions about the variations we expect to see between original and replication studies.

Any variations that exist between original and replication studies in terms of a study design aspect are reported within the context of the research questions. When summarizing our findings, we specify that these findings are based on X of 24 original studies and Y of 6 replication studies. If there are no variations between original and replication studies in terms of a study design aspect, we report the results in terms of original studies to avoid inflating the use of replicated design practices.

#### **4.1.5 Quality Assessment of Included Studies**

For systematic literature reviews, Kitchenham and Charters (B Kitchenham and Charters 2007) recommend quality assessment of included studies to assess the validity of the evidence as the evidence is synthesized to answer specific research questions. We assess the level of detail for the empirical protocol and various contextual factors provided in the primary studies, which is not typically required for mapping studies. For each paper, we consider the following five questions to assess this level of detail:

- Do the authors clearly state research questions or hypotheses for the study?
- Do the authors describe the sample population in terms of demographics and experience (including knowledge of patterns)?
- Do the authors explicitly describe the patterns they are using and rationale for selecting these patterns?
- Do the authors provide details about their findings and applicability of the findings?
- Do the authors discuss limitations / threats to validity of their study?

In answering each of the above questions, we use a scale from 0-3 (0=Not at all; 1=Somewhat; 2=Mostly; 3= Fully) as used by Kitchenham et al. (Barbara Kitchenham et al.

2012) and compute the overall score by averaging the scores for individual questions. A higher quality score indicates that the researchers have provided a detailed report of the empirical study. Detailed reporting is important for supporting replication and analysis of a study. We provide the results of this quality assessment in Appendix B. Figure 3 provides a breakdown of quality assessment scores over the years in which the primary studies are published. Studies often lacked in details related to the sample population and the threats to validity. However, almost 82% of the included papers have an average score over 2.5 indicating that the studies are of fairly good quality. The quality score for studies published in recent years is consistently high.

The quality scores assigned based on the above criteria reflect our perceived clarity and comprehensiveness of the presentation of information that we aimed to collect for our mapping study. These scores may not be reflective of the quality in terms of empirical design or of the publication venue.

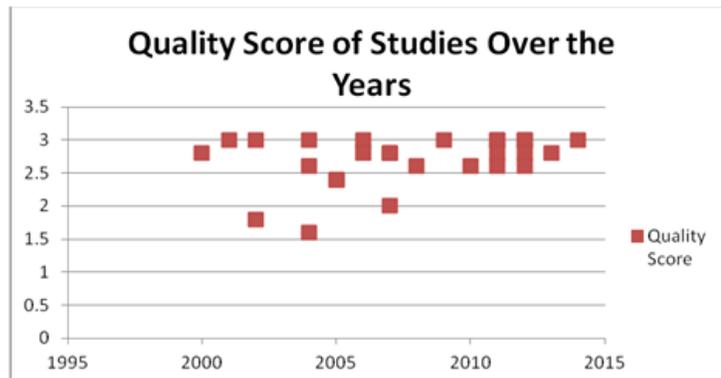


Figure 3. Quality Score of Studies Over the Years

#### 4.1.6 Data Extraction and Analysis Methodology

We have analyzed each study and extracted relevant data to answer our research questions. In Table 3, we list the data items extracted from each study.

Table 3. Extracted Data Items

<b>Category</b>	<b>Data Items</b>
<b>Research Questions</b>	Research questions, hypothesis, assertions, constructs explored based on research questions
<b>Participants</b>	Role, experience level, method of determining expertise, incentive, sample size, grouping factors
<b>Preparatory Material</b>	Preparatory material, method of delivery of material
<b>Tasks</b>	Task type, lifecycle emphasis, number of tasks, magnitude of task, differences in tasks for participants, termination criteria for task
<b>Patterns and Tools</b>	Domain of patterns, patterns studied, rationale for selecting these patterns, programs, tools or frameworks used
<b>Evaluation</b>	Constructs used for evaluation, associated measures used for evaluation, approaches to minimize subjective assessment
<b>Validity</b>	Any reported threats to internal, external, construct and conclusion validity
<b>Replications</b>	Type of replication, motivation for replication, differences in findings between original study and replication

We extract each of the above data item from each study and organize the data from different studies into common categories. In the spirit of Glaser, we perform grounded analysis to allow the structure, categories and themes to emerge from the data (Adolph, Hall, and Kruchten 2008) while guiding our inquiry by our research questions. For instance, consider Table 6 listing factors for grouping participants. In the beginning, we do not have any category for the factors. From the first study, we identify that participants are grouped based on experience so we add a category 'experience-based'. In the second study, we find that grouping is done on the basis of experience as well as based on the differences in information and tasks. Thus we identify two additional categories 'pattern vs. non-pattern information' and 'pattern vs. non-pattern task'. We continue the analysis until we have gone through all the studies, placing studies either in already identified categories or creating new categories based on the data item. In some cases, we might have to combine multiple categories into a single category or divide into distinct categories to manage the level of details and clarity of information. For some data items, such as threats to validity, we have

already established categories in the software engineering literature (Wohlin et al. 2000). We use already established categories where applicable to categorize the data extracted during our analysis. Research questions MS-RQ1 – MS-RQ5 are addressed below by compiling and analyzing study design attributes of the included studies.

## **4.2 Results and Analysis**

We present the results of our mapping study in this section. We report the details related to a replicated study separately only when differences exist between the original study and the replicated study in terms of a particular study design aspect.

### **4.2.1 MS-RQ1: Research Questions and Hypotheses**

*MS-RQ1: What research questions and hypothesis have been empirically investigated in reference to the application of software patterns?*

This research question directs us to understand the specific aspects of software pattern application that the researchers seek to explore and understand. Research questions are interrogative statements that target a single phenomenon or concept and serve as a sign post to the investigator (Creswell 2003). Empirical design and quantitative measurement begins with framing the research questions and deciding what to measure as dependent and independent variables. Hypotheses are falsifiable statements that link the independent and dependent variables and a statistically significant measure is used to attempt to disprove a hypothesis. A null hypothesis states that no statistically significant difference exists in a set of given observations. Whereas an alternate hypothesis is the negation of the null hypothesis. An assertion is a true-false statement, usually without a proof or reasoning. In Appendix C, we reproduce the research questions (prefix  $R_i$ ) and hypotheses (prefix  $H_{0i}$  for Null

Hypothesis; H<sub>ni</sub> for Alternate Hypothesis), verbatim where reported. When a paper supplied none of these explicitly, we list the questions inferred from study description (*italicized*).

Based on the research questions and hypothesis of the primary studies, we identify: a) themes that the studies focus on in terms of software patterns usage, and b) constructs each study is interested in measuring. We identified six broad themes related to software pattern usage. If a study focuses on the role of patterns while modifying an existing solution, we categorize it as related to the theme of 'maintenance'. When new solutions are created, we categorize it as 'construction' theme. Studies that focus on training novices using patterns or introducing practitioners to new domain using patterns are categorized under 'training' theme. Studies where research questions explore how solutions involving patterns compare with alternate solutions are categorized as 'performance' theme. We list all the identified themes in Table 4 below. Note that each study can belong to multiple themes. Most commonly investigated theme, explored in 10 of 24 original studies, and all 6 replications, is the role of software patterns in maintenance-related activities.

Table 4. Themes Related to Software Patterns Usage

<b>Theme</b>	<b>Studies focusing on the theme</b>
<b>Maintenance</b> - <i>modifying existing solutions</i>	S02, S03-A, S03-B, S04, S08, S10, S11, S14, S16, S17
<b>Construction</b> - <i>creating new solutions</i>	S02, S06, S09, S12, S15, S20, S21
<b>Training</b> - <i>learning problem solving in a new domain</i>	S01-A, S01-B, S01-C, S02, S05, S06, S08, S13
<b>Performance</b> - <i>providing better solutions</i>	S04, S06, S07, S10, S12, S19, S21
<b>Documentation</b> - <i>capturing or recovering rationales and decisions</i>	S03-A, S03-B, S15, S17, S18, S20
<b>Communication</b> - <i>supporting team communication</i>	S06, S11, S21

Based on keywords in the research questions, we also identify the constructs each study is interested in measuring when using software patterns, as shown in Figure 4. Constructs provide a granular view of what the researchers are interested in exploring as compared to

the broad themes. In social sciences, constructs are social and psychological concepts that cannot be measured directly, but require measuring one or more variables indirectly (Shadish, Cook, and Campbell 2002). Whereas height, weight and age are commonly accepted variables with standard units of measure, the concepts of productivity, usability and communicability are constructs that are measured indirectly. We organize the constructs as code-level, design-level and knowledge and communication. Some constructs may fall into multiple categories. For instance, 'design skills' is both a design-level and knowledge and communication construct. These constructs also help in identifying subtle differences between the studies. For instance, we can make a distinction between studies on architectural patterns exploring 'usability' versus 'modularity' of the architecture. We may also compare constructs extracted from research questions with the constructs used to actually evaluate the outcome. Ideally these constructs should be similar so the goals, questions and metrics of the study are aligned (V. Basili 1992).

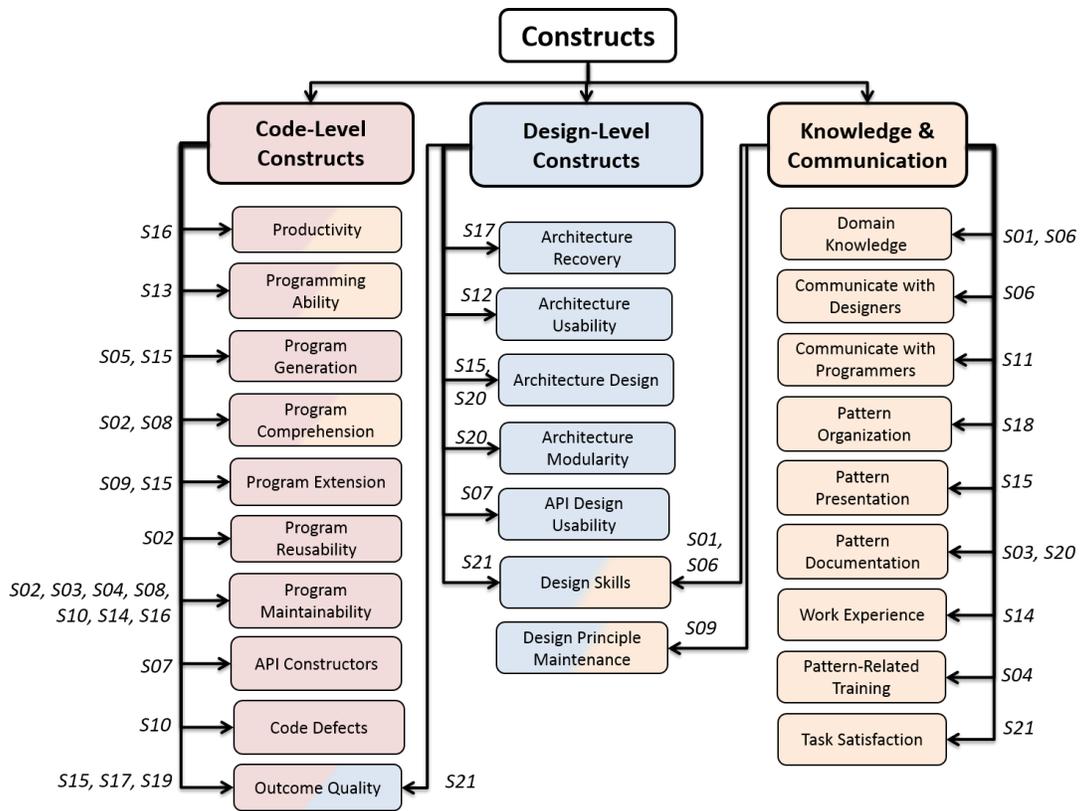


Figure 4. Constructs Hierarchy Based on the Research Questions.

#### 4.2.2 MS-RQ2: Empirical Design Context Factors

MS-RQ2: *What is the context of empirical research efforts, such as participant demographics, patterns and task details, which investigate the application of software patterns?*

This research question directs us to identify 'who' are the participants (demographics etc.) and 'what' task do they have to perform (problem, patterns etc.). The context of empirical research efforts can help in establishing the applicability and generalizability of the findings and support future replications. We discuss each of the context factors below.

*Demographics:* Studies included in our analysis provide varying levels of details related to participants' role, experience level, and the type of incentive given to the participants. They also provide details related to the sample size of the participants. Table 5 provides a classification of the studies in terms of these demographic categories. We observed that 24 of the 30 empirical studies recruited undergraduate or graduate students enrolled in a computer science degree program. Four studies involved a mix of students and professionals while two studies recruited software professionals.

Table 5. Details Related to Participants

<b>Participant Role</b>	<ul style="list-style-type: none"> <li>• <b>Undergraduates</b> (S01-A, S01-B, S03-A, S03-B, S03-R1, S04-R2, S04-R5, S05, S10, S13, S14, S16, S19)</li> <li>• <b>Graduate students</b> (S02, S03-A, S04-R2, S04-R3, S04-R4, S06, S08, S09, S11, S12, S14, S15, S16, S17, S18, S20)</li> <li>• <b>Mix of students and professionals</b> (S01-B, S06, S17, S21)</li> <li>• <b>Professionals only</b> (S04, S04-R1)</li> </ul>
<b>Experience Level</b>	<ul style="list-style-type: none"> <li>• <b>5 Levels:</b> 0-1 yrs, 1-2 yrs, 2-4 yrs, 4-6 yrs, 6 &lt; yrs (S21)</li> <li>• <b>3 Levels:</b> Level 3 (20+ hrs training; limited practical experience), Level 2 (8-10 hrs training; no or some practical experience), Level 1 (2-4 hrs training; no or limited practical experience) (S01-A)</li> <li>• <b>2 Levels:</b> Expert vs. Novice (S01-B, S06, S11); Experienced vs. Inexperienced (S08, S14, S16)</li> <li>• <b>No discrete levels</b> (S01-C, S02, S03-A, S03-B, S04, S05, S07, S09, S10, S03-R1, S04-R1, S04-R2, S04-R3, S04-R4, S04-R5, S12, S13, S15, S17, S18, S19, S20)</li> </ul>
<b>Method of Determining Expertise</b>	<ul style="list-style-type: none"> <li>• <b>Self-assessed</b> (S04, S08, S17, S21)</li> <li>• <b>Test-based</b> (S03)</li> <li>• <b>Prior-performance based</b> (S13)</li> <li>• <b>Employer-rating based</b> (S04-R1)</li> </ul>
<b>Incentives</b>	<ul style="list-style-type: none"> <li>• <b>Volunteer</b> (S01-A, S04-R2, S04, S05, S07, S13, S17, S21)</li> <li>• <b>Assignment Credit</b> (S01-A, S01-C, S02, S04-R4, S04-R5, S09, S11, S16, S18, S19, S04-R5)</li> <li>• <b>Monetary</b> (S04-R1)</li> <li>• <b>Not specified</b> (S01-B, S03-A, S03-B, S03-R1, S04-R3, S08, S10, S12, S14, S15, S20)</li> </ul>
<b>Participant Sample Size</b>	<ul style="list-style-type: none"> <li>• <b>&lt; 20</b> (S01-A, S01-B, S05, S07, S11, S12, S13, S15, S04-R2, S04-R3, S04-R4)</li> <li>• <b>20 - 50</b> (S01-C, S02, S03-B, S04, S06, S08, S17, S20, S03-R1, S04-R1, S04-R5)</li> <li>• <b>51 - 100</b> (S03-A, S09, S18, S21)</li> <li>• <b>&gt;100</b> (S10, S14, S16, S19)</li> </ul>

A majority of the included studies (16 of 24 original studies; all 6 replications) did not specify a discrete experience level for participants as researchers explored factors other than the role of experience level on study outcome. Six studies grouped participants into two discrete levels of expertise. Only two studies used more than two levels of expertise. Study S04-R1 is the only study to offer monetary incentives to participants based on the

participant's experience level, as rated by their employers. Credit for an assignment as part of a course was offered as incentive in 11 studies, considering that majority of the studies recruited students. Nearly half of the studies did not explicitly state the incentives given to participants. We observed that 22 of 30 studies included in our analysis have 50 or fewer participants.

*Groups:* Studies in our analysis used different factors to divide participants into groups. The grouping criteria for each study is listed in Table 6. Most commonly used criteria for grouping participants, used in 13 studies, is to provide one group of participants pattern-related information such as classes that participate in a particular design pattern. The other group is not provided any pattern-related information. Another commonly used factor for grouping participants, used in 9 studies, is to assign pattern versus non-pattern tasks to different participants. For example, one group of participants is given a maintenance task on a program version containing design patterns, and the other group is given the same maintenance task on a different program version without design patterns. The performance of one group of participants is compared against the other group of participants. To minimize presence of confounding factors, groups should be balanced in terms of other factors (Yin 2009), such as background knowledge and experience, unless experience is also a factor in the study as is the case in seven studies. Factors listed in Table 6 are between-subject factors (Lane 2011) in which different groups of participants receive different experimental treatments. Studies S02, S04 and S11 additionally compared performance of each participant pre and post-training related to software patterns. In these studies, pattern-related training is used as a within-subjects factor (Lane 2011) in which same participant receives multiple

levels of a treatment. Each participant is given multiple similar tasks to perform. However, some tasks are given before a course on design patterns and some afterwards. The performance of each participant in the later tasks is compared to their performance in the earlier tasks, prior to the course or training. To ensure that any observed improvement in performance is due to the training and not due to the participant gaining proficiency in performing the task, learning or practice effect should be countered (Yin 2009). One approach is to use counterbalancing by giving tasks in different orders to different participants.

Table 6. Factors Used to Group Participants Across Studies

Factors for Grouping Participants	S01-A	S01-B	S01-C	S02	S03-A	S03-B	S04	S05	S06	S07	S08	S09	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21
Experience-based	X	X							X		X			X			X		X					
Individual vs. pair																								X
Patterns vs. non-pattern tasks		X		X			X			X	X		X				X		X					X
Patterns vs. non-pattern information		X			X	X			X	X	X				X	X	X		X	X		X		X
Less vs. more information			X												X			X			X		X	

Eight studies used multiple factors to group participants, with two studies (S04, S11) using both a between-subjects and within-subjects factor. Study S04 grouped participants based on patterns vs. non-patterns tasks. Additionally, participants performed the tasks before and after training related to patterns. Twenty studies used factors based on differences in information (last three factors in Table 6) to group participants. Two studies (S05, S09) did not divide participants into multiple groups. Study S05 explored the difficulties faced by

novices when using design patterns. Study S09 reported a case study on how well participants, using state design pattern in their solution, adhered to the open-closed design principle.

*Task:* In the studies, participants were asked to perform tasks related to software pattern application. We have identified three different categories of tasks based on lifecycle emphasis:

- Selection Task – participants select a relevant pattern to solve the problem and apply the pattern;
- Instantiation Task – participants are evaluated on their ability to apply a selected the pattern; or
- Maintenance Task – participants are asked to modify existing design or code with instantiated patterns, i.e., the investigators applied one or more patterns beforehand to the software design or code and presented that as part of the task. Understanding how patterns are instantiated in an existing system can help in identifying how to modify the system.

In terms of task type, some studies require participants to complete design tasks (D), others to implement code (I), and some require both (DI). Table 9 summarizes our analysis for lifecycle emphasis and task type. Assignment of studies to task categories is not mutually exclusive as a few studies examined both selection and instantiation of patterns. Eleven studies focus on design tasks only. Twelve studies involve implementation tasks only. One study focuses on both design and implementation tasks (S02). Ten of the studies involving

design tasks are carried out during pattern selection and instantiation. There is no specific lifecycle emphasis for implementation tasks.

Table 7. Task Categorization (D: *Design*; I: *Implementation*; DI: *Design and Implementation*)

Pattern Task Categorization	S01-A	S01-B	S01-C	S02	S03-A	S03-B	S04	S05	S06	S07	S08	S09	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21
<b>Selection</b>	D	D	I	DI				D	D							I		I			D		D	D
<b>Instantiation</b>	D	D	I	DI				D	D	I		I			D	I		I			D	D	D	D
<b>Maintenance</b>					I	I	I				D		I	I			I		I	D				

*Patterns:* The details related to software patterns include not only the domain of the patterns (e.g., object-oriented, software security) but also the exact patterns that were evaluated (e.g., GoF patterns). Additionally, to understand the context in which patterns are used in a study, it is important to document the rationale for selecting particular patterns and the tools or frameworks within which the patterns were applied. We list the details related to the software patterns evaluated in the included studies in Table 8. Not all studies specified the exact patterns that were studied, which can limit the applicability of the study findings and preclude future replications. Fourteen of the 24 included studies (and all six replication studies) evaluate the use of object-oriented design patterns, specifically the GoF patterns. Four studies evaluate software architectural patterns, with two of these studies looking at J2EE patterns. No other patterns are explored in multiple studies. No particular tool or framework is used across multiple studies except for JHotDraw<sup>5</sup> application employed in three studies, all conducted by the same group of researchers.

---

<sup>5</sup> <http://www.jhotdraw.org/>

Table 8. Details of Software Patterns in Selected Studies

Category	Study-wise Breakdown
<b>Pattern Domain</b>	<ul style="list-style-type: none"> <li>• Object-oriented (S02, S03-A, S03-B, S04, S05, S07, S08, S09, S10, S11, S13, S14, S16, S21)</li> <li>• Software architecture (S12, S15, S17, S20)</li> <li>• Software security (S18, S19)</li> <li>• Ubiquitous computing (S06)</li> <li>• Collaborative learning (S01-A, S01-C)</li> <li>• Simulation environment (S01-B)</li> </ul>
<b>Patterns; Rationale for Selecting Patterns</b>	<ul style="list-style-type: none"> <li>• Ten ThinkLets; Selection based on research, usage and expert opinion (S01-A);</li> <li>• Building blocks within Arena simulation environment; Selection based on popularity and availability of simulation experts and vendors (S01-B);</li> <li>• Patterns for computer-mediated interaction; Selected as part of lab curriculum (S01-C);</li> <li>• GoF Design Patterns (S02, S03-A, S03-B, S04, S05, S07, S08, S09, S10, S11, S13, S14, S16 ); Selected as part of course curriculum (S02, S09); Selection based on four most frequently occurring design patterns (S14, S16); Selected to study properties of a particular pattern or set of patterns (S04, S07, S08, S09, S10); Used to study the effects of distributed cognition (S21);</li> <li>• Usability-Supporting Architecture Patterns (USAP); Selected based on the need to support a specific usability concern (S12);</li> <li>• J2EE Patterns, Architectural design patterns (S15, S17); Selected to support development of mobile applications (S15); Selected based on 'most well-known' architectural patterns (S17); Used to study patterns participants relationships (S20);</li> <li>• Security patterns; Selected to meet the needs of healthcare security and privacy scenarios (S18);</li> <li>• Pattern-based method for Secure Development (PbSD); Selected based on problem domain (S19);</li> </ul>

### 4.2.3 MS-RQ3: Constructs and Measures

MS-RQ3: *What constructs and associated measures have been used to evaluate the application of software patterns?*

This research question directs us to identify and categorize the commonly used constructs and measures. Categorizing commonly used measures makes it more likely that same measures would be used for the same construct in similar situations, supporting comparison and synthesis of findings. We have categorized the included studies in terms of 10 different constructs with 31 associated measures based on the evaluation criteria used to assess the outcome, as given in Table 9. Studies in our analysis often used multiple measures, 3-4 different measures on average. In Figure 5, we summarize how frequently each of the 10 constructs is used in the primary studies.

Table 9. Constructs and Associated Measures Extracted from Included Studies

Constructs	Associated Measures
Efficiency in problem solving	<i>Time to complete</i> (S01-A, S01-B, S01-C, S03, S04, S06, S07, S11, S12, S13, S14, S16, S18, S19, S21)
	<i>Learning efficiency / Ease of understanding and performing the task</i> (S01-A, S01-B, S01-C, S03, S06, S07, S08, S11, S13)
	<i>Pattern selection ratio</i> (S18)
Quality of solution	<i>Subject matter expert evaluation</i> (S01-A, S01-B, S01-C, S05, S06, S17)
	<i>As perceived by participants</i> (S19)
	<i>Adherence to design principles</i> (S09)
	<i>Atomic design features</i> (S21)
Correctness of solution	<i>Useful / working / accurate solution</i> (S01-B, S01-C, S05, S06, S09, S10, S20)
	<i>Number of errors</i> (S03, S04, S10)
	<i>Number of failing tests</i> (S14)
Completeness of solution	<i>Requirements fulfillment</i> (S01-B, S03, S06, S07, S09, S10, S12)
	<i>Recall and precision, as compared to a 'gold standard' solution</i> (S19)
	<i>Quantity of results</i> (S17)
Complexity of application	<i>Cognitive load, qualitatively assessed</i> (S01-A, S01-B, S01-C)
	<i>Eye focus, quantitative assessment of focus of attention during the task</i> (S08)
	<i>Cyclomatic complexity, or number of linearly-independent paths</i> (S15)
	<i>Ease of understanding the solution, qualitatively assessed</i> (S20)
Usability of patterns	<i>Pattern selection</i> (S02, S05, S13, S19)
	<i>Pattern application</i> (S02, S05, S09, S10, S12, S13)
	<i>Task satisfaction measure, qualitative assessment</i> (S21)
Communicability of pattern knowledge	<i>Oral communication</i> (S06, S11, S15)
	<i>Written Documentation</i> (S15, S20)
Creativity of solution	<i>Subject matter expert evaluation</i> (S06)
Modularity of solution	<i>Coupling</i> (S02)
	<i>Cohesion</i> (S02)
	<i>Weighted Methods per Class</i> (S02)
	<i>Architecture decomposition</i> (S20)
Size of solution	<i>Lines of Code</i> (S02)
	<i>Number of Classes</i> (S02)
	<i>Number of Operations</i> (S02)
	<i>Number of Attributes</i> (S02)

Efficiency is the most commonly used construct, measured in 14 studies, followed by correctness and completeness, measured in 9 studies each. Usability of patterns is explored in eight studies. In our analysis, the measures used for different constructs include both quantitative measures (e.g., time to complete a task) and qualitative measures (e.g., subject matter expert evaluation). Also, the measures are used to evaluate both the final artifact produced as an outcome of the study and the process involved in producing the artifacts.

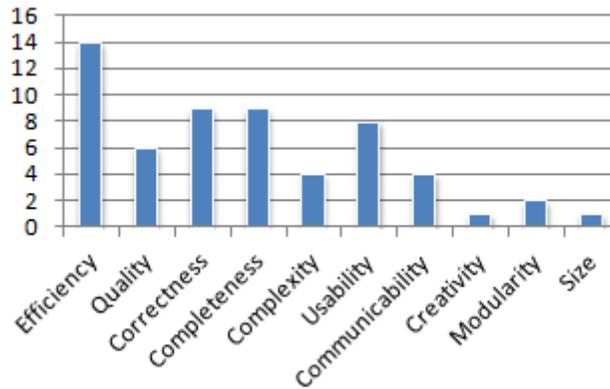


Figure 5. Frequencies of Constructs Used in Primary Studies

#### 4.2.4 MS-RQ4: Threats to Validity of Included Studies

MS-RQ4: *What threats to validity related to the study design and execution have been considered by the researchers?*

This research question directs us to identify and categorize the threats to validity that have been considered and mitigated at various steps by the researchers. Threats to validity of empirical evaluation can be mitigated by case study or experiment design. In situations where designs cannot mitigate threats, those threats must be reported as potential limitations. Ten (10) of the included studies did not discuss any threats to validity. Twenty-one (21) studies discuss one or more of the four general types of threats to validity that will be discussed in the following subsections and categorized in Table 10. While there can be additional, unidentified threats to validity not considered or addressed by the authors, we are primarily synthesizing and reporting the threats that are explicitly considered and listed by the authors of original studies. Table 10 should not be considered as a complete list of all the potential threats to validity of the included studies. We have used the established categories for types of threats to validity as listed by Wohlin et al. (Wohlin et al. 2000) whenever applicable.

Table 10. Threats to Validity of Included Studies

Validity Type	Threat Type
<b>Internal Validity</b>	Selection (S03, S04, S09, S10, S04-R1, S11, S17, S18, S19, S20, S21)
	Mortality, or participant attrition (S03, S04, S14, S04-R3)
	Testing and training (S19)
	Diffusion or imitation of treatments (S08, S09, S10, S14, S16, S17, S18, S04-R2, S04-R4)
	Resentful demoralization (S03, S04-R1, S16)
	Instrumentation (S04, S04-R1, S04-R2, S04-R3, S04-R5, S08, S16)
	Maturation, or learning and fatigue effects (S07, S08, S11, S14, S04-R4)
	Interactions with selection (S04-R1, S21)
	Presence of un-identified patterns (S14, S16, S17)
	Compensatory equalization of treatments (S14, S16, S18, S04-R5)
<b>Construct Validity</b>	Mono-operation bias (S08, S17)
	Mono-method bias (S08, S17)
	Hypothesis guessing (S04-R5, S08, S19)
	Evaluation apprehension (S04-R1, S04-R4, S08, S16)
	Observable measures used (S04-R4, S04-R5, S17, S21)
<b>External Validity</b>	Representativeness of training and preparatory material (S02, S04-R5, S18)
	Representativeness of sample population (S02, S03, S04, S04-R2, S08, S09, S11, S16, S17, S18, S19, S20)
	Uncontrolled task selection (S02)
	Experimental constraints that limit realism (S03, S04, S04-R1, S11, S17, S19, S20, S21)
	Task generalizability (S03, S04, S04-R1, S04-R2, S07, S10, S11, S14, S16, S18, S19)
	Task representativeness (S03, S04, S07, S08, S09, S10, S14, S18)
	Pattern representativeness (S04, S10, S14)
	Tool and language representativeness (S04-R1, S14, S16)
<b>Conclusion Validity</b>	Random irrelevancies in experimental setting (S04-R5, S08)
	Insufficient number of participants or data points (S03, S08, S14)
	Low statistical power (S09, S14)
	Reliability of measures (S03, S04-R5, S16, S17, S19, S20)
	Violated assumptions of statistical tests (S17, S20)

*Internal Validity:* The two most commonly reported threats to internal validity, discussed in 11 and 9 studies respectively, are: selection bias due to variation in expertise among participants and treatment diffusion. Selection bias due to variation in expertise is mitigated by employing counter-balancing or random block assignment. Counter-balancing reduces the effect that a specific factor (e.g., expertise, tasks) may have on a group’s measures. For instance, study S03 employs counter-balancing to distribute demographic factors across the treatment and control groups. Study S04 uses random blocked assignment to explicitly balance the groups in terms of background and expertise. Other commonly considered threats

to internal validity include whether treatments have been equalized across groups (4 studies), learning or fatigue effects when performing multiple tasks in a sequence (5 studies), and potential errors in instrumentation such as accuracy in measurement of time (7 studies).

*Construct Validity:* Evaluation apprehension, a human tendency to try to look better or fear of being evaluated, is the most commonly identified threat to construct validity as reported in four studies. Participants may experience apprehensions during the experiment, such as fear of poor performance. These apprehensions can hinder participants' potential to perform to the best of their abilities whether they are based on actual or perceived liabilities associated with participation in the experiment. Measuring performance of participants in the presence of such unintended apprehensions can lead to an inaccurate assessment of participants' abilities, which can threaten construct validity. Blinding can minimize biases and apprehensions on part of the experimenter and participants (Barbara Kitchenham et al. 2008), leading to increased validity of the findings. In study S08, double blinding was used in which participants were not informed about the goal of the study and a level of indirection was used so that results could not be traced back to any specific participant.

*External Validity:* Two commonly identified threats to external validity, reported in 12 and 11 studies respectively, are representativeness of sample population and task generalizability. Generalizability of results is limited in the absence of explicit theories of software construction and maintenance. Generalizability is further constrained by the considerable variations among patterns. In 4 of the 30 studies, the research questions explicitly focus on teaching design skills to students, which is the participant population of interest. For the remaining studies, the research questions of these studies (as listed in Section

4.2.1) and the nature of tasks given to the participants (e.g., development tasks, maintenance tasks, tasks related to software architecture) indicate that the sample would ideally be drawn from a population of professional software engineers of varying expertise to improve the representativeness of sample population.

*Conclusion Validity:* Reliability of measures (such as subjective assessment of outcome) was the most commonly identified threat to conclusion validity, as reported in six of the studies. Another issue that threatens conclusion validity is the need to have a sufficient number of participants and data points from which to draw reliable and statistically significant conclusions. Moreover, appropriate type of analysis must be carried out based on the type of measurement scale used so that correct statistical inferences can be drawn. Using incorrect type of analysis can threaten conclusion validity, as discussed in S17.

#### **4.2.5 MS-RQ5: Replicated Studies**

*MS-RQ5: How do replicated studies of software pattern application compare to original studies in term of study design?*

This research question directs us to classify the replication studies and identify factors that may be conducive to replication. Two studies included in our analysis, S03 and S04, have been replicated one and five times respectively. We are interested in identifying how similar the replications are to the original studies in terms of study design. We classify the replication studies according the definitions established in the literature as shown in Table 11. All the replications are differentiated replications. Three replications only involve a subset of tasks as compared to the original study. In one case, the replication study examines

additional factors than the ones explored in the original study. In all other cases, replication studies explore the same factors.

Table 11. Classification of Replication Studies

Ref. Code	Classification	Details
S03-R1	Differentiated Replication	<ul style="list-style-type: none"> <li>• <i>Difference in empirical setting:</i> S03-R1 is a web-based activity whereas the original experiments (S03) were paper-based. Additional variations are related to participants' demographics and structuring of pattern-specific documentation used in the replication study. S03-R1 is also an independent replication as it is performed by a different set of researchers than those involved in the original study.</li> <li>• <i>Difference in participants' demographics:</i> Participants in replicated experiment are undergraduates whereas original experiment involved both undergraduate and graduate participants.</li> </ul>
S04-R1	Differentiated Replication	<ul style="list-style-type: none"> <li>• <i>Difference in empirical setting:</i> The setting of the replication study was closer to a real programming environment (tasks were carried out on a computer instead of paper; participants were selected from multiple consultancy companies; participants were provided monetary compensation for participation) as compared to the original study S04.</li> <li>• <i>Difference in participants' demographics:</i> Participants in both studies are professionals. However, expertise is self-reported in the original study whereas it is based on employer's rating in the replication.</li> </ul>
S04-R2, S04-R3, S04-R5	Differentiated (Partial) Replications	<ul style="list-style-type: none"> <li>• <i>Subset of tasks and factors:</i> The replications involve only a subset of tasks and consider a subset of factors (dropping out the effect of training) when compared to the original study S04.</li> <li>• <i>Difference in empirical setting:</i> Tasks were carried out on a computer instead of paper. Original study uses C++ programming language whereas replicated studies use C++, Java or C#.</li> <li>• <i>Difference in participants' demographics:</i> Participants in replicated experiments are graduate and undergraduate students whereas participants in the original study are professionals.</li> </ul>
S04-R4	Differentiated Replication	<ul style="list-style-type: none"> <li>• <i>Additional factors studied:</i> S04-R4 introduces an additional factor (availability of graphical design of the programs) which may have impacted the outcome.</li> <li>• <i>Difference in empirical setting:</i> Tasks were carried out on a computer instead of paper. Original study uses C++ programming language whereas replicated studies use C++, Java or C#.</li> <li>• <i>Difference in participants' demographics:</i> Participants in replicated experiments are graduate students whereas participants in the original study are professionals.</li> </ul>

In general, replication studies consider the same constructs as the original study however they may use an alternate or additional measure. For instance, S04-R4 defined its own scale for correctness for one of the experiments in the study but used the same scale for the replicated experiment. Of the six replication studies, only S04-R4 considered an additional factor in that participants were familiarized with the graphical design of the software before

carrying out the tasks. Findings of the replication of S03 are consistent with the original study that pattern documentation in code is useful in increasing the efficiency during maintenance activities. However, findings of replications of S04 differ from the original study and are inconclusive in establishing the usefulness of various design patterns.

Both the studies that have been replicated have a quality score of 3 (Appendix-B), indicating that the researchers have provided a detailed report of the original study. Of the 24 original studies, only 5 others have a score of 3 based on our quality assessment. We list factors that might be conducive for replication, as identified from the original and replication studies, in Table 12. Availability of either a replication package, including study materials, or involvement of one or more of the original researchers may also support the replication effort. Four replications of S04 (S04-R2 to S04-R5) are carried out as part of a joint replication effort. S04-R2 involves one of the original researchers who loosely collaborate with the rest of the researchers (S04-R3 to S04-R5). The purpose of a joint replication is to collect sufficient data points for joint evaluation and analysis. All the four replications are carried out using a replication web portal to conduct the experiment and collect results however a joint evaluation of the results has not been reported yet.

Table 12. Factors Conducive for Replication

Factors Conducive for Replication	-R1 S03	-R1 S04	-R2 S04	-R3 S04	-R4 S04	-R5 S04
Availability of a replication package	X					
Detailed reporting of original study	X	X	X	X	X	X
One or more of the original researchers involved in replication		X	X			
Joint replication effort			X	X	X	X

### **4.3 Synthesis and Recommendations**

We want to promote empirical evaluation of software patterns. Ability to replicate the studies and synthesize different outcomes is an important aspect of empiricism. For this, we need to understand and account for the variations in the context variables. Based on our analysis, we have identified considerations that can support replication and comparison of findings across studies as discussed in the following subsections.

#### **4.3.1 Identification of Suitable Participants**

In terms of participants' demographics, students often provide a sample of convenience for empirical software engineering research, as observed in our analysis as well. However, students may be a suitable demographics based on the study goals. For instance, eight studies explicitly relate to the theme of training using patterns as listed in Table 4. Novices, such as students or software professionals new to the problem domain, form a suitable demographic in these studies. Graduate students often have a fair amount of industry experience and can be representative of junior software practitioners. For instance, in S09, participants had an average industry experience of five years. In other cases, students may be substituted for professionals depending on the nature of the task. Host et al. (Höst and Regnell 2000) found only minor differences between students and professionals in carrying out various software engineering tasks requiring "general understanding of dependencies and relationships", such as project impact assessment. Students may be used in similar tasks in place of professionals. Carver et al. (J. C. Carver, Jaccheri, and Morasca 2010) have provided a checklist to consider when selecting students as study participants. They also highlight requirements related to ethics, validity, and generalizability of the study in accordance with the overall study goals

when using students as participants. Additional studies comparing students and professionals under various empirical settings can be conducted to understand similarities and differences among students and professionals.

Some studies require participants from multiple demographics. For instance, S01 compares novices versus experienced users in problem solving using patterns. In such cases, both demographic groups should be distinguishable and clear criteria for rating participants to determine expertise level should be established, as discussed in next section. Even when grouping of participants is not carried out based on experience level, we still need to rate participants on experience level to create balanced groups and to assess applicability of the findings (Kleinschmager and Hanenberg. 2011).

Investigators should also control for participant expertise to determine what effect the pattern use contributes to the quality of the result as either independent or in conjunction with experience level. However, the reliance on experts without a standard mechanism to measure expertise limits the comparability of results: 'expert' in one study might be considered at a 'medium' or 'novice' level in another study. In addition, several studies employed experts to evaluate the extent to which an artifact exhibits quality, correctness and creativity. However, to lend credibility to the evaluation, we again need to establish what makes these evaluators experts in the given problem domain (Hibshi et al. 2014).

*Establishing a standard measure for experience level and expertise of participants across studies can help in identifying the demographics to which the findings are applicable. However, experience is difficult to establish and is based on multiple factors including participants background as well as familiarity with the given problem domain, set of patterns, and set of tasks. As a start, studies should report all of the above factors in detail. Students may be a suitable demographics based on study goals, nature of tasks, and work experience of students. Variations among students in terms of work experience, knowledge of patterns, and knowledge of problem domain should also be factored in so that applicability and generalizability of the findings could be established.*

#### **4.3.2 Tasks and Training on Task**

Researchers have used different problem sets, tools and frameworks to perform the given task, which limits the synthesis of findings across the studies. Sim et al. (Sim, Easterbrook, and Holt. 2003) observe that benchmarks facilitate technological progress and community building within a research area. Researchers in a number of computer science fields, such as image processing, network traffic monitoring, and recently social network analysis, routinely use standard problem sets to compare and contrast the performance of different algorithms and solution methodologies. In software engineering Heckman et al. (Heckman and Williams. 2008) have established a benchmark for evaluating static analysis alert prioritization and classification techniques. Software patterns research can also benefit from such standardization. Standard problem sets could be published with accompanying training

materials, such as textual formats and example presentations of the patterns to be given to participants.

*Researchers use different problem sets, tools and frameworks to perform a given task, which limits the synthesis of findings across the studies. Subtle differences in the duration and kind of training may impact performance. To support comparative evaluation, investigators should develop a standard set of problems and training materials for the patterns of interest. Availability of a package containing problem sets, instructions, and training material used in a study, is found conducive for replication. Moreover, when evaluating the effect of different training material on study outcomes, the provision of training material should be followed by a comprehension test to see if training had any effect on knowledge acquisition.*

### **4.3.3 Study Themes and Evaluation Constructs**

Researchers have investigated various themes related to the use of software patterns and evaluated different constructs to determine the effect of pattern use on desirable outcomes. Based on the classification used earlier (see Table 4), we have identified how frequently each construct is used to evaluate the outcome for each theme in Figure 6. When evaluating the use of software patterns in maintenance, most frequently used constructs are efficiency, correctness and completeness respectively. These are also the most commonly used constructs for evaluation in general (Figure 5). Usability, in terms of identifying applicable patterns and correctly instantiating patterns in the context of the problem, is also used frequently as an evaluation criterion for studies related to the themes of software construction

and performance. As one would expect, communicability during problem solving is the most commonly used construct for studies belonging to the theme of communication. Only time when efficiency is not one of the most frequently used criteria is while evaluating the use of software patterns during software construction.

Based on our analysis, time to complete a task can serve as a suitable measure for efficiency. However, when time is self-reported by participants, the measurement might not be as reliable. If incomplete or erroneous solutions are present, we also need to account for time needed to complete the task or to correct the errors. Measures for efficiency, completeness and correctness can be used in conjunction to have a realistic assessment of participants' performance. 'Learning efficiency', as self-reported by participants is also used to measure efficiency. Although self-reported qualitative measures may not be reliable, they can provide useful insights into the problem solving process and can be used to generate hypotheses for further evaluation. In a number of studies, researchers used qualitative measures such as quality, communicability or usability as evaluated by subject matter experts. When using qualitative measures, minimizing subjectivity is an important consideration. Researchers should consider using multiple evaluators and report inter-rater agreement between evaluators. When oracles, checklists or 'gold standards' are used to evaluate the outcome, researchers should describe mechanisms used to create these artifacts to lend validity to the findings of the study (Yin 2009).

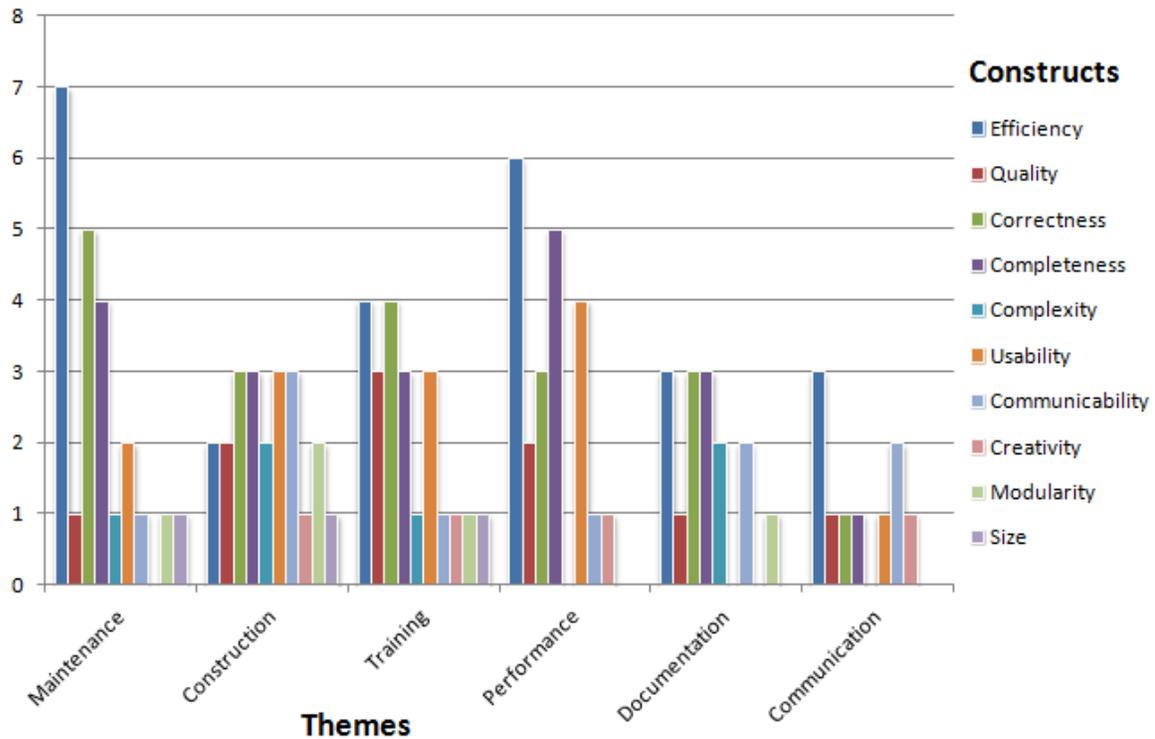


Figure 6. Frequencies of constructs used in the primary studies for each theme.

Often a construct that is being explored in the study is not well aligned with the measurement taken. For instance, studies S01 (A, B, C) assessed whether pattern use improves domain knowledge and design skills, whereas the variables measured primarily concern efficiency (time to complete, and ease of performing the task) and not knowledge acquisition. Because design skill is highly subjective and there are no established metrics, this is a reasonably difficult construct to measure and convenient proxy variables are easily mistaken for the construct. Such constructs may be qualitatively evaluated by subject matter experts. Research is ongoing to understand the impact of software patterns on various quality attributes (Galster and Avgeriou 2012). In general, software engineering investigators need to conduct more fundamental research on which measures correspond to which independent

and dependent variables and under what conditions those measures are both reliable and valid.

*Measures for efficiency, completeness and correctness can be used in conjunction to have a realistic assessment of participants' performance. Self-reported qualitative measures may not be reliable but can provide useful insights into the problem solving process. For constructs that are difficult to measure and have no established metrics, such as design quality, multiple subject matter experts may be employed to evaluate the outcome, preferably using pre-specified quality checklists and reporting inter-rater*

#### **4.4 Threats to Validity of Our Mapping Study**

We now discuss the threats to validity of our mapping study.

*Selection of search terms and digital libraries:* Selection of search terms and digital libraries can preclude some relevant studies from appearing in our search results. We searched four digital libraries and two journals that tend to cover high quality publications in software engineering. We have also looked at the references of 57 papers, shortlisted after exclusion based on title and abstracts, to identify additional studies that we might have missed during primary search. However, studies published in proceedings not indexed by the four digital libraries may not appear in our analysis.

*Selection of studies:* We are interested in empirical evaluation of software pattern application involving human participants. Our selection criteria would exclude studies that consider a software artifact in isolation or focus on structural or quality aspect of patterns without involving pattern application. We would also exclude surveys where participants

provide their opinion about software patterns without using them in problem solving in an empirical setting. Our findings should be considered applicable only in the context of the selected studies. We have also defined a criterion for quality assessment of included studies based on the aims and research questions of our study. We did not exclude any study based on the quality assessment results however the quality of included studies is fairly uniform.

*Reliability of data extraction and categorization:* We extracted and categorized data from the included studies to answer our research questions. We have identified each category by carrying out a constant comparison of instances of data across all studies in accordance with grounded analysis practices. We report all identified categories without omission in our analysis. The extraction and categorization process was carried out by the first author, a graduate student with over five years of work experience in software engineering and design patterns. The first two co-authors provided input to resolve ambiguities during the process. In this respect, the extraction and categorization process is partially validated.

## **4.5 Discussion**

We have reported the results of our systematic mapping study documenting and analyzing the current state of empirical research on the use of software patterns in problem solving. While analyzing structural aspects of patterns in isolation can provide insights into the inherent properties of a pattern, use of patterns during problem solving involves additional context factors such as participants, problems, domains and time constraints. A pattern that has good quality attributes may not be useful during problem solving if not instantiated properly in the context of the problem being solved (Jalil and Noah. 2007). Characterization of existing empirical research, evaluating the use of software patterns by

human participants, is a first step towards understanding the dynamics of pattern application in problem solving.

We selected 30 primary studies that evaluate design patterns, software architecture patterns and security patterns, among others. Five of the included studies report case studies while the remaining are experiments. We classified the primary studies in terms of their research questions, empirical design context factors, measures, and threats to validity. Despite the availability of guidelines for conducting empirical studies in software engineering (Barbara Kitchenham et al. 2008), we found gaps in adherence to the guidelines. For instance, when using qualitative measures, no detail on minimizing biases and subjectivity in assessment is provided 27% of the time. Moreover, 10 of the primary studies do not discuss any threats to validity of the study design and execution. However, the quality score for studies published in recent years is consistently high indicating that the researchers are providing more details in terms of study design and execution.

We found that while evaluating the problem solving process, measures such as 'time to complete a task' and 'learning efficiency' are consistently used across studies. This might indicate that empirical evaluation of software patterns is converging towards a set of standard measures to evaluate how software patterns support problem solving process. However, we didn't identify measures that are consistently used to evaluate the final artifact itself even though majority of the measures are related to the artifact. Since we are only considering studies involving human participants, studies evaluating the final artifact that do not involve human participants are not included in our analysis. An analysis of such studies might lead to identification of commonly used measures for evaluating the final artifact. Different problem

domains, patterns and tasks might account for the use of different measures for evaluating the final artifact as well. Two-thirds of the primary studies have explicitly considered threats to validity of study design. While internal and external validity is frequently assessed, researchers are increasingly considering construct validity as well as validity of the conclusions drawn as a result of the study.

We identified that subtle differences in study design can significantly limit comparison of experimental results across studies. For instance, despite the common use of participant categories as either novices or experts, we found that we could not compare these categories among different studies. Each study uses a different criterion to measure expertise based on the problem domain and relative training in solving the given problem since expertise is not independent of the pattern and problem domains in such experiments. Observations based on experience level of participants, without establishing a standard definition for experience level, can be difficult to generalize. These issues can affect the ability to compare results across studies, which sample from different demographics. As software engineering continues to borrow case study and experimental design practices from other disciplines, such as sociology and psychology, more work is needed to adapt these methods to software engineering and streamline practices to the point that quantitative comparisons can be made. In addition, establishing baselines in terms of the experience level of participants, the problem sets participants work on, and the measures used to evaluate participants' performance can support replication and comparison of findings across studies.

The results of our mapping study highlight the current state of empirical research on software pattern application in terms of design, measurements and use in problem solving.

Based on our mapping study, we have identified commonly employed empirical protocols, the level of details in reporting various context factors, and commonly used measures. Future researchers can leverage this information when designing studies while considering various threats to validity as reported herein. By using common measures and protocols, researchers can design new studies, replicate existing studies and be able to compare the findings. However, we observed that a number of context factors limit comparability of the findings. We have discussed these aspects in the context of our mapping study. Further evaluation of software patterns in empirical settings is needed to explore and address these issues by emphasizing reliability, repeatability and comparability of the studies.

Findings from our mapping study have guided the design and execution of the empirical studies and replications that we have conducted to evaluate our security requirements patterns and framework.

## 5 DIGS – A FRAMEWORK FOR DISCOVERING GOALS FOR SECURITY

### REQUIREMENTS ENGINEERING

In this chapter, we address the following research question:

- *RQ1: What is the core set of security properties that can be used to classify security goals and requirements of a system?*

We have developed DIGS, a framework for systematically Discovering Goals for Security (Riaz et al. 2016). The security properties identified as part of the DIGS framework are used for categorizing sentences in the input artifacts when applying SRD process. The security properties are also used for structuring the security requirements patterns in the repository used for specifying security requirements during SRD process. In addition to supporting the discovery of security goals, DIGS provide an underlying theoretical framework that connects the different elements of the SRD process.

The functional requirements and a list of the assets of a software system are input to the DIGS framework, and security goals associated with the initial, or any additionally identified, assets are its output. DIGS helps an analyst in systematic discovery of a system's security goals, such as goals related to confidentiality or accountability of assets, for different security actions (i.e., preventing, detecting, or responding to a breach) by codifying the pertinent knowledge as a set of security goal patterns (Section 5.2). When using DIGS, an analyst makes a conscious choice to select or not select a security goal for an asset, which can be reasoned about and documented, minimizing errors of omission. Analysts may also revisit and identify goals not considered in an earlier analysis. DIGS supports discovery of security goals and helps organize security goals related to the assets. This organization helps

in quickly identifying areas where goals have not been specified and that may need additional security fortification. Moreover, we map the security goal patterns to candidate security mechanisms that can also help in operationalizing the goals. We evaluate DIGS in identifying implied security goals via a controlled experiment (Section 5.5).

## 5.1 Elements of DIGS Framework

In Figure 7, we provide an overview of the elements of DIGS framework and their relations. Each element is described in the following subsections.

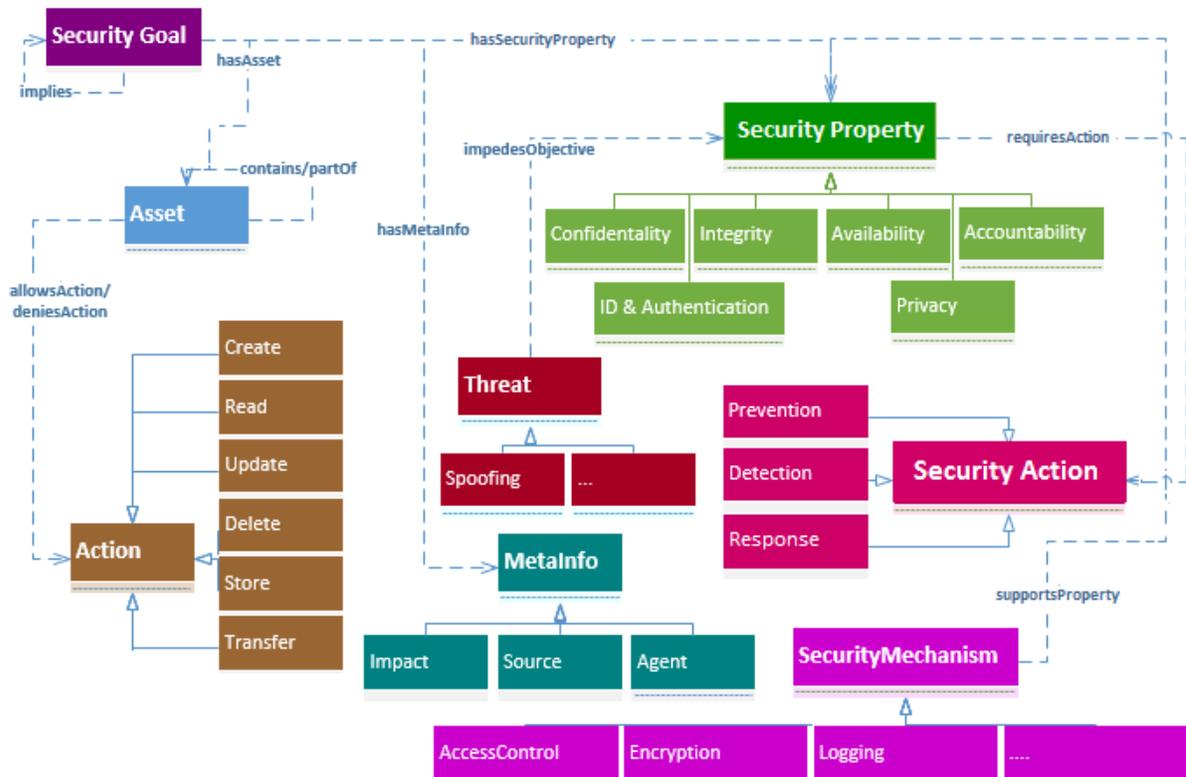


Figure 7. Elements of DIGS Framework

### 5.1.1 Assets

DIGS accepts as input a list of assets used and controlled by the software system, along with the functional requirements. The assets can be provided separately (if using DIGS in

isolation) or identified from the sentences that imply one or more security properties in the natural language requirements artifacts (if using as part of SRD). The assets of the software system are its sensitive resources and services, such as patient's health record, that need to be protected. Assets can be mutually related, for instance, one asset can be composed of other assets. Assets can be ranked in terms of their security risks. The security risk of an asset is related both to how valuable an asset is and how easily the asset can be attacked. Relative security risks for assets can be computed using existing techniques, such as protection poker (Williams, Meneely, and Shipley 2010). In certain domains, such as military and healthcare, assets are assigned to predefined classes, mostly concerning with different levels of confidentiality and privacy associated with the asset. We do not assume any specific classification for the assets, and ranking assets by security risks is optional. However, if the ranking is available, it can guide the selection of appropriate security mechanisms for operationalizing the goals.

### **5.1.2 Actions**

For each information asset, we have identified a list of action categories that can be permitted or prohibited, as shown in Figure 7. We can either explicitly specify prohibited actions, or use the closed-world assumption<sup>6</sup> where any action that is not explicitly permitted on an asset is prohibited. In addition to the standard CRUD (*create, read, update, delete*) actions, we add two additional action categories: (1) storage; and (2) transfer of information,

---

<sup>6</sup> presumption that a statement that is true is also known to be true

based on our previous analysis of over 11,000 requirements sentences (Riaz, King, et al. 2014). We define these two additional action categories as follows:

- *store*: actions related to storage and backup of the assets at rest, e.g., backing up log files.
- *transfer*: actions related to transfer or sharing of the assets, e.g., sending patient health record from one service to another.

These six actions help us consider security of assets starting from the creation of an asset, through usage, storage, or transfer of assets, until the asset expires.

### **5.1.3 Security Properties**

We model the security goals of a system in terms of the assets that need to be protected and the security properties we want to have for the assets, as shown in Figure 7. For instance, a security goal can be to ensure ‘confidentiality of a patient record’ where ‘confidentiality’ is the property and ‘patient record’ is the asset. By identifying the security properties expressed or implied by a particular sentence within a document, we gain an understanding of the intent of the sentence as well as possible requirements and mechanisms to establish that intent.

While certain sets of security properties are widely known such as “Confidentiality, Integrity, and Availability (CIA) Triad”, we want to ensure the completeness of the set of security properties. To synthesize a list of core security properties of software systems, we systematically examined (1) six security standards [(“Common Criteria for Information Technology Security Evaluation, Version 3.1. Release 4” 2012), (“Special Publication 800-53 Revision 4 - Security and Privacy Controls for Federal Information Systems and Organizations” 2013), (“Underlying Technical Models for Information Technology Security”

2001), (“Standards for Security Categorization of Federal Information and Information Systems” 2004), (“Minimum Security Requirements for Federal Information and Information Systems ” 2006), (“Federal Information Security Management Act” 2002)]; (2) two taxonomies of security properties and requirements [(Firesmith 2004), (Lamsweerde 2004)]; and (3) two security seminal papers and books [(Schumacher et al. 2006), (Saltzer and Schroeder 1974)]. Based upon this examination, we define each of the resulting security property in Table 13 below. Each security property counters a specific threat in the Microsoft STRIDE<sup>7</sup> threat model.

Table 13. Core Set of Security Properties

<b>Security Property</b>	<b>Description</b>	<b>Counters STRIDE Threat</b>
Confidentiality (C)	The degree to which the "data is disclosed only as intended". (Schumacher et al. 2006)	Information Disclosure
Integrity (I)	The degree to which a system or component guards against improper modification or destruction of computer programs or data. (“Standards for Security Categorization of Federal Information and Information Systems” 2004)	Tampering Elevation of Privileges
Availability (A)	"The degree to which a system or component is operational and accessible when required for use." (“IEEE Standard Glossary of Software Engineering Terminology” 1990)	Denial of Service
Identification & Authentication (ID or IA)	The need to establish that "a claimed identity is valid" for a user, process or device. (“Underlying Technical Models for Information Technology Security” 2001)	Spoofing Elevation of Privileges
Accountability (AY)	The degree to which actions affecting software assets "can be traced to the actor responsible for the action." (Schumacher et al. 2006)	Repudiation
Privacy (PR)	The degree to which "an actor can understand and control how their information is used." (Riaz, King, et al. 2014)	Information Disclosure

<sup>7</sup> <https://msdn.microsoft.com/en-us/magazine/cc163519.aspx>

The actions on the various assets suggest applicable security properties.

- Confidentiality is important when performing ‘read’, ‘store’ and ‘transfer’ actions.
- Integrity is important when performing ‘create, update, delete’ and ‘transfer’ actions.
- Availability is important when performing all six action types. For 'Availability', the asset will be a service or a system functionality.
- Identification & Authentication is important when a system is accessed, prior to performing any of the six action types. If some actions are allowed without authentication, they should be explicitly specified.
- Accountability is important when performing any of the six action types.
- Privacy is important if the owner of information can exercise control over who can access the information during the actions ‘read’, ‘store’, and ‘transfer’

We use the actions that are performed on the assets in the system as a guide to indicate when various security properties should be considered for the assets. We validated our classification of security properties and consolidated the guidelines for identifying the six properties through an analysis of over 11,000 requirements sentences from healthcare domain (Riaz, King, et al. 2014).

#### **5.1.4 Security Actions**

Proactively preventing a security breach is the ideal scenario. However, security breaches do occur. In case of a breach, we can know that a breach has occurred and take remedial actions only if the goals related to detecting and responding to a breach have been incorporated in the system (LaPiedra 2002). Consider the security goal to ensure the confidentiality of a patient’s health record. In case of a security breach, we should detect and

respond to the breach as well. The goal of confidentiality of patient's health record can thus be refined into three goals: prevent breach of confidentiality; detect any breach of confidentiality; and respond to each breach of confidentiality. We capture these refinements in our framework, as shown in Figure 7, and define the following three main security action types:

- *Prevent (p)*: proactively prevent a security breach (Schumacher et al. 2006).
- *Detect (d)*: in case of a security breach, detect the breach (Schumacher et al. 2006).
- *Respond (r)*: in case of a security breach, respond to the detected breach (Schumacher et al. 2006)(Firesmith 2004).

Considering each of these security actions during requirements engineering can help in discovering a more comprehensive set of security goals. Moreover, in certain exceptional situations, we may allow access to assets, such as 'patient health record', and later reason about and detect if the access was in accordance with the privacy guidelines or whether a breach has occurred.

### **5.1.5 Security Mechanisms**

A security mechanism is a method, tool, procedure, or control put in place for operationalizing one or more security goals. Different mechanisms may be selected to support a security goal depending on the security action and asset's risk assessment. For instance, access control and encryption are two security mechanisms that support preventing a breach of confidentiality, whereas auditing mechanisms support the detection of a security breach. Similarly, for high-risk assets, we may employ additional mechanisms than for low-risk assets. NIST Special Publication 800-53 ("Special Publication 800-53 Revision 4 -

Security and Privacy Controls for Federal Information Systems and Organizations” 2013), specifies a list of security and privacy controls for information systems. NIST categorizes controls in terms of different families such as access control (AC), audit and accountability (AU), identification and authentication (IA), media protection (MP). NIST also provides information about priority and usage of control based on the impact of a security breach. We have additionally mapped NIST controls to the security goals (i.e., the security properties that the control supports and the security actions in which it is applicable). For instance, in the IA-family of NIST controls, IA-2, IA-3, and IA-9 map to preventing a breach of identification and authentication of actors (users, devices and services respectively). Controls IA-5 and IA-6 map to the implied goals (see Section 5.3) of preventing breaches of confidentiality and integrity of the authentication mechanism itself. Controls IA-10 and IA-11 map to both preventing as well as responding to a breach of authentication mechanism. The complete mapping is available on our project website<sup>8</sup> and provides guidance toward applicable controls based on the identified security goals. We have also developed security requirements patterns, based on NIST controls, for each security goal by abstracting and grouping related controls that support the same security goals (Section 6.3).

## **5.2 Security Goal Patterns**

Identifying security goals for a system is one of the initial steps during security requirements engineering (Mead, Houg, and Stehney 2005). To support the analysis of security of assets across multiple dimension, we have identified 18 patterns of security goals

---

<sup>8</sup> <https://sites.google.com/site/digsstudy/>

that cover all combinations of the 6 security properties and 3 security actions discussed earlier. The template given in Figure 8 can be used to generate the 18 security goal patterns that we have identified.

<prevent | detect | respond to>  
*a breach of*  
 <Confidentiality | Integrity | Availability | Identification & Authentication |  
 Accountability | Privacy>  
*of <asset>*  
 [*when <actor> <performs action>*]

Figure 8. Template for Security Goals

To abbreviate, each pattern is assigned a unique identifier as follows:

<p | d | r>-<C | I | A | ID | AY | PR>,

e.g., d-PR means ‘detect a breach of Privacy’.

In Table 14, we summarize the goal patterns and list the actions that indicate when different security properties should be considered for specifying security goals. For example, <read | store | transfer> type actions indicate a need for Confidentiality.

We can specify security goals for key system assets using the security goal patterns. The actions performed on the assets guide the choice of the applicable security properties. For instance, while reading the asset ‘patient health record’, we consider security properties of confidentiality, accountability, and privacy of health records as well as availability of system functionality to allow the read action. For each security property, we also consider goals related to all three security actions.

When using as part of the SRD process for identifying security goals, we do not need to provide an explicit list of assets for the system. The assets and corresponding security

properties can be identified from sentences that imply one or more security properties in the input artifacts.

Table 14. Security goal patterns

Security Action	Security Property	Asset	Actor	Action type (if applicable)
<b>&lt;prevent (p)   detect (d)   respond to (r)&gt;</b> <i>a breach of</i>	<b>Confidentiality (C)</b>	of <asset>	when <actor> performs	<read   store   transfer>
	<b>Integrity (I)</b>			<create   update   delete   transfer>
	<b>Availability* (A)</b>			<create   read   update   delete   store   transfer>
	<b>Id &amp; Authentication ** (ID)</b>			
	<b>Accountability (AY)</b>			
<b>Privacy (PR)</b>	<read   store   transfer>			

\* Asset will be a service or system functionality.

\*\* Applicable prior to any system access by default. Explicitly indicate the actions or assets that do not need authentication, if any.

We provide a set of example security goals that are identified for a patient’s health record in Figure 9. Goal A corresponds with preventing a breach of confidentiality. We can select appropriate security mechanisms, such as NIST control AC-3 for access enforcement, to operationalize this goal. Goal B is related to detecting a breach of privacy. We can use control AU-12 related to audit generation, to operationalize the goal. Goal C is related to responding to a breach of accountability for all types of actions listed in Section 5.1.2. Information about the relative security risk of assets may also guide the selection of security mechanisms. Example goals shown in Figure 9 are generated using the patterns as follows:

- *Goal A:* prevent a breach of Confidentiality of patient health record when user reads the data (i.e., p-C)

- *Goal B*: detect a breach of Privacy of patient health record when user reads the data (i.e., d-PR)
- *Goal C*: respond to a breach of Accountability of patient health record (i.e., r-AY)

	A	B	C
Asset	• Patient health record	• Patient health record	• Patient health record
Action	• Read	• Read	• All
Property	• Confidentiality	• Privacy	• Accountability
Security Action	• Prevent a breach	• Detect a breach	• Respond to a breach
Applicable Control(s)	• Access Enforcements (AC-3)	• Audit Generation (AU-12)	• Respond to Audit Processing Failures (AU-5)

Figure 9. Security goals for ‘patient health record’.

To reduce complexity during analysis, we can group the assets that have the same security goals and risks by creating equivalence classes of assets. Moreover, some related assets might have similar goals and should be considered together during analysis:

- Asset A is a type of Asset B (e.g., password is a type of credential)
  - Goals for B are applicable to A
- Asset A contains Asset B (e.g., patient health record contains SSN)
  - Goals for A are applicable to B
- Asset A is a part of Asset B (e.g., SSN is a part of patient health record)
  - Goals for A are applicable to B

We may also identify related goals that are operationalized through similar mechanisms.

### 5.3 Implied Security Goal Patterns

Based on the initial set of security goals that are identified, other security goals might be applicable. For instance, we might create new assets (e.g., audit records in Figure 9) or incorporate new functionality in the system (e.g., access enforcement mechanisms in Figure 9) to meet the initially identified goals. Security of these new assets or functionality is implied for the overall security of the system. As an example, two security goals that are implied for the security of audit records are to prevent a breach of confidentiality and integrity of the audit records (Goals D and E), as shown in Figure 10.

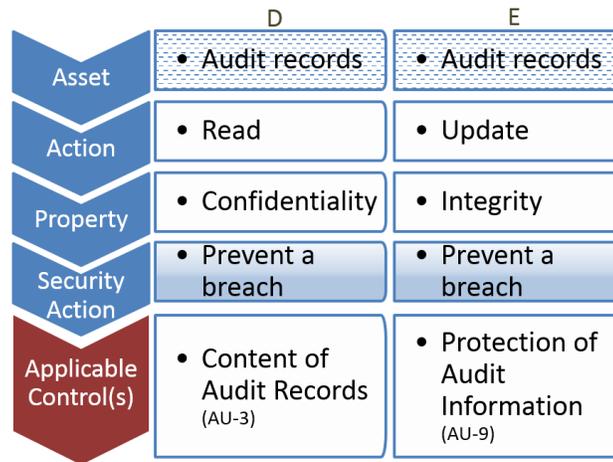


Figure 10. Implied security goals for ‘audit records’.

For each of the initial goal patterns, we explicitly capture the implied goals to consider. For instance, to prevent a breach of confidentiality of assets, an implied goal is the integrity of access enforcement mechanisms. Similarly, to detect a breach of confidentiality of assets, an implied goal is the integrity of audit records. To respond to a breach of confidentiality of assets, an implied goal is to have mechanisms in place to temporarily limit system availability. The list of implied security goals is given in Table 15. We also specify when an

implied goal indicates the need for new assets or security-related functionality to be added to the system. For instance, access enforcement mechanisms may employ login credentials, result in the creation of encrypted assets, or in the creation of security-related metadata (e.g., access control lists, security attributes). We can iteratively apply the 18 goal patterns for any newly created assets to have a comprehensive analysis of the security of system's assets.

Table 15. Implied security goals associated with 18 security goal patterns

Security Action	Security Property	Additional Security Actions to Consider for Security Goals
prevent a breach of	<Confidentiality   Integrity>	<b>Id &amp; Authentication</b> of actors <b>Availability</b> of <i>access enforcement mechanisms</i> (e.g. <i>authentication and cryptographic services</i> **) <b>Privacy</b> of assets
	Availability	<b>Availability</b> of <i>backup functionality</i> **
	Id & Authentication	<b>Confidentiality</b> of <i>identifiers and authenticators</i> * <b>Integrity</b> of <i>identifiers and authenticators</i> *
	Accountability	<b>Id &amp; Authentication</b> of actors <b>Availability</b> of <i>logging or monitoring services</i> ** <b>Integrity</b> of <i>audit records</i> *
	Privacy	<b>Confidentiality</b> of assets <b>Integrity</b> of assets, including <i>consent forms</i> *
detect a breach of	<Confidentiality   Integrity   Availability   Id & Authentication   Privacy>	<b>Accountability</b> of actions <b>Availability</b> of <i>logging or monitoring services</i> ** <b>Confidentiality</b> and <b>Integrity</b> of <i>audit records</i> *
	Accountability	<b>Availability</b> of <i>logging or monitoring services</i> **
respond to a breach of	<Confidentiality   Integrity   Privacy>	<b>Limit Availability</b> of system functionality <b>Availability</b> of <i>security assessment and response services</i> **
	<Id & Authentication   Accountability>	<b>Availability</b> of <i>alternate authentication and logging functionality</i> **
	Availability	<b>Availability</b> of <i>backup and restore functionality</i> ** <b>Confidentiality</b> and <b>Integrity</b> of <i>backup assets</i> * <b>Limit Availability</b> of system functionality

\* Consider adding this new asset and related functionality to the system if not already available.

\*\* Consider adding this new functionality to the system if not already available.

## 5.4 Steps for Applying DIGS

The functional requirements and assets of a software system are input to the DIGS framework and security goals associated with the initial, or additionally identified, assets are the output.

We outline the steps for applying DIGS for identifying security goals in Figure 11.

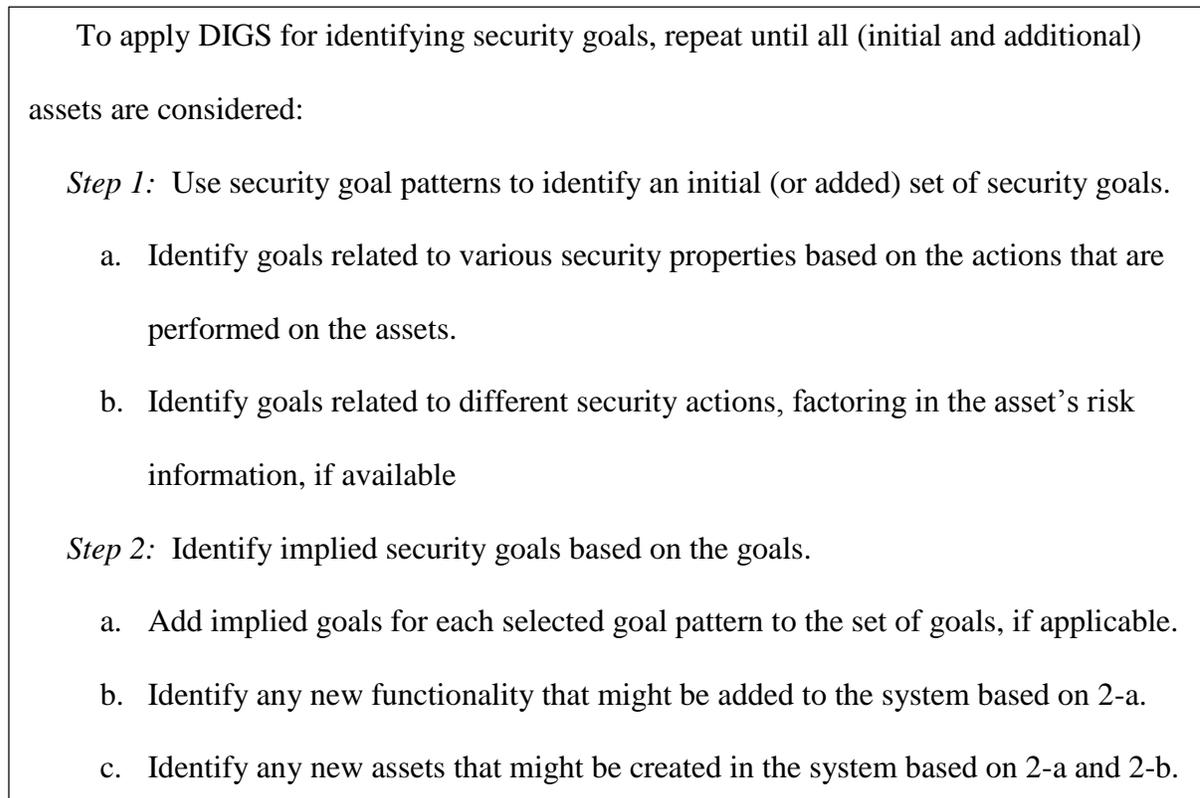


Figure 11. Steps for Applying DIGS

Consider the asset 'health record'. In Step 1, we identify all applicable security goal patterns for 'health record' by identifying the security properties (Step 1-a) and security actions (Step 1-b). When using DIGS as part of the SRD process, we already know the security properties (Step 1-a) for assets based on the classification of sentences containing those assets and we can directly start at Step 1-b.

In Step 2-a, we will look at the rows corresponding to all goal patterns selected in Step 1. For instance, for p-C (prevent a breach of confidentiality), we identify goal for availability of access enforcement mechanism in Step 2-a. Here, ‘access enforcement mechanism’ is the new functionality identified in Step 2-b. Similarly, ‘identifier and authenticators’ might be a potentially new type of asset related to access enforcement mechanism identified in Step 2-c. After identifying all the implied goals in this way, we go back to Step1 to identify any additional goals for the new assets. For ‘access enforcement mechanism’, we already identified goal for availability and now consider the remaining patterns to have a comprehensive analysis. The process will come to an end when no new functionality or asset is identified in Step 2-b and Step 2-c. In general, we expect the analyst to iterate through the steps no more than 2-3 times before saturation. Selecting all goal patterns or implied goals may not be feasible. Our objective is that an analyst be able to consider these goals and make conscious tradeoffs about including the respective security goals.

In addition to the discovery of security goals, DIGS supports the organization of the security goals by assets, security properties, and security actions. This organization helps in quickly assessing areas where goals have not been specified and that may need additional security fortification. Analysts may revisit and identify goals not considered in an earlier analysis. In this regard, DIGS may also be used for identifying potentially missing security requirements by mapping existing requirements to DIGS security goal patterns.

## **5.5 Empirical Evaluation Methodology**

We evaluate DIGS in identifying implied security goals via a controlled experiment where 28 participants analyzed two real-world systems from mobile banking and human

resource management domains. We report our methodology for conducting the experiment, as adopted from Jedlitschka et al. (Jedlitschka, Ciolkowski, and Pfahl 2008), in this section. The artifacts used during the study include training material given prior to the study, reference material available during the study, and the forms to submit the task responses. The experiment artifacts and task details are available online<sup>9</sup>.

### 5.5.1 Goals, Hypotheses, and Metrics

We conducted this experiment to evaluate whether DIGS supports the systematic and thorough discovery of security goals for a system. We analyze the initial set of security goals that participants identify using the 18 security goal patterns as well as any implied security goals identified based on the initial security goals. We did not control for knowledge of security goal patterns (Section 5.2) as both control and treatment groups were already familiarized with these security goal patterns.

The factors of interest that we controlled for are:

- Support of a systematic process for identifying the security goals using the DIGS framework (Section 5.4).
- Explicit knowledge of implied security goals (Section 5.3).

We explore the following null hypotheses:

- $H_{01}$ : Support of a systematic process does not impact a participant's ability to identify *different types of security goals using security goal patterns*.

---

<sup>9</sup> <https://sites.google.com/site/digsstudy/>

- $H_{02}$ : Explicit knowledge about *implied security goals* does not impact a participant's ability to identify such goals.

In Table 16, we list the metrics used for testing each hypothesis. We compute the metrics for each participant's response based on an oracle of security goals (see Section 5.5.5) developed a priori to the evaluation. For each response, we count goals as follows:

- True Positive (TP): A security goal identified by participant that is in the oracle.
- False Positive (FP): A security goal identified by participant that is not in the oracle.
- True Negative (TN): A security goal not identified by participant that is not in the oracle.
- False Negative (FN): A security goal not identified by the participant that is in the oracle.

Hypothesis  $H_{01}$  considers differences due to the systematic process used by treatment group given the same knowledge about security goal patterns as control. We test  $H_{01}$  using the metrics of precision and recall of the identified security goals. Additionally, we consider recall of security goals related to each security action to see if differences were consistent across the action types.

Hypothesis  $H_{02}$  is based on the differences due to the knowledge of implied goals available to treatment group. We test  $H_{02}$  by evaluating the recall of security goals identified initially versus the recall of implied goals identified from the discovered initial goals.

Table 16. Metrics used for evaluation of DIGS

<b>H<sub>01</sub></b>	Precision of security goals identified by individual participants. [TP / (TP + FP)]
	Recall of security goals identified by individual participants. [TP / (TP + FN)]
	Recall of security goals identified by individual participants, grouped by <i>security actions (i.e., prevention, detection, response)</i> .
<b>H<sub>02</sub></b>	Recall of security goals identified by individual participants, grouped by <i>discovery actions (i.e., initial or implied)</i> .

### 5.5.2 Participants

Our study participants were graduate students enrolled in a 16-week Computer and Network Security graduate course (CSC 574, Spring 2016) offered at NCSU. All the students received coursework credit for completing the task, similar to other class exercises. However, students could opt out of participating in the study<sup>10</sup>, which would preclude the inclusion of their work in the study results. Of the 29 students present for the lecture, 28 gave consent to participate in the study. We assigned participants to treatment and control groups based on a pre-task quiz (Section 4.3). Each group had 14 participants.

Each participant was assigned a unique random access code to use throughout the tasks so we can link participants' responses across all the tasks they performed. However, we recorded no personally identifiable information about the participants. At the end of the task, participants filled out a post-task questionnaire to document their academic and work experience in computer science and security. Participants in both groups had an average of

---

<sup>10</sup> The study was approved by the NCSU IRB (6548) and HRPO.

five years' experience in computer science. Participants had around one year of academic experience related to computer security, on average.

### 5.5.3 Experimental Design

The study consisted of three parts: a pre-task quiz, the main task, and a post-task questionnaire.

*Pre-task quiz* contained 15 multiple choice questions to assess the background knowledge of participants related to security goals based on the provided training material. Participants had 10 minutes to complete the quiz. Once the participants submitted the pre-task quiz (via a Google form), we automatically evaluated the responses. We assigned the responses into three terciles. We randomly assigned half of the participants from each tercile to the treatment and half to the control groups. The average pretask quiz scores for the control and treatment groups were 12.2 and 11.6, respectively, out of 15 points. As a result, we assume that neither group was inherently better at identifying security goals prior to the experiment.

*Main task* consisted of identifying security goals for two software systems given the system description and key assets, analyzing both systems in randomized order. For the main task, we provided a high-level description of a subset of features for the following two systems to each participant for analysis:

- iHRIS Manage<sup>11</sup> supports the Ministry of Health and other service delivery organizations to track, manage, deploy, and map the health workforce.

---

<sup>11</sup><http://www.ihris.org/ihris-suite/health-workforce-software/ihris-manage/>

- Cyclos, SMS banking module<sup>12</sup>, part of a secure and scalable payment software.

We selected these software systems as they manage diverse sets of assets. Operations on these assets cover all action types and require consideration for different security goals, thus allowing for a detailed analysis of the DIGS framework. Moreover, a description of key system features is also available online.

Each participant analyzed both systems, in random order, to identify security goals for each system. Both groups already had knowledge of security goal patterns, as presented in Table 1. Additionally, participants in the treatment group had knowledge of implied security goals based on the initial goals. All of the participants were asked to identify the security goals for the system. Participants in both the treatment and control groups were encouraged that once they have identified an *initial* set of security goals, they should try to identify any *implied* goals based on the discovered initial goals. Differences between the groups, in terms of available knowledge, are summarized in Table 17.

Table 17. Differences between Control and Treatment – DIGS Evaluation

	<b>Control</b>	<b>Treatment</b>
<b>Knowledge</b>	Security goal patterns (Section 5.2)	Security goal patterns (Section 5.2) + Implied security goals (Section 5.3)
<b>Process</b>	No specific methodology but suggestion to factor in provided knowledge	Steps outlined for applying DIGS (Section 5.4)

Participants were given 50 minutes to complete both tasks and allocated as much of the time as they wanted for each system. Treatment group participants additionally had to allocate time to understand the systematic process and implied goals given as part of

<sup>12</sup><http://www.cyclos.org/mobilebanking/>

reference material during that time. One participant, in the control group, only provided responses for Cyclos. All other participants analyzed both systems.

*Post-task questionnaire*, given at the conclusion of the main task, consisted of questions related to background experience of participants, as well as self-assessment on whether the tasks and methodology to complete the tasks were clear. The purpose of the latter is to determine whether students in the treatment group were able to understand the instructions given to them. This information could potentially explain the evidence or lack of statistical differences between the two task groups. For example, results may indicate no differences between the task groups because the treatment group did not understand the instructions.

#### 5.5.4 Experimental Analysis Methodology

The experimental design followed that of an analysis of variance (ANOVA) of a split-plot design (Montgomery 2012) where the whole-plot factor was the task group (control or treatment) and the split-plot factor was the system (iHRIS and Cyclos). We used the restricted maximum likelihood (REML) procedure (Harville 1977) to fit the statistical model:

$$y_{ijk} = \mu + \alpha_i + e(W)_{ij} + \beta_k + (\alpha\beta)_{ik} + e(S)_{ijk}$$

where  $y_{ijk}$  is the response of interest,  $\mu$  represents the grand mean,  $\alpha_i$  represents the effect of the  $i$ -th task group,  $\beta_k$  represents the effect of the  $k$ -th system,  $(\alpha\beta)_{ik}$  represents the interaction effect between the task and system group,  $e(W)_{ij}$  represents the whole-plot error, and  $e(S)_{ijk}$  represents the split-plot error. Both error terms are independent, normally distributed random variables with zero mean and variances of  $\sigma_W^2$  and  $\sigma_S^2$ , respectively, and are mutually independent. Testing for task group differences had fewer residual degrees-of-

freedom and so had less power than testing for system differences or task-system interactions. The effect size of any significant differences involving the task groups was further investigated using pairwise differences. All analyses were performed in JMP Pro 12<sup>13</sup>.

### **5.5.5 Oracle of Security Goals**

Prior to the evaluation, two of the authors created an oracle of all identifiable security goals for each asset in both the systems used in the study (iHRIS and Cyclos). The researchers involved in the creation of the oracle have 5 and 15 years of relevant experience. As a first step, both researchers individually voted ‘YES’ or ‘NO’ for each security goal that can be assigned to an asset based on the 18 patterns. The researchers had substantial agreement at the end of the individual voting based on the Cohen’s Kappa score (0.754 for iHRIS; 0.87 for Cyclos; 0.814 overall). The 22 disagreements, out of 252 possible votes, were resolved with discussion where the person voting in favor was able to convince the other person to include the goal in the oracle. The end result was a consolidated oracle, where any goal voted ‘YES’ by either of the researchers was included in the oracle. The iHRIS and Cyclos systems had 144 and 108 total possible security goals, respectively, of which 108 and 61 were applicable to the system (voted ‘YES’). Given the large number of goals in the oracle for the allocated time (almost 3-4 goals to identify per minute), we do not expect to see high recall values. Each goal in the oracle was categorized as follows:

- Security action: a) prevention; b) detection; or c) response.

---

<sup>13</sup>[http://www.jmp.com/en\\_us/software/jmp-pro.html](http://www.jmp.com/en_us/software/jmp-pro.html)

- Discovery phase: a) identified based on the initial analysis of system assets using goal patterns (Initial goals); or b) identified based on initial goals (Implied goals).

When analyzing participants' responses, we examined whether different categories of goals were identified by the participants, blind to the group each participant belonged to. During the analysis, we did not find any goal not in the oracle already.

## **5.6 Results and Analysis**

We evaluated the participants' responses and present the results in this section.

### **5.6.1 H<sub>01</sub>: Security Goal Patterns**

As shown in Table 17, both the control and the treatment groups had knowledge of the security goal patterns. We wanted to evaluate if both groups can identify security goals based on the provided goal patterns with or without the DIGS process.

We begin by investigating any significant differences based on the metrics of precision and recall of security goals between control and treatment using the ANOVA test (Section 5.5.4). All tests have a numerator degree-of-freedom of 1 and denominator degrees-of-freedom of approximately 26, based on the Kenward-Roger method, which is the default for JMP. Figure 12 shows plots of the means and standard errors of the responses by task group and system. An increasing slope in the lines indicates superior performance by the treatment group. In nearly all cases, we see the treatment means are higher than the control means. However, the standard errors are large.

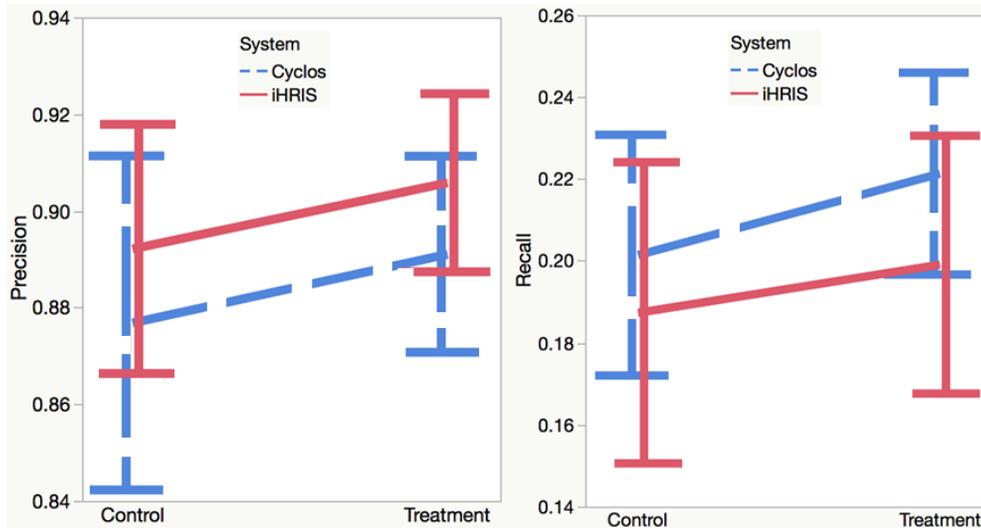


Figure 12. Means and standard errors of Precision and Recall by Task Group and System – DIGS Evaluation.

The differences between the performance of the treatment and control groups are not statistically significant, however, as shown in Table 18. On average, both groups had high precision (control: 0.88; treatment: 0.9) but low recall (control: 0.18; treatment; 0.21) since participants are more likely to have false negatives than false positives. Low recall can be explained by a number of factors including the limited time and resources available to participants, and the fact that missing a security goal would have no real consequences for the participants in an empirical setting. Participants spent between 15 and 19 minutes on each system on average identifying around 1 security goal per minute on average (control: 0.89; treatment: 1.24). The time spent on identifying security goals would likely be significantly greater in real life where the stakes are much higher. It is also likely that in practice a team of analysts and stakeholders would identify the system’s security goals rather than an individual (see Section 5.6.3).

Table 18. Analysis of variance for Precision and Recall – DIGS Evaluation.

Effect	Precision		Recall	
	F-value	P-value	F-value	P-value
<b>Task</b>	0.314	0.580	0.168	0.685
<b>System</b>	0.329	0.571	0.579	0.454
<b>Task* System</b>	0.000	0.989	0.045	0.834

Next, we analyzed the recall of security goals for each of the three security actions to see if participants considered different goal patterns. The analysis of standardized responses for the correctly identified prevention, detection, and response security goals revealed no significant differences between either the task groups or systems. Moreover, we did not find a significant difference in the recall values for goals related to each security action in either group. Table 19 and Figure 13 show the results.

Table 19. Analysis of variance for standardized responses for the correctly identified Prevention, Detection, and Response security goals.

Effect	Prevention		Detection		Response	
	F-val	P-val	F-val	P-val	F-val	P-val
<b>Task</b>	0.659	0.424	0.094	0.762	0.100	0.754
<b>System</b>	0.097	0.759	0.002	0.963	2.848	0.104
<b>Task* System</b>	0.389	0.538	0.850	0.365	0.000	0.990

We analyzed qualitative feedback from participants to see if the treatment and control groups had different perceptions about the task. In the treatment group, 11 of the 14 participants had a positive sentiment about performing the task while 3 participants felt unsure. In contrast, only 4 participants in the control group had a positive sentiment about performing the task, 6 felt unsure and 4 did not respond. Our results indicate that the treatment group was more confident in how the task should be completed, although this increased confidence did not significantly improve their performance in the given time and resource constraints.

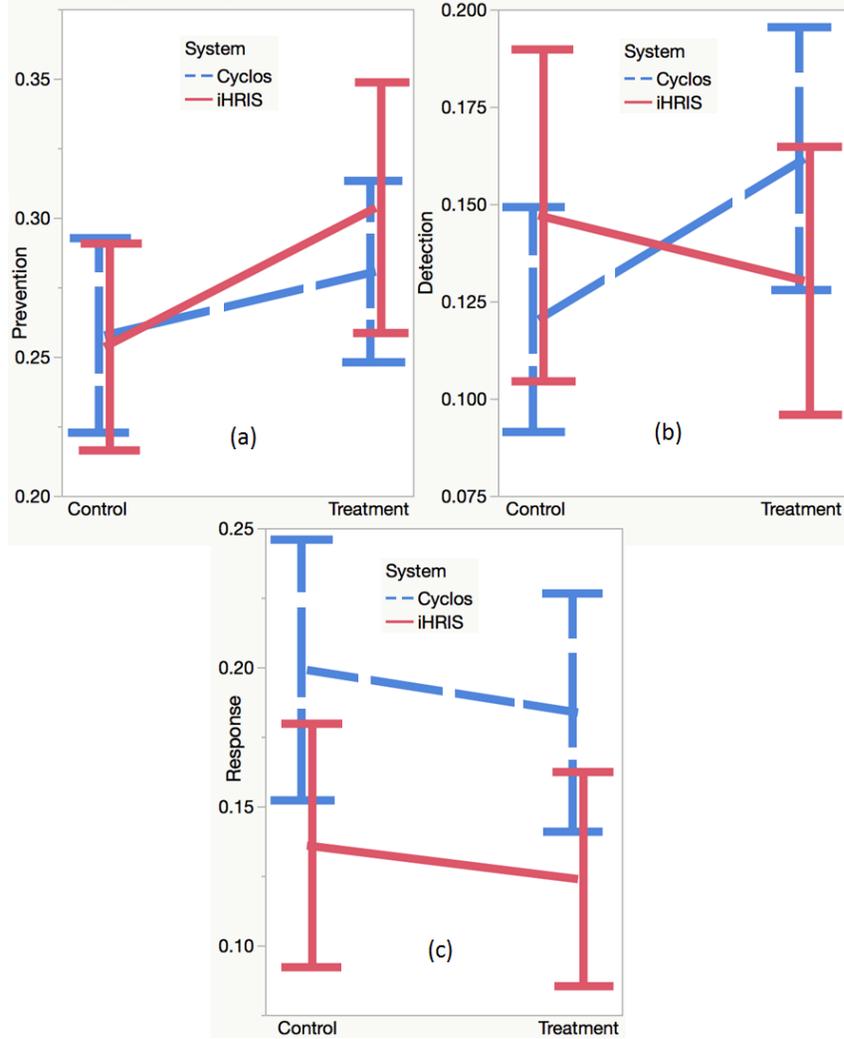


Figure 13. Means and standard errors of standardized responses for the correctly identified (a) Prevention, (b) Detection, and (c) Response security goals.

*Participants in both groups identified goals related to different security actions albeit the overall recall was low. Although providing a systematic process on top of the knowledge codified as security goal patterns does not seem to significantly improve the discovery of security goals, the systematic process can lead to a more positive experience while performing the task.*

## 5.6.2 H<sub>02</sub>: Implied Security Goals

We evaluate if the differences in knowledge of implied security goals had an effect on the quantity of implied security goals identified by both groups. We investigated the differences between the groups based on the proportion of correct goals that were identified for the initial set of assets (initial goals) and the proportion of correctly identified implied goals. Figure 14 has the respective mean and standard errors by task group and system.

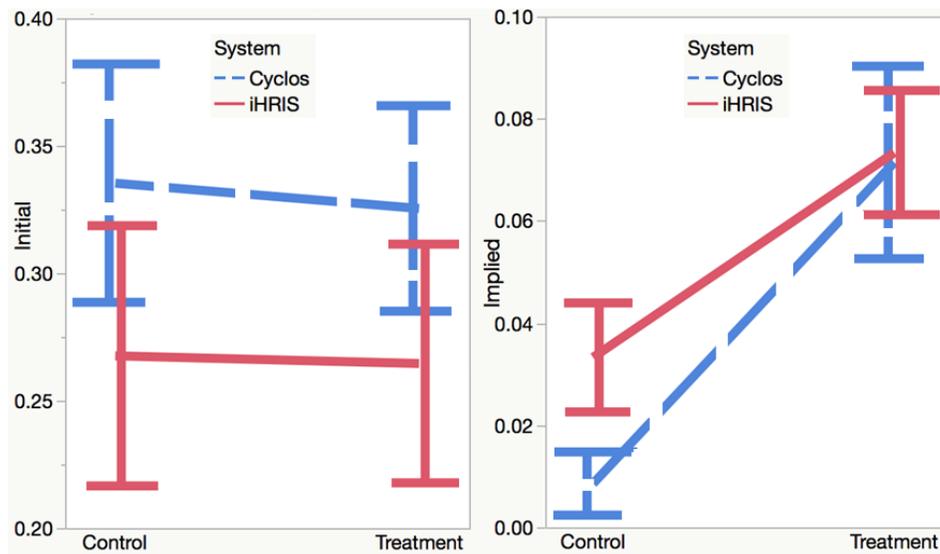


Figure 14. Means and standard errors of standardized Initial and Implied goals correctly identified.

Participants in the treatment group identified significantly more implied security goals as compared to the control group based on the p-value (0.003) as shown in Table 20. Overall, a small proportion of implied security goals were identified for both the treatment and control groups. This was due to the proportions being calculated with respect to the total number of implied security goals in the oracle, while the participants were only able to identify such goals from their set of discovered initial goals. Differences in the proportions would be even

more pronounced if we standardized the proportions with respect to the possible implied goals they could identify.

Table 20. Analysis of variance for standardized Initial and Implied goals.

	Initial		Implied	
Effect	F-value	P-value	F-value	P-value
Task	0.020	0.888	10.60	<b>0.003</b>
System	3.253	0.083	2.251	0.146
Task* System	0.004	0.953	1.638	0.212

Thus we reject the null hypothesis  $H_{02}$  that explicit knowledge of implied security goals does not impact participants' ability to elicit these implied goals.

*Explicitly coding the knowledge about implied security goals supports the discovery of these goals. However, only a small proportion of implied security goals were identified for both the treatment and control groups overall.*

### 5.6.3 Combined Security Goals by Group

Given that the recall by individual participants was low overall (Section 5.6.1), we wanted to compare the control and treatment groups in terms of the recall of the combined set of security goals identified by each group. We considered each group as if they were a team of analysts or stakeholders individually working to identify a set of security goals for the system.

For the combined analysis of recall for each group, any goal in the oracle that was identified by any of the participants in a group was counted as identified by that group. We list the recall values for initial and implied goal, as well as total recall for both the systems for treatment and control groups in Table 21. The control group, as a team, identified 61-62%

of applicable security goals, whereas the treatment group identified 74-79% of all goals as a team. The difference in recall between treatment and control groups is more pronounced for the implied goals (12-24% for control vs. 40-54% for treatment). Of the 131 (101 initial; 30 implied) distinct security goals correctly identified by participants overall, the treatment group identified 27 goals (9 initial; 18 implied) that control did not. Whereas the control group identified only one goal that the treatment group did not. We cannot make any claims about the statistical significance of the observed differences based on these recall values since we did not replicate the groups in our experiment (i.e., we had only one control and one treatment group). However, given that neither group was inherently better at identifying security goals to begin with, these results indicate that participants in the treatment group identified a larger pool of security goals based on the additional knowledge and systematic process available to them. These results additionally indicate that different individuals may prioritize different security goals and working as a team, they might be able to carry out a more comprehensive analysis of the system's security.

Table 21. Recall of combined security goals – DIGS  
*[C: Control; T: Treatment]*

Goals	iHRIS		Cyclos	
	C	T	C	T
<b>Initial</b>	0.82	0.92	0.94	0.97
<b>Implied</b>	0.24	0.54	0.12	0.4
<b>Total</b>	<b>0.62</b>	<b>0.79</b>	<b>0.61</b>	<b>0.74</b>

*Participants using DIGS, when considered as a team, identified a larger pool of security goals as compared to the control group.*

### 5.6.4 Breakdown of Identified Goal Patterns

We looked at the breakdown of the types of security goals that are identified by the participants in comparison to the security goals in the oracle for both iHRIS (Figure 15) and Cyclos (Figure 16).

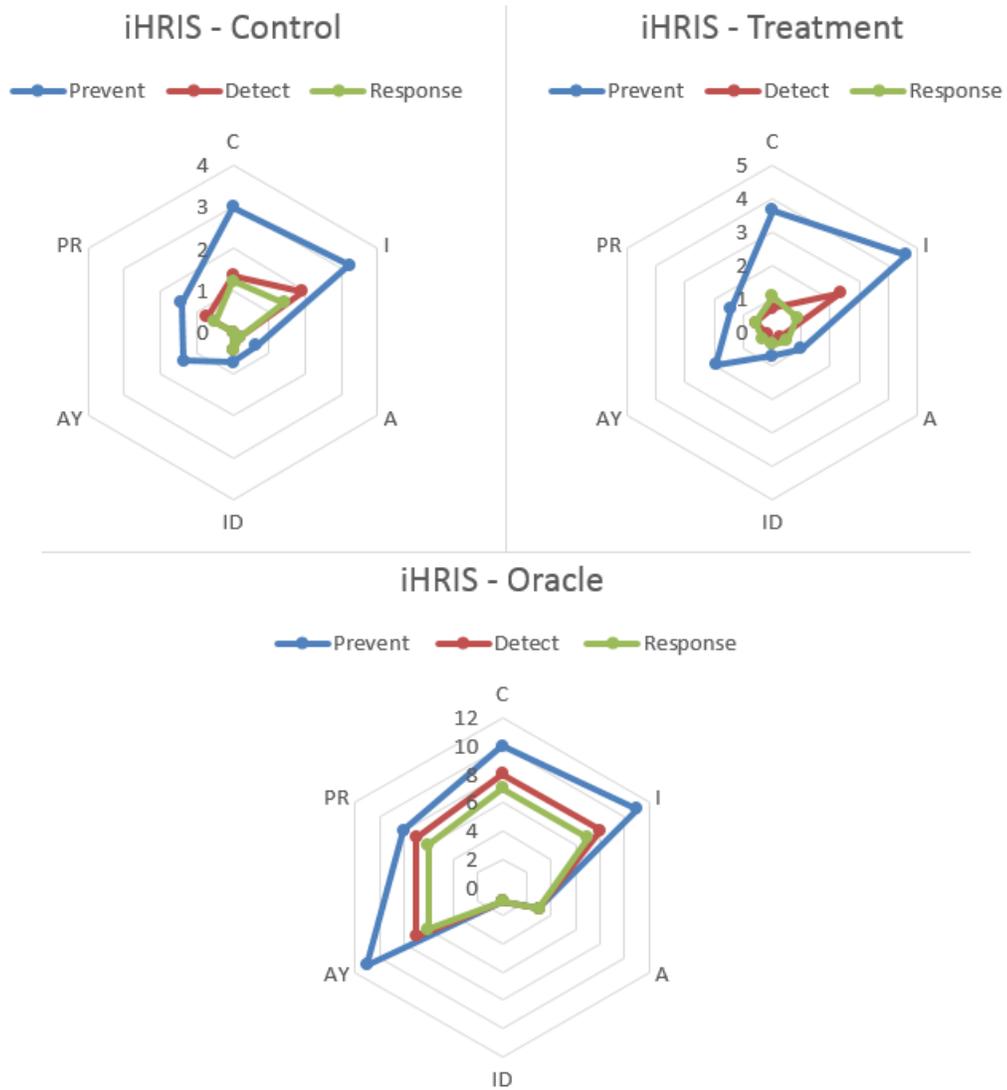


Figure 15. Breakdown of Security Goals in the Oracle vs. Identified – iHRIS

For iHRIS, the most common goals in the Oracle are related to preventing, detecting and responding to breaches of confidentiality, integrity, accountability and privacy. Whereas the

most commonly identified security goals for iHRIS by the participants are related to preventing a breach of confidentiality and integrity as well as detecting breaches of integrity.

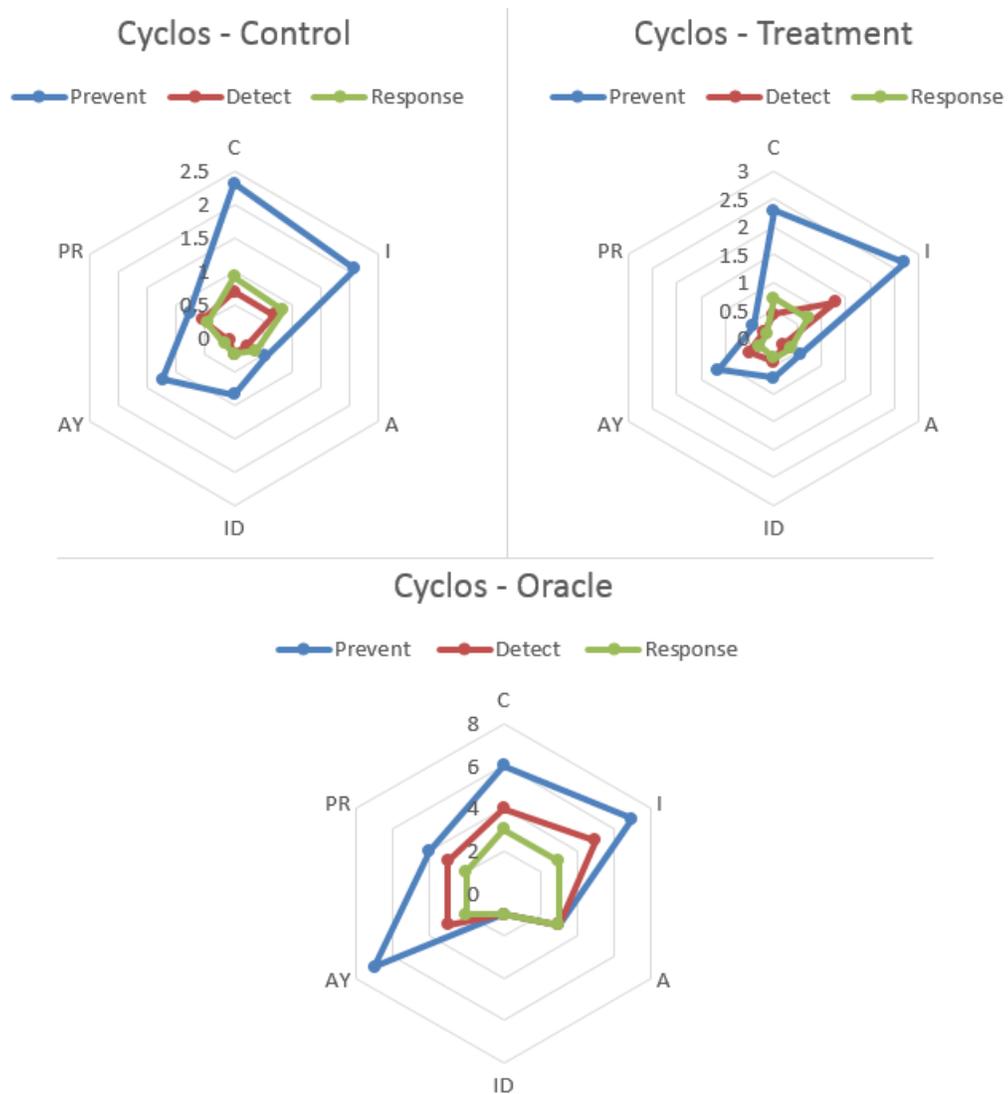


Figure 16. Breakdown of Security Goals in the Oracle vs. Identified – Cyclos

For Cyclos, the most common goals in the Oracle are related to preventing, detecting and responding to breaches of confidentiality, integrity and accountability. Whereas the most commonly identified security goals are related to preventing a breach of confidentiality and

integrity as well as detecting and responding to breaches of integrity. Based on the findings, we see the trend of identifying similar security goals in both systems.

The goals for identification and authentication and availability applied to the whole system instead of individual resources. Therefore, we see a smaller number of such goals in the oracle as well as among the goals identified by the participants.

### **5.6.5 Threats to Validity**

We considered following threats to internal validity:

*Selection:* We used results of a pre-task quiz to create groups such that no group was inherently better at identifying security goals than the other. As part of the post-task questionnaire, we further asked participants about relevant expertise. Consequently, groups were evenly balanced in terms of security expertise (approximately one year of academic security experience) except for one participant in the control group who had over eight years of security experience. That participant had the highest total recall in the control group.

*Interactions with selection:* We do not have knowledge about how motivated or security-aware each participant was. This knowledge might have further helped in assessing why some participants are more inclined to identify security goals as compared to others.

*Training:* In the time given to perform the task, participants in the treatment group had to additionally understand DIGS whereas control group did not have to understand any new methodology. Although participants using DIGS understood the framework based on the feedback, allowing participants in the treatment group to understand DIGS prior to the task might have levelled the teams in terms of time available to solely focus on the task.

We considered following threats to external validity:

*Representativeness of sample population:* Participants had around one year of security related academic experience on average and a few months of security related work experience on average. In this respect, they can be considered equivalent to entry-level, non-expert security practitioners.

*Task representativeness:* We provided a high-level description of two real world systems from different domains for the task. The task was fairly representative. However, the time and resources to carry out the task were limited.

*Experimental constraints that limit realism:* Security analysts are usually familiar with the problem domain and work as part of a team to identify security goals over a period of weeks or months. The experimental constraints could have led to overall low recall of security goals.

We considered following threats to construct validity:

*Measures used:* We used standard measures of precision and recall computed against a priori established oracle to assess participants' performance in identifying security goals.

## **5.7 Discussion and Lessons Learned**

We have presented the DIGS framework and results evaluating the framework using a split-plot design, allowing us to extract more information about the split-plot effects (iHRIS vs Cyclos) and their interactions with the task group without having to add more participants. Two key components of DIGS are the security goal patterns and explicitly documented implied security goals. Participants in the control group were partially familiar with DIGS, specifically, security goal patterns. Both groups identified security goals corresponding to different patterns, covering multiple security properties and actions captured in the patterns.

Consequently, we could not evaluate how the control group might have performed in the absence of the knowledge of security goal patterns. However, based on the findings related to implied goals in the current experiment and findings from our previous experiments (Riaz, Slankas, et al. 2014), we would expect to see more pronounced differences between control and treatment in terms of the different types of security goals identified and recall of security goals.

Participants using DIGS reported a more positive experience performing the task as compared to the control group and seemed robust to the effects of fatigue in the current experimental setup. The availability of a systematic process might have lessened the cognitive load on participants thus leading to a more positive experience. This effect is worth exploring in future studies.

Participants having similar understanding of the background concepts based on pre-quiz, performed differently when identifying the security goals. One of the reasons may be that participants who are similar in capability may be different in terms of security-awareness (Hibshi et al. 2014). Future experiments should factor in that additional information when creating groups. Assigning the tasks to groups rather than individuals may additionally affect the discovery of security goals, potentially improving recall.

Our results indicate that participants are able to consider security goals commensurate to the knowledge available to them. Although the overall recall was low, participants using DIGS reported a more positive experience while performing the given tasks. Moreover, when considered as a team, participants using DIGS identified a larger pool of security goals as compared to the control group.

We capture goals related to prevention, detection and response actions for various security properties. We also capture relations among security goals in the form of implied security goals. However, in some cases, the goals identified using DIGS might lead to redundant security requirements. For instance, the goal for detecting breaches often imply a need for accountability of the users and corresponding actions. The requirements generated to support these goals (e.g., auditing of events, generating audit reports) might be closely related or even redundant. Similarly, to support confidentiality and integrity, we might use similar security requirements for access control and enforcement. Further analysis is needed to identify the situations in which two goals may lead to redundant security requirements and should be considered together to streamline the analysis. If some goals are identified that will always lead to redundant requirements, it might indicate the need to merge the corresponding goal patterns.

## **5.8 Summary**

We have presented the DIGS framework for systematically discovering security goals for the assets in a system. DIGS supports organizing the security goals related to a particular asset, security property, or security action. This organization is intended to help in quickly identifying areas where goals have not been specified and that may need additional security fortification. By providing a set of 18 security goals patterns and corresponding implied goals, we assist an analyst to consider security of assets from multiple dimensions. We have also developed security requirements patterns to support each of the identified security goals (Section 6). In future, we plan to integrate DIGS in a tool to automate parts of the analysis. We have evaluated DIGS via a controlled experiment where 28 participants analyzed systems

from mobile banking and human resource management domains. Our results indicate that participants considered security goals commensurate to the knowledge available to them. Although the overall recall was low given the empirical constraints, participants using DIGS identified more implied goals and felt more confident in completing the task. We conclude that explicitly providing the additional knowledge for the identification of implied security goals significantly increased the chances of discovering such goals, thereby improving coverage of stakeholder security requirements, even if they are unstated. Our research contributes towards systematic identification of security goals and helps in considering security requirements to meet those goals that may have been missed otherwise.

## **6 PREVENTION, DETECTION AND RESPONSE PATTERNS FOR SECURITY**

### **REQUIREMENTS ENGINEERING**

The output of the SRD process is a set of candidate security requirements that are implied, rather than explicitly stated, in the input requirements artifacts. We need to have a set of security requirements patterns that can be instantiated in the context of a system to explicitly specify the implied security requirements. We have performed three sets of activities for developing security requirements patterns to support the specification of security requirements for the security properties identified from natural language requirements artifacts: (1) analyze existing literature on software patterns, problem solving and cognition to outline the process for developing software patterns; (2) review strategies for specifying reusable security requirements and security requirements patterns; and (3) propose and evaluate a systematic process for developing security requirements patterns from existing security requirements knowledge sources. We catalogue the identified security requirements patterns in terms of the security goals that are supported by the requirements patterns. Our security requirements patterns cover a comprehensive set of security properties while considering goals related to the prevention of, detection of and response to security breaches. The contextual information provided as part of the patterns can support the analyst in applying the appropriate patterns in the context of a software system.

#### **6.1 Understanding the Process for Developing Patterns**

Identifying and formulating patterns is an important aspect of natural intelligence. The highest levels of learning, according to Bloom's taxonomy, involves analysis and synthesis,

or breaking wholes into parts and recombining these parts into new wholes (Bloom and Krathwohl 1956). Similarly, Alistair Cockburn identifies the highest levels (2 and 3) of software developer competence as the ability to mix and match elements of a process or to construct their own patterns (Boehm and Turner 2003). Based on existing literature on software patterns, problem solving and cognition, we have identified core building blocks of the pattern development process in software engineering to guide the efforts of new pattern writers.

### **6.1.1 Knowledge Sources**

Knowledge sources serve as immediate input to the pattern development process whether the patterns are discovered in a top-down manner or systematically identified through an exploratory bottom-up process. A knowledge source can be:

- Individual experiences (Gamma et al. 1995)
- Common challenges in a domain (Schmidt 2007)
- Common vulnerabilities (Smith and Williams 2012)
- Standards and Best practices (Schumacher et al. 2006)
- Enterprise and system architecture resources (Schumacher et al. 2006)
- Software artifacts (Dong, Zhao, and Peng 2009)
- Specifications and documentations (Withall 2007)
- People solving the problem (Kerth and Cunningham 1997)
- Processes used to solve a problem (Kolfshoten et al. 2010) (Hibshi et al. 2014)

Often a combination of these sources is used and synthesized to identify recurring problem themes and commonly adapted solutions for resolving these problems.

### 6.1.2 Knowledge Synthesis and Pattern Identification

Synthesizing available knowledge sources involves generation of larger and complex mental schemas at a higher level of learning and cognition (Bloom and Krathwohl 1956)(Kolfshoten et al. 2010). Complex schemas are helpful in information storage and retrieval and allow processing of larger pieces of information simultaneously to facilitate problem solving. Experts in a problem domain have larger schemas when compared to novices. They are able to recognize recurring problems and incorporate directions for solving a problem based on past experience (Sweller 1988).

Kohls et al. have discussed implications for mining patterns by looking at the schema theory and identifying parallels between design patterns and problem schemas (Kohls and Scheiter 2008). A problem schema is the unit of knowledge available, usually to domain experts, after having solved a problem for the first time. If a recurrent problem is encountered, a ready-to-use solution procedure, or problem schema, can be applied which contains both the classification of problem and the skills to solve the problem (Kohls and Scheiter 2008).

Kerth et al. (Kerth and Cunningham 1997) note that patterns may be identified in varying ways however three approaches are generally used:

- *Introspective* approach is based on identifying recurring problems encountered over years of experience and abstracting out the solution methodology employed. It may reflect an individual's style or preference;

- *Artifactual* approach is where the pattern investigator studies numerous software artifacts developed by different teams solving similar problems to identify patterns; and
- *Sociological* approach studies the people involved in building similar systems and identifying patterns using an interactive approach, such as interviews.

Withall (Withall 2007) has described two general approaches for developing requirement patterns:

- *Systematic* approach looks at all available requirements specifications (knowledge sources) to identify requirements of interest and see if they fit an existing pattern or indicate a new pattern. The patterns generated this way can be later refined iteratively;
- *Opportunistic* approach is when you look at a specific requirement and note that there might be other requirements of the same type and a requirement pattern may be helpful in generating this type of requirements.

Withall (Withall 2007) employs the *systematic* approach for his requirements patterns and argues that this approach can indicate prevalence of different types of requirement patterns by maintaining relevant stats during the systematic process. Smith et al. (Smith and Williams 2012) have used a similar approach based on a grounded analysis of commonly occurring security vulnerabilities to identify black-box security test patterns.

Several researchers have looked at pattern mining from machine learning perspective (Dong, Zhao, and Peng 2009). However, these approaches are limited to identifying already known patterns and have limited application to discovering new patterns.

### **6.1.3 Pattern Representation**

Pattern representation is the process of formalizing and documenting the pattern in a format that facilitates knowledge capture and reuse. Meszaros et al. (Meszaros and Doble 1996) have documented best practices for writing patterns in the form of meta-patterns and discuss the mandatory and optional sections of a pattern representation. A pattern generally documents:

- types of problems that are addressed by the pattern;
- situation or context in which the pattern is applicable;
- general template of the core solution;
- forces and tradeoffs involved; and
- one or more examples of pattern's application
- known uses for the pattern

Other sections can be added based on the domain of the pattern. Withall (Withall 2007) includes additional sections on design and testing considerations for requirements patterns to specify constraints for proper implementation and testing. Multiple representation schemes can be used for each section with natural language representation (Schumacher et al. 2006) and formal modeling notations (Gamma et al. 1995) being the most common.

### **6.1.4 Pattern Refinement**

We acquire knowledge continually and incrementally. Pattern development is an iterative and often on-going process incorporating feedback from experts and users (Vlissides 1998). Pattern refinement can result in tailoring the pattern to accommodate minor changes to make

the pattern more understandable and applicable (Withall 2007). Refinement process may also lead to more pronounced changes:

- *Splitting* a pattern if it is over-generalized or complex (Withall 2007);
- *Abstracting* two or more similar patterns into a single pattern if it is over-specified; or
- *Specializing* by defining a specialized pattern related to an existing pattern (Vlissides 1998).

### **6.1.5 Pattern Application**

The highest learning and expertise levels are typically represented in only a small percentage of professionals (Boehm and Turner 2003). However a much larger proportion of professionals possess the skillset to apply patterns to problem descriptions based on factors such as: how well the pattern is documented; how well the problem is presented; and how relevant the examples are to the problem at hand. Some of these factors can be taken into account while writing patterns to improve applicability and reuse. Based on our systematic review (Riaz, Breaux, and Williams 2015), we have identified three main cognitive stages that are encountered when using patterns for problem solving:

1) *Pattern Selection*: Pattern selection is the process of identifying appropriate patterns that can be used to solve a given problem. Experiments, aimed at assessing the ability of users to successfully use software patterns, have found that users, especially novices, often struggle during pattern selection process (Jalil and Noah. 2007). Providing multiple domain-specific examples as part of pattern documentation aided pattern selection by the users. Use of tools that can provide suggestions for applicable patterns was also found helpful in some cases (Kolfshoten et al. 2010). In some cases, wording used in a problem statement or

certain keywords that hint when a pattern is applicable may account for why some patterns are easier to select as compared to the others (Jalil and Noah. 2007). Provision of keywords or phrases indicating patterns' applicability may be useful for requirement patterns which are often expressed in natural language.

2) *Pattern Instantiation*: Pattern Instantiation: Pattern instantiation or application is the process of mapping pattern description or template to a concrete pattern instance. Pattern instantiation follows the stage of appropriate pattern selection however a pattern may at times be pre-selected to be instantiated. Pattern instantiation can be challenging, especially for novice users and may result in errors and omissions when going from generic pattern description to specific pattern instance (Jalil and Noah. 2007).

3) *Pattern Maintenance*: Pattern maintenance is the process of modifying existing pattern instances to extend or adapt existing solutions. Pattern comprehension is an integral part of pattern maintenance as comprehension of the existing pattern instances is necessary to successfully conduct a maintenance activity. Patterns are found useful for maintenance activities and when participants are aware of the existence of patterns in a system, they are more likely to utilize these patterns during maintenance (Vocak et al. 2004).

## **6.2 Considerations for Writing Security Requirements Patterns**

Based on the work in reusable security requirements and security requirement patterns, we have identified the following considerations when writing security requirements patterns:

### **6.2.1 Sources of Security Requirements**

Security patterns for secure design and architecture are often identified based on one or more security standards written by domain experts over multiple iterations (Schumacher et

al. 2006). Standards can be useful for identifying security requirements patterns as they explicitly capture security problems and standard security requirements, albeit in a more concrete form as compared to a requirement pattern. Enterprise security standards, such as ISO 17799 and ISO 13335, provide a hierarchical organization of security properties at management and implementation levels. Standards also leverage best practices and a common vocabulary (Schumacher et al. 2006).

Security certifications and assessment techniques such as Common Criteria<sup>14</sup> for Information Technology Security Evaluation (CC) ensure that products can be evaluated to a certain degree of assurance if they fulfill particular security properties or targets. Each product is assigned a protection profile. Protection profile of an operating system will be different from the protection profile of a web-based application and can be used to identify recurring security requirements.

NIST Special Publication 800-53 (“Special Publication 800-53 Revision 4 - Security and Privacy Controls for Federal Information Systems and Organizations” 2013), specifies a comprehensive list of security and privacy controls for information systems. NIST documents the baseline controls as well as control enhancements for the baseline controls in case additional security fortification is needed in certain areas. We have used the NIST baseline controls to empirically develop security requirements patterns that support various security goals (Section 6.3).

---

<sup>14</sup> <http://www.commoncriteriaportal.org/>

### **6.2.2 Identifying Reusable Security Requirements**

Software systems that share common security properties often also share security requirements (Firesmith 2004). Security properties can be used to categorize different types of security requirements including authentication, authorization, immunity, integrity, intrusion detection, nonrepudiation, privacy, security auditing, survivability, physical protection and system maintenance (Firesmith 2003). Security requirements that help in achieving the same objective across multiple systems may be generalizable in the form of a pattern. Web-based systems may have similar security concerns related to DoS (Denial of Service) attacks. Information systems have the common requirements for integrity and confidentiality.

Requirement specifications of multiple systems are also a good source for identifying security requirements patterns by looking for common requirements related to security. This approach is helpful when creating a comprehensive set of industry-specific security requirements patterns (Withall 2007)(Riaz, King, et al. 2014).

### **6.2.3 Decoupling Requirements from Mechanisms**

Requirements specify what a system must do and not how it should do. A common challenge in specifying security requirements is that requirements are often coupled with the underlying security mechanisms and architectural or algorithmic constraints, pre-committing to a specific design and implementation. This limits the possibility of identifying reusable security requirements. Firesmith (Firesmith 2003) has provided guidelines and examples that can help in achieving this separation between security requirements and mechanisms. Once the design and implementation details are taken out of the security requirements, common

patterns among the requirements can be identified. Design and testing considerations can be included as part of the pattern template where such considerations are useful for instantiation of the requirement pattern (Withall 2007).

#### **6.2.4 Managing Conflicts and Ambiguities**

A security requirement pattern can be instantiated in the context of multiple systems and may result in conflicting and ambiguous requirements if not properly applied. Some of these conflicts can arise due to differences in terminology and representation, which is often a source of ambiguity in requirements specification. Anton et al. (A.I. Antón, J.B. Earp, and Carter 2003) have identified that one of the risks related to aligning requirements with security and privacy policies is conflict between and among policies and requirements. Even when the original set of requirements are written clearly and unambiguously, they might not readily mix with the system-specific set of requirements. E.g., when mapping from a security requirement pattern for access control to concrete access control requirements, we need to replace abstract templates in the pattern to application-specific terminology for concrete requirements. Extracting terminology from the domain (Riaz, King, et al. 2014) or established standards can minimize such conflicts.

Another issue is to understand how other system requirements are affected by the addition of a set of reusable security requirements. Conflicting requirements may arise as a consequence, necessitating conflict resolution and requirements prioritization. Adherence to quality standards specified by IEEE and other organizations can be helpful in mitigating some of these challenges (A. Toval et al. 2002).

### **6.2.5 Maintaining Dependencies and Traceability**

Security requirements can stem from diverse sources including security standards, security policies, security properties, threats and vulnerabilities analysis, risk assessment, regulatory or compliance requirements. Maintaining traceability to the source of a requirement and documenting the rationale behind a given requirement is important for requirements maintainability and prioritization (A. Toval et al. 2002). Requirements traceability is a challenge in itself, especially in the face of changing requirements. Ramesh et al. (B. Ramesh and Jarke 2001) have identified different types of traceability links including: satisfies, evolves-to, rationale and dependency links. Patterns can capture traceability links by maintaining dependencies between patterns and grouping related requirements in a single pattern to reuse as a set e.g., non-repudiation requirements pattern leads to a general set of requirements appropriate to non-repudiation services (Riaz, King, et al. 2014).

Capturing the intent of the requirements is important to minimize information loss when porting requirements from system-specific to reusable. However, identifying the intent of a requirement, whether it is system-specific or meant to be reused, is difficult at times. Maintaining a reference to the source of requirement and including considerations for design and implementation can be used to explicitly document intent (Withall 2007).

## **6.3 Systematic Process for Developing Security Requirements Patterns**

In this section, we present our methodology for systematically developing security requirements patterns from security knowledge sources. Riaz et al. (Riaz and Williams 2012) have identified the key elements of the process for identifying patterns including knowledge

sources, knowledge synthesis and pattern identification, and pattern representation. We discuss each of these below in the context of our process for identifying security requirements patterns.

### **6.3.1 Identifying and Analyzing the Knowledge Source**

Knowledge sources are the input to the pattern identification process. We use the NIST Special Publication 800-53 (“Special Publication 800-53 Revision 4 - Security and Privacy Controls for Federal Information Systems and Organizations” 2013) security and privacy controls as the knowledge source for developing the patterns discussed in this paper. However, the process described here can be applied to other knowledge sources as well to identify security requirements patterns. NIST documents a comprehensive set of security and privacy controls for federal information systems. However, the controls provide a valuable resource for systems that manage sensitive information such as web and enterprise applications, healthcare systems, and similar networked applications. NIST catalogs the controls in terms of 18 different families ranging from access control (AC), audit and accountability (AU) to physical and environmental protection (PE) and personnel security (PS). Overall, NIST documents over 190 baseline controls within these families, as well as control enhancements for the baseline controls in case a particular area requires additional security fortification. Each control is assigned a unique identifier capturing the family the control belongs to and the order in the list within the family. For instance, AC-1 is the first control listed as part of the access control family. NIST also assigns priority levels to the controls in terms of which controls should be implemented first (P1), next (P2) and last (P3) that provides additional guidance to the analysts. For instance, control AC-3 related to

‘access enforcement’ is at priority P1. Whereas control AC-7 related to ‘unsuccessful logon attempts’ is at P2 as it comes after access enforcement is in place. The controls are related to policy, configuration, management and technical aspects of security and privacy of an information system. For the purpose of developing security requirements patterns, we focus on the 114 technical controls that deal with security-related functionality of the information system. Controls that are implemented at the organizational level beyond the scope of the information system, such as policy specification, personnel training and software development lifecycle management, are not included in our analysis.

We have identified a set of 18 security goals patterns in Table 14 using the DIGS framework. We use the goal patterns as a structure for organizing the NIST controls and subsequent security requirements patterns developed from the controls. As a first step for developing security requirements patterns, we organize the NIST controls in terms of the security goals that each control is related to. Each control supports one or more of the six security properties and addresses security concerns during one or more of the three security actions related to prevention, detection and response to security breaches. Most of the controls, such as access control and authentication, focus on preventing security breaches from happening in the first place. A number of controls, such as the ones related to auditing and monitoring of the information system, deal with the security action of detecting a security breach that could not be prevented. Whereas controls related to incident response and taking remedial actions (disconnecting, blocking users for instance) fall in the category of responding to a security breaches.

We provide a summary of the technical NIST controls for each security property and action type in Table 22. The numbers in parentheses provide the total counts within each category. Since a control can belong to multiple categories, the sum of counts in each row and column is not meaningful. Most of the technical controls are related to preventing security breaches from occurring in the first place. However, in case a breach does occur, NIST provides a proportional number of controls for detecting and responding to potential breaches. We look at the controls that belong to the same category to identify common themes and patterns as explained in the subsequent section. The complete mapping is available at the project website<sup>15</sup>.

Table 22. Summary of NIST controls by Security Goals

	<b>Prevention</b> (84)	<b>Detection</b> (27)	<b>Response</b> (28)
<b>Confidentiality (62)</b>	44	14	10
<b>Integrity (76)</b>	55	21	16
<b>Availability (39)</b>	26	11	20
<b>Identification &amp; Authentication (43)</b>	28	14	9
<b>Accountability (28)</b>	13	11	7
<b>Privacy (49)</b>	31	15	6

Cataloging the controls, as discussed here, allows us to consider related controls simultaneously. For instance, by having all the controls related to detecting security breaches together, we can identify various approaches used to detect the breaches. Once the controls are catalogued in terms of the security goals, patterns of security requirements start emerging.

---

<sup>15</sup><https://sites.google.com/site/digsstudy/>

The cataloging of NIST controls enabled us to analyze similarities among controls that are classified in the same way. We classify each control in terms of one or more security properties and one or more security actions. For instance, the control AC-3 (access enforcement) is mapped to the properties of confidentiality, integrity and privacy and the ‘prevent’ security action indicating that the control supports preventing a breach of confidentiality, integrity and privacy. The control SI-4 (information system monitoring), is mapped to all the security properties and the ‘detect’ security action as it supports detecting a breach of security properties. The control SI-6 (security function verification) supports both detection and response security actions for all security properties.

We convert the classified controls from textual sentences to word vectors (Joachims 1998) to identify attributes (keywords) related to each classification. We then compute the information gain values (Quinlan 1986), an indicator of how much an attribute contributes towards predicting the classification, to rank the attributes.

In Table 23, we list the top 20 attributes (top ~1% of attributes) related to the controls classified for each security property. Some of the attributes are common among all the properties due to overlapping security properties and similarity of language in the document. However, we see some distinct attributes as well. Identify, approve, authorize and policy are among the top attributes associated with confidentiality. Incident, response, contingency, recover and availability are among the top attributes associated with the property of availability. Audit, incident, record, report, investigate and monitor are among top attributes for accountability. For privacy, we see attributes related to different types of storage media.

Table 23. Top 20 ranked attributes for security properties – NIST controls

C	identify, operate, approve, authorize, process, execute, policy, transmit, classify, risk, authenticate, eliminate, name, combination, incident, traffic, mission/business, activate, ability, standard, media
I	identify, malicious, manage, authenticate, combination, implement, code, component, implement, enforce, configure, impact, logon, need, social, focus, covert, unique, capacity, situation, exceed
A	incident, media, identify, operational, response, contingency, recover, virtualize, availability, mechanism, act, behalf, occur, require, component, authenticate, resolve, complete, client, authoritative, wireless
ID	authenticate, identify, period, session, incident, federal, device, storage, number, support, status, access, response, time, user, require, include, commercial, local, order, directive, remote
AY	audit, capability, incident, record, generate, address, authenticate, general, report, frequency, investigate, monitor, alternate, granularity, after-the-fact, subset, procedure, integrity, legal, event
PR	operate, identify, authorize, number, authenticate, transmit, classify, failure, legal, apply, ability, incident, hard, media, category, non-digital, disk, risk, magnetic, device, refer, commensurate

In Table 24, we provide the ranked attributes associated with each of the security actions related to preventing, detecting and responding to breaches. We again see overlap of attributes as well as distinct attributes. For instance, for controls related to preventing breaches, the top attributes include incident, action and maintenance. For detecting breaches, the top attributes include detect, incident, and monitor. For responding to breach, we see attributes including failure, down, response, increase and alternative.

Table 24. Top 20 ranked attributes for security actions – NIST controls

P	incident, detect, action, response, between, report, human, monitor, maintenance, equipment, unauthorized, impact, alternative, aspect, personnel, control, digital, software, until, include
d	response, detect, incident, monitor, personnel, frequency, policy, tool, unauthorized, human, report, program, maintenance, equipment, aspect, organization, obtain, role, between, legal, source, digital
r	failure, external, integrity, down, confidentiality, response, increase, mode, alternative, applicable, federal, available, mobile, further, assign, organization-defined, policy, content, media, partition

Although we do not use the attributes as keywords to identify security requirements patterns, some of the attributes indicate themes across the controls that can be useful for grouping related controls and assigning descriptive names to the identified patterns. An individual researcher identified the initial mapping of security goals for the NIST controls.

We created a random set of mapped controls for training a second researcher (~20% of controls in the training set). The second researcher used the training set to understand the mapping process and independently assigned mappings to a separate randomly created test set (~20% of controls in the test set). Based on the Cohen's kappa statistics (Viera 2005), the two raters had substantial agreement in determining whether a control was technical or not (Kappa score = 0.62). The raters also had moderate agreement in terms of the security properties and actions each control mapped to (Kappa score = 0.46). Kappa score tends to go down as the number of categories increase.

### **6.3.2 Synthesizing Knowledge for Pattern Identification**

We look at the controls related to each security action individually to synthesize the information and develop security requirements patterns. We employ a systematic process for identifying patterns using an artefactual approach Kerth et al. (Kerth and Cunningham 1997). The steps for identifying security requirements patterns from the controls, catalogued by security goals (see previous section), is as follows:

- 1) *Group controls by security action type:* As a first step, we create separate lists of all the controls related to each security action identified during the analysis of the knowledge source. Some controls will appear in multiple lists (e.g., if a control talks about both prevention and response). In those cases, we consider the elements of the control that are pertinent to the security action under consideration. For instance, we create a list of all the controls related to 'response' security action.
- 2) *Combine controls by security properties:* For the controls related to a particular action (e.g., response), we identify all the combinations of security properties that the

controls belong to. Some controls, such as incident monitoring (IR-5), support all six security properties while other controls, such as component authenticity (SA-19) support only one or two of the security properties. Color coding can be used at this step to keep track of the different combinations of security properties. For instance, we identify that controls AU-5 (response to audit processing failures) and AU-15 (alternate audit capability) both map to ‘accountability’ for the set of controls related to ‘response’ action and are considered together at this step.

3) *Identify templates of security requirements:* At this point, we are looking at a small enough subset of security controls to start abstracting out templates of security requirements. The templates capture the information provided as part of the control at a high level of abstraction, factoring in the scope of the control. For instance, we have controls related to protecting the information (e.g., detect unauthorized disclosure of information), protecting the users (e.g., recording unsuccessful login attempts), protecting the system itself (e.g., detect if code is untrusted or malicious), as well as controls for protecting the security-related functionality (e.g., verify integrity of security functions). The information provided as part of each control is represented in at least one template. We keep track of the controls that are used to develop each template by documenting the corresponding NIST identifier. For the control AU-5 (response to audit processing failures) we identify the template ‘The system shall respond to audit processing failures by alerting <authorized user>’. For the control AU-15 (alternate audit capability), we identify the template ‘The system shall have

provision for alternate audit capability to record <designated actions> if the primary audit capability fails’.

- 4) *Refine templates of security requirements:* In this step, we iteratively refine the identified templates by merging similar templates or splitting a complex template into multiple templates. We also remove redundant templates. For instance, if a template related to confidentiality is similar to a template that is applicable to all the security properties, we can remove or merge the template for confidentiality. This process also resolves cases where the classification of controls may have been inconsistent.
- 5) *Identify requirements templates with similar themes:* In this step, we look at the refined security requirements templates to identify templates that are related (e.g., all the templates that are related to monitoring of resources) and thus should be considered as a group to comprehensively address a specific security problem. These groups of templates will form the solution part of the security requirements patterns.
- 6) *Document contextual information:* During the process of identifying, refining and grouping templates, we document any additional contextual information that may arise. For instance, some controls may be applicable when a user is locally accessing the system while other controls are related to remote access. Similarly, the controls may specify the type of device that is used to access the system (desktop versus mobile for instance). The contextual information, along with the templates, will be captured in the corresponding security requirements patterns and help in identifying the applicability of the patterns.

At the end of applying the above steps, we have identified the common themes across the NIST controls and documented the necessary contextual information to start documenting the patterns.

### 6.3.3 Documenting Security Requirements Patterns

We adapt the requirements patterns template by Chung et al. (Chung et al. 2012) for documenting our security requirements patterns. We briefly discuss the elements of the patterns as follows (*\* indicates necessary elements*):

- *Name\**: A unique identifier and a descriptive name for the pattern. The identifier abbreviates the security goals that the patterns supports for quick lookup. For example, P-C\_PR-1 indicates that the requirements generated with this pattern will support the goals for preventing a breach of confidentiality and privacy.
- *Problem\**: The security-specific problem the pattern addresses. We define the problem in terms of the security goals that the pattern supports (e.g., preventing breach of confidentiality or responding to a breach of availability).
- *Context\**: The security context in which the pattern may apply. Context elements can include:
  - Type of the pattern\*: technical, policy, management, configuration.
  - Scope of the pattern\*: information, user, system, communication, security.
    - *Information*: indicates that the pattern supports generating requirements for security of information assets in the system.

- *System*: indicates that the pattern generates requirements for the security of the information system or components of the information system.
  - *User*: indicates that the pattern generates requirements for the security of the users of the information system.
  - *Communication*: indicates that the pattern generates requirements for the security of the communication channels that connect various components of the information system or connect information system with the users of the system.
  - *Security*: indicates that the pattern generates requirements for the security of the security functionality and mechanisms themselves.
    - Type of device the pattern applies to: mobile, desktop.
    - Type of communication channel: local, remote, wireless.
    - Type of users: registered, un-registered.
    - Organization managing resources: internal, external.
- *Solution\**: Group of parameterized templates of related security requirements that should be instantiated in the context of a system to address the security-specific problem.
- *Example*: An example of how to apply the pattern during security requirements engineering to generate security requirements; Known uses of the pattern where the requirements generated from the pattern have been incorporated in a real system.

- *Source\**: Reference to the source artifact(s) used to develop this pattern (e.g., NIST controls identifiers).
- *See Also\**: Other requirements patterns that are related or implied based on the current pattern; other sources that provide information about additional security fortification for the given problem.

We use natural language, which is among the most commonly used representation for requirements patterns (Schumacher et al. 2006)(Withall 2007), to document our security requirements patterns. Our pattern template, however, does not preclude use of additional notations to document the solution. Some of the most common security knowledge sources, including NIST and Common Criteria, also use natural language representation for the security controls and requirements.

#### **6.4 Discussion of Identified Security Requirements Patterns**

We have identified 20 prevention patterns (Appendix-D), 7 detection patterns (Appendix-E) and 8 response patterns (Appendix-F) for security requirements engineering based on our analysis of the NIST controls baselines. We summarize the identified patterns in Table 25.

Our security requirements patterns support the security goal patterns (Section 5.2) and also capture the implied security goals (Section 5.3) for each pattern as illustrated through an example in the following section. We evaluate our patterns through a case study in Chapter 9.

Table 25. Prevention, Detection and Response Patterns for Security Requirements

Security Action	Scope of Pattern	Pattern Identifier and Name
<b>Prevention</b>	Information	P-C_I_PR-1: Prevent unauthorized access to information
	Information	P-C_I_PR-2: Limit authorized access to information.
	Information	P-C_I_PR-3: Restrict access to media.
	Information	P-C_PR-1: Prevent information leakage.
	Communication	P-C_PR-2: Prevent unauthorized remote sensing and activation.
	System	P-AY-1: Enable auditing of events.
	System	P-AY-2: Generate audit records and reports.
	Security	P-AY-3: Protect confidentiality and integrity of audit records.
	System	P-I-1: Runtime protection of system integrity.
	System	P-I_A-1: Limit unnecessary exposure of system functionality.
	User	P-ID-1: Identify and authenticate system users.
	System	P-ID-2: Establish authenticity of system components.
	Security	P-C_I_ID-1: Manage security of identifiers and authenticators.
	Security	P-C_I_ID-2: Manage security of cryptographic functions.
	Security / Communication	P-C_I_ID-3: Manage security of communication session.
	Communication	P-CIA_PR-1: Manage secure internal and external connections.
	Communication	P-I_A_ID-1: Setup secure name and address resolution functions.
	System	P-ALL-1: Enable continuous monitoring.
	System	P-ALL-2: Limit system exposure to persistent attackers.
	System	P-ALL-3: Deceive persistent attackers.
<b>Detection</b>	Information	D-C_PR-1: Detect unauthorized disclosure of information.
	User	D-ID-1: Detect malicious activity during authentication.
	System	D-I-1: Detect malicious system modification attempts.
	System	D-I-2: Detect vulnerabilities in the system.
	Security	D-I-3: Detect problems with security functions.
	System	D-ALL-1: Monitor for security incidents.
	System	D-ALL-2: Analyze audit records to detect malicious activity.
<b>Response</b>	Information	R-C_PR-1: Respond to unauthorized disclosure of information.
	User	R-ID-1: Respond to malicious activity during authentication.
	System	R-I-1: Respond to vulnerabilities in the system.
	Information	R-A-1: Setup backups and recovery procedures.
	System	R-A-2: Prevent and respond to denial of service.
	Security	R-AY-1: Respond to failures in accountability.
	System	R-CIA-1: Respond to system failure conditions.
	Security	R-SEC-1: Respond to threats to security functions.

### 6.4.1 Supporting Security Goals through Security Requirements Patterns

We discuss an example security requirements pattern from our patterns catalog to illustrate how the security requirements patterns support various security goals and implications of those goals (Riaz et al. 2016).

As an example, consider the pattern “P-C\_I\_PR-1: Prevent unauthorized access to information” listed in Appendix D.

- *Name\**: P-C\_I\_PR-1: Prevent unauthorized access to information.
- *Problem\**: <prevent> a breach of <Confidentiality | Integrity | Privacy> of information.
- *Context\**: All access to information should be in accordance with an access control / privacy policy and applicable regulations.
  - Type of the pattern\*: Technical
  - Scope of the pattern\*: Information
- *Solution\**: The system shall:
  - a) enforce that all access to <information | system> are in accordance with <applicable access control policies | access control decisions>.
  - b) enforce that flow of information <within the system | between systems> is in accordance with applicable <information flow policies | access restrictions>.
  - c) <handle | retain> information <within | output from> the system in accordance with <applicable laws | policies | standards | regulations | operational requirements>.

- d) protect the <confidentiality | integrity | privacy> of information <at rest | in use | in transmission>.
- e) have provision to <establish | assign | retain> security attributes for information <at rest | in use | in transmission>.
- *Source\**: AC-3, AC-4, AC-16, AC-21, AC-24, SC-16, SC-8, SC-28, SI-12.
- *See Also\**: P-C\_I\_PR-2: Limit authorized access to information, P-C\_I\_PR-3: Media restriction and protection, P-C\_PR-1: Prevent information leakage, P-ID-1: System users' identification and authentication, P-AY-1: Enable auditing of events, D-C\_PR-1: Detect unauthorized disclosure of information, R-C\_PR-1: Respond to unauthorized disclosure of information.

The pattern covers security requirements related to preventing unauthorized access to information when in use, at rest or in transmission. The three security goals that are supported by the example pattern include preventing a breach of confidentiality, preventing a breach of integrity and preventing a breach of privacy of information, as listed in the 'Problem' part of the pattern.

The 'Solution' section of the pattern contains a set of security requirements templates that as a group generate security requirements to address the security goals specified in the 'Problem' section.

In the 'Source' section of the pattern description, we list the identifiers for NIST controls that are used to develop this pattern and can be referenced for further details.

The 'See Also' section of the pattern points to other patterns that are related or should be considered based on the implied security goals. For example, to support the goal of

preventing a breach of confidentiality and integrity, an implied goal is identification and authentication of the actors (Riaz et al. 2016). This relationship is captured by including the pattern “P-ID-1: System users’ identification and authentication” as part of the related patterns in the ‘See Also’ section. The pattern P-ID-1 includes security requirements templates for uniquely identifying users and allowing access to system only after authentication as part of the solution. Another related security goal is integrity of access enforcement mechanism. This relationship is captured by including the pattern “P-C\_I\_ID-1: Manage security of identifiers and authenticators” as part of the ‘See Also’ section of the pattern P-ID-1.

To maintain accountability when the information is accessed, a related pattern, as listed in the ‘See Also’ section, is “P-AY-1: Enable auditing of events”. Pattern P-AY-1 can generate security requirements for the capability to perform audits in accordance with an audit and accountability policy and procedures (identifying auditable events, defining frequency of audits, allocating sufficient audit storage etc.). The ‘See Also’ section also points to patterns for detecting and responding to unauthorized disclosure of information to provide a comprehensive set of security requirements related to confidentiality and privacy of information assets in the system.

#### **6.4.2 Limitations**

We have considered the baseline NIST controls in our analysis. The control enhancements provided by NIST can provide additional security information. Once a pattern is selected and the analysts is interested in additional information, they can refer to the respective control enhancements provided by NIST for additional guidance. Moreover, NIST

controls are related to information systems and patterns developed from these controls may not be readily applicable to other types of systems.

The list of identified security requirements patterns, although extensive, is not a comprehensive list by any means. The systematic process documented in this paper can be used to identify additional patterns or refine existing patterns by including additional sources of security requirements knowledge, such as Common Criteria (“Common Criteria for Information Technology Security Evaluation, Version 3.1. Release 4” 2012).

Some of the NIST controls are more specific while some are at a higher level of abstraction. For instance, the control AC-12 ‘Session Termination’ specifically talks about terminating user sessions if a security breach is detected. Whereas the control RA-5 ‘Vulnerability Scanning’ is more generic. These differences are also reflected in the patterns that are identified based on the respective controls (For instance, R-ID-1: Respond to malicious activity during authentication related to AC-12 and D-I-2: Detect vulnerabilities in the system related to RA-5).

## **6.5 Summary**

Identifying and specifying security requirements patterns is a non-trivial task and requires expertise in system security, requirements engineering and pattern development. Security community has established a number of knowledge sources, including security catalogues and controls, that capture security expertise and can support elicitation of security requirements. However, these resources may not be readily accessible to security non-experts due to limited understanding regarding the applicability of the available information in the context of a given system. Providing additional guidance on how and when to leverage the

available security information in the context of the given system by systematically developing security requirements patterns from existing knowledge sources can support the security requirements engineering efforts. A significant upfront cost in terms of time and effort is needed to specify security requirements patterns in a manner that makes them useful in the long run. However, a security requirements pattern specification that facilitates operationalization of security requirements in the context of multiple systems can lead to overall reduction in time and effort required for security requirements engineering. Security requirements patterns can help minimize the ratio between upfront cost of identifying common and recurring requirements versus delayed gratification once the requirements patterns have been successfully developed, applied and reused.

We propose a systematic process for developing security requirements patterns from existing security requirements knowledge sources by cataloging the security controls in terms of security goals that the controls support. Our patterns cover a comprehensive set of security properties while considering the prevention of, detection of and response to security breaches. The contextual information provided as part of the patterns can support the analyst in applying the appropriate patterns in the context of a software system. Our security requirements patterns enable an analyst to operationalize security goals of a software system by supporting integration of security requirements related to a comprehensive set of security goal patterns and implied goals. Our work has extended the scope of security requirements patterns to support security requirements engineering for a diverse set of systems and security goals. We have also streamlined pattern representation based on considerations identified from related literature to facilitate pattern applicability and usage.

## 7 IDENTIFICATION OF SECURITY PROPERTIES IMPLIED BY SENTENCES IN NATURAL LANGUAGE ARTIFACTS

In this chapter, we address the following research questions:

- *RQ2: What features in the natural language requirements artifacts can help in the identification of security properties that are implied by individual sentences?*
- *RQ3: How effectively can we automate the process of identifying security properties that are implied by individual sentences in the natural language requirements artifacts?*

We present our process for automatic identification of security properties implied by sentences in an input set of natural language requirements artifacts. We incorporate supervised machine learning techniques to classify the sentences in the input in terms of their security properties (such as, confidentiality, integrity, availability). The output classifications are used to suggest security requirements patterns that support the identified security properties. We also evaluate the effectiveness of our process in automatically identifying security properties through case studies of requirements artifacts from healthcare and networks domain. Our classification results can guide an analyst in creating an appropriate set of security requirements and in organizing the resultant set of security requirements.

## 7.1 Methodology

We have developed a tool-assisted process<sup>16</sup> for identifying natural language sentences that have security implications. Our tool takes natural language requirements artifacts (requirement specifications, feature requests, etc.) as input. If a labeled corpus of sentences with implied security properties for the current problem domain is available, we also provide the labeled domain corpus as input. Otherwise, our process provides steps for creating the labeled domain corpus. The tool parses the artifacts as individual text sentences and identifies which (if any) security properties relate to each sentence. We explain the steps of the process below.

### 7.1.1 Step 1: Pre-process Input Artifacts

We convert the input artifact into a text document as a first step. Our tool for automatic identification of security properties accepts text only format as input since we leverage natural language processing and text classification in our process. Generally, the conversion can be accomplished through the “Save As” format within Microsoft Word or other document applications. As such, this process will not convert tables or images properly and the analyst will need to manually perform the conversion for those sections. Once the artifact has been prepared, the tool will first read the entire text into the system.

We can parse individual elements in the input document for text classification. We use a sentence as the unit of classification since we are working with requirements artifacts.

---

<sup>16</sup> Source code available at: <http://go.ncsu.edu/securitydiscoverer/>

However, in some cases, we may want to use a set of sentences describing a single concept as unit of analysis.

To identify individual sentences and provide additional context about the sentences, the tool applies a concise document grammar to label each sentence in the text to a specific type:

- *title*: Sentences that follow capitalization rules for titles.
- *list start*: These sentences represent the header or description of a list that follows.
- *list element*: These sentences represent individual items contained within an ordered or unordered list. These sentences are combined with the start of the list when sent to the parser and for classification. Combining the two provides additional context to both human analysts and machine classifiers.
- *normal sentence*: These sentences are not considered as titles, list starts, or list elements.

Further, we identify heading and list identifiers (e.g., “4.1.1” and “•”) and remove those identifiers from sentences used in classification. These identifiers create superficial differences among sentences and can possibly skew classification results. Any identifiers missed during this step (e.g., document specific identifiers) can be removed at the time of feature selection as well. While we do not expect the documents to be well-formed, our process works better with shorter, well-formed sentences.

### **7.1.2 Step 2: Develop a Domain Corpus**

A domain corpus is a set of labeled sentences or documents that have already been classified in terms of the applicable security properties. If a domain corpus is already available, we skip to the next step in the process.

The creation of domain corpus for the first time is a manual activity. To minimize subjectivity in the process, we created and iteratively refined a classification guide, as shown in Table 26, to specify when each security property is indicated. In general, the actions on the assets are indicative of the various security properties as discussed in Section 5.1.3.

Table 26. Classification Guide – Manual Classification

<b>Confidentiality (C)</b>	<ul style="list-style-type: none"> <li>• presence of read-type verbs (read, view, display, share, alert, use, export, retrieve, report etc.) when accessing a sensitive resource</li> <li>• specification of securing or restricting access to a sensitive resource</li> <li>• specification of user roles and access rules (e.g., granting or revoking access)</li> </ul>
<b>Integrity (I)</b>	<ul style="list-style-type: none"> <li>• presence of write-type verbs (create, update, delete, save, auto-populate, review, verify, generate, define, apply etc.) when accessing a sensitive resource</li> <li>• need for accuracy, consistent understanding and interpretation of sensitive information</li> <li>• actions involving auto-population, merging, or partitioning of sensitive information</li> <li>• environmental checks (firewall, denial of service, virus checking, app firewalls)</li> </ul>
<b>Availability (A)</b>	<ul style="list-style-type: none"> <li>• time or location constraints on information (e.g., real-time information)</li> <li>• data retention, historical or longitudinal data (available over a period of time)</li> <li>• checks and limits on resource utilization</li> <li>• backups, replication or archiving of resources</li> </ul>
<b>Identification &amp; Authentication (ID)</b>	<ul style="list-style-type: none"> <li>• need to identify / verify the entity (e.g., user or system) accessing a sensitive resource</li> <li>• need to identify / verify the source of data (e.g., patient originated data vs. clinic originated data)</li> </ul>
<b>Accountability (AY)</b>	<ul style="list-style-type: none"> <li>• recording of actions (e.g., accessing a resource) related to sensitive resources</li> <li>• recording of events (e.g., alerts and notifications) related to sensitive resources</li> <li>• ability to reconstruct history of actions and events</li> <li>• need to prevent an actor from denying they performed an action</li> <li>• transactions of data (create, read, update, delete, merge, unmerge, assign etc.)</li> </ul>
<b>Privacy (PR)</b>	<ul style="list-style-type: none"> <li>• ability of an individual to exercise control over the collection, use and dissemination of his or her personally identifiable information (e.g., by providing consent)</li> <li>• gathering, usage, disclosure, and management of user or client data</li> <li>• masking of the identity of specific users or individuals</li> <li>• legal and compliance issues related to personally identifiable information</li> </ul>

We employ multiple researchers to manually read each natural language sentence in the input documents, and classify the sentence with relevant security properties as follows:

- 1) Convert the document into text-only format.
- 2) Import one text document into the tool, and parse the document into individual sentences using natural language processing (Section 7.1.1).
- 3) Manually classify each sentence in terms of the security properties that are implied by that sentence.
  - a. *Classification Phase*: For each document, two researchers individually classify each sentence to identify security properties that apply to the sentence in accordance with the classification guide. In the beginning, the researchers may discuss any ambiguities and iteratively update the classification guide if needed. The classification phase results in the creation of two separate output files (one per researcher) for each input document.
  - b. *Validation Phase*: We generate a difference report from the two initial classifications. The researchers discuss to resolve the differences or involve an additional researcher to generate consensus for creating a final, consolidated classified corpus document. We also document inter-rater agreement based on the individual classifications to assess reliability of the classification process.

We classify each sentence in the input into zero or more security properties. The classifier can be created in one of three ways: 1) training a new classifier by manually classifying sentences for security properties from related projects; 2) utilizing an existing classifier; or 3) utilizing the tool in an interactive fashion to provide recommendations for classifications to aid the manual process.

### 7.1.3 Step 3: Predict Security Properties

We use supervised machine learning algorithms for predicting security properties for the input requirements artifacts based on the corresponding domain classifier. Following steps are involved in the text classification process:

- 1) Convert the input text into feature vectors.
- 2) Select features for classification.
- 3) Train a machine learning model based on the selected features to predict security properties.

Depending on the machine learning algorithm used for classification, we would convert the text into different feature vectors. We have utilized three different classifiers for supervised machine learning including a  $k$ -NN classifier, Naïve Bayes classifier and Sequential Minimum Optimization (SMO) classifier for training Support Vector Machines (SVM).

For  $k$ -NN classifier, we look at the  $k$  nearest neighbors of the sentence that we want to classify. To determine the closest sentence(s), we apply a custom distance function based upon a modified version of Levenshtein distance (Levenshtein 1966) developed by Slankas et al. (Slankas et al. 2014). The distance value acts as the feature used for classification.

For Naïve Bayes and SVM classifiers, we use the Weka (Hall et al. 2009) implementation of the algorithms. We convert the input sentence to word vectors (Joachims 1998) which serve as the features for classification. We select the word vectors that are most useful for predicting a specific security property based on the associated information gain values (Quinlan 1986). An important consideration for making predictions based on text documents

is that different text documents may have different vocabulary. When we use word vectors as features for classification (e.g., in SVM classifier), we need to create word vectors based on the vocabulary from the documents in the domain classifier (training set, for which the classification is known) as well as the input requirements artifacts that we need to predict (test set, for which the classification is predicted). We use batch-processing to simultaneously create the vocabulary based on the sentences in the training and test sets.

## **7.2 A Case Study in Healthcare Domain**

In this section, we discuss the application of our methodology on a set of documents for the healthcare domain. We document the process for collecting and preparing the selected documents for use within our study. We use the following research questions to guide our analysis:

***RQ2.1:** How often are security properties explicitly stated or implied in natural language requirements artifacts?*

***RQ2.2:** What similarities (words, phrases, grammatical structure, etc.) exist among sentences that imply a particular security property?*

***RQ3.1:** How effectively can security properties be identified and extracted from natural language project documents?*

***RQ3.2:** Are there factors that potentially influence the classifier performance in identifying security properties?*

### **7.2.1 Study Documents**

Security is an important consideration in a number of domains, including healthcare. For the purpose of this study, we have selected project documentation from healthcare systems,

standards, and best practices primarily used in two different countries (United States and Canada) to increase the generalizability of our findings within the healthcare domain. To include a variety of document types in our study, we select the following six freely-available healthcare documents for our study:

- Certification Commission for Healthcare Information Technology (CCHIT)<sup>17</sup> Ambulatory Certification Criteria – a set of standards/certification criteria that outlines functional requirements of ambulatory EHR systems
- Emergency Department Information Systems Functional Document<sup>18</sup> – a set of functional data standards and conformance criteria provided by Health Level 7 International for EHR systems
- Pan-Canadian Nursing EHR Business and Functional Elements Supporting Clinical Practice from the Canadian Health Infoway<sup>19</sup> – set of privacy and security requirements for Canadian EHR systems
- Open Source Clinical Application Resource (OSCAR) Feature Requests<sup>20</sup> – a set of informal descriptions of requests for additional functionality of the Canadian OSCAR EHR system, submitted by stakeholders (including users)
- Canada Health Infoway Electronic Health Record (EHR) Privacy and Security Requirements from the Canadian Health Infoway<sup>21</sup> – set of privacy and security requirements for Canadian EHR systems

---

<sup>17</sup> <https://www.cchit.org/>

<sup>18</sup> <http://www.hl7.org/>

<sup>19</sup> <https://www.infoway-inforoute.ca/>

<sup>20</sup> <http://oscarcanada.org/>

- Virtual Lifetime Electronic Records (VLER) user stories<sup>22</sup> – a set of functional requirements for the United States Department of Veteran’s Affairs VLER technology initiative

The breakdown of sentences in each document is listed in Table 27. The artifacts contained a total of 10,963 sentences. Due to regulation and standardization, documents in healthcare domain generally tend to be well-formed. However, we select a variety of document types, including feature requests that are not well-formed. The selected documents are from USA and Canada. Use of different spellings for same words (e.g., color vs. colour) may also affect the performance of our classifier.

### **7.2.2 Domain Corpus**

We developed a corpus of labeled requirements sentences for the healthcare domain by manually classifying the 10,963 sentences in the selected documents in terms of the security properties that are implied by those sentences. We used the domain corpus to train our tool and evaluate the performance of our classifier. The researchers spent a total of approximately 160 person-hours to create and validate the domain corpus that we use for further analysis. Researchers had a moderate agreement (Koch 1977) on whether a sentence was related to security or not (indicated by a kappa score of 0.54). Of the sentences that were related to security, we had an almost perfect agreement in terms of whether a sentence explicitly talks about security or implies a need for security (kappa score of 0.85). We also had a fair agreement on classification of properties for each sentence (kappa score of 0.32; kappa score

---

<sup>21</sup> <https://www.infoway-inforoute.ca/>

<sup>22</sup> <http://www.va.gov/vler/>

tends to decrease as classification categories increase). Table 27 provides a document-wise breakdown of sentences classified per security property. Each sentence could be classified in terms of zero or more security properties.

Table 27. Healthcare Documents and Associated Security Property Counts

Doc. ID	Document Title	#Sentences	Security Properties						
			C	I	ID	A	AY	PR	None
CT	Certification Commission for Healthcare Information Technology (CCHIT) Certified 2011 Ambulatory EHR Criteria	331	252	214	19	14	260	5	6
ED	Emergency Department Information Systems Functional Document	2328	1162	1173	75	35	1354	76	773
NU	Pan-Canadian Nursing EHR Business and Functional Elements Supporting Clinical Practice	264	67	77	4	26	43	10	96
OR	Open Source Clinical Application Resource (OSCAR) Feature Requests	5081	696	974	104	10	1184	18	3735
PS	Canada Health Infoway Electronic Health Record (EHR) Privacy and Security Requirements	1623	146	120	43	31	149	85	928
VL	Virtual Lifetime Electronic Record User Stories	1336	693	731	13	19	797	10	375
<b>Total # (%)</b> :		10963	3016 (27%)	3289 (30%)	258 (~2%)	135 (~1%)	3787 (34%)	204 (2%)	5913 (54%)

### 7.2.3 Breakdown of Security Properties

*RQ2.1: How often are security properties explicitly stated or implied in natural language requirements artifacts?*

Based on the domain classifier, we identified that 46% of the sentences in input artifacts relate to security. Given that we selected documents from industry standards and best practices related to the healthcare domain (which involves protected health information), sentences related to security intuitively form a large proportion of the document. Of all the sentences related to security, only 28% explicitly mention security (13% of total sentences, similar to earlier findings (Slankas and Williams 2013)), while 72% are functional requirements with security implication (an additional 33% of total sentences). If these implied security properties are not considered, requirements engineers may overlook key

security requirements. Table 28 provides a document-wise breakdown of sentences and whether security properties were implied or explicitly stated.

Table 28. Breakdown of Sentences with Explicit or Implicit Security Requirements – Healthcare

Doc ID	Total Sentences	Explicit Security # (%)	Implicit Security # (%)	Total Security #(%)	Not Security Related # (%)
CT	331	89 (27%)	236 (71%)	325 (98%)	6 (2%)
ED	2328	274 (12%)	1281 (55%)	1555 (67%)	773 (33%)
NU	264	41 (16%)	127 (48%)	264 (64%)	96 (36%)
OR	5081	174 (3%)	1172 (23%)	1346 (26%)	3735 (74%)
PS	1623	628 (39%)	67 (4%)	695 (43%)	928 (57%)
VL	1336	185 (14%)	776 (58%)	961 (72%)	375 (28%)
<b>Total</b>	<b>10963</b>	<b>1391 (13%)</b>	<b>3659 (33%)</b>	<b>5050 (46%)</b>	<b>5913 (54%)</b>

From the sentences that are implicitly related to security, we identified the security properties that are implied by each sentence (Table 27). We observed variations among the six document in terms of the breakdown of security properties. Overall, the top three implied security properties are accountability (34% of all sentences), integrity (30%) and confidentiality (27%). Privacy (2%), identification & authentication (~2%), and availability (~1%) properties were implied by only a small percentage of all sentences. Our results indicate that 93% of the sentences that implied a security property implied more than one security property.

Table 29 presents the 10 most frequently occurring security property groups. Confidentiality and accountability each appear in 7 of 10 top property groups, suggesting that confidentiality and accountability are common security properties for healthcare systems. Integrity appears in 6 of 10 top property groupings.

The confidentiality, integrity, and accountability properties appear together in the classifications of 2,232 sentences (20% of all sentences classified), suggesting a strong

relationship among the three. For example, the sentence “The system shall provide a means to edit discharge instructions for a particular patient” [ED] implies that the confidentiality of discharge instructions should be maintained since it is protected health information; that the integrity of the discharge instruction data upon editing should be maintained; and that accountability should ensure that the user editing the discharge instructions can be held responsible.

Table 29. Frequently Occurring Property Groups – Healthcare

Frequency # (% sec-relevant)	Property Group
2232 (44%)	Confidentiality, Integrity, Accountability
702 (14%)	Integrity, Accountability
443 (9%)	Confidentiality, Accountability
106 (2%)	Confidentiality, Integrity
104 (2%)	Confidentiality, Identification & Authentication
98 (2%)	Confidentiality, Accountability, Privacy
95 (~2%)	Integrity, Accountability, Privacy
90 (~2%)	Integrity, Identification & Authentication, Accountability
86 (~2%)	Confidentiality, Identification & Authentication, Accountability
83 (~2%)	Confidentiality, Integrity, Privacy

Confidentiality and accountability appear together in the classifications of 2,859 sentences (26% of all sentences classified). The act of controlling access to sensitive data to help promote confidentiality is closely tied to the act of ensuring that a complete list of users who have accessed the sensitive data may be maintained for accountability. Therefore, in our study oracle, sentences that involve create/read/update/delete actions upon sensitive data are often classified as implying both confidentiality and accountability. Integrity and accountability appear together for 3,119 sentences (28.5% of all sentences classified). With respect to accountability, integrity helps ensure that the traces of user activity in the system

may not be corrupted, modified, or damaged so that users can always be held accountable. Privacy and identification/authentication properties also appear in the top ten property groupings, but are much less common. Privacy and identification/authentication often appear in combination with confidentiality, integrity, and/or accountability properties.

#### **7.2.4 Similarities Among Sentences**

*RQ2.2: What similarities (words, phrases, grammatical structure, etc.) exist among sentences that imply a particular security property?*

Overall, keywords are the primary indicator of security properties for identification/authentication, availability, and privacy. However, for many confidentiality, integrity, and accountability sentences, the grammatical structure of the sentence is often the same and included keywords that correspond to specific types of actions (e.g., display, create).

Table 30 presents the top 20 keywords listed for security property. To extract the top 20 keywords for each security property, we utilized the information gain (Quinlan 1986) attribute selector within Weka. Yang and Pedersen (Y. Yang and Pedersen 1997) found information gain to be the most effective method for feature selection in text classification. The set of keywords is very similar for confidentiality, integrity, and accountability properties. This suggests a noticeable relationship among confidentiality, integrity, and accountability properties. The top keywords also indicate the set of actions that are associated with each security property. For instance, read-type actions (list, display) and transfer-type actions (send) are associated with confidentiality. We also notice the presence of write-type words (create, generate) for sentences related to integrity.

Table 30. Top 20 Keywords by Security Properties – Healthcare Documents Study

Property	Keywords
Confidentiality	<i>system, provide, ability, patient, result, vler, exam, capture, datum, record, send, display, medication, information, list, requirement, status, consuming, order, complete</i>
Integrity	<i>system, provide, ability, vler, exam, send, capture, result, datum, store, consuming, patient, pass, click, pick-list, status, application, element, create, generate</i>
Availability	<i>run, availability, datum, retain, time, year, nurse, destroy, application, legally, recent, retention, care, maximum, real-time, information, period, destruction, record, historical</i>
Identification & Authentication	<i>authentication, login, mac2002, username, oscar, user, authenticate, identify, cash, identity, myoscar, password, waitlist, log, registration, list2012, regen, uniquely, credentials, valid</i>
Accountability	<i>system, ability, provide, vler, exam, result, send, consuming, click, pass, patient, capture, pick-list, datum, application, audit, status, store, record, list</i>
Privacy	<i>consent, patient, person, phi, disclosure, purpose, privacy, directive, require, organization, ehrus, law, authorization, information, connect, disclose, healthcare, inform, jurisdiction, collect</i>

Keywords “system”, “provide”, and “ability” commonly appear in sentences classified as confidentiality, integrity, and/or accountability. Sentences classified as confidentiality, integrity, and/or accountability often appear in the form: “The system shall provide the ability to <action> <resource>”. For example, “The system should provide the ability to check medications against a list of drugs noted to be ineffective for the patient in the past” [ED]. Since the resource in the example sentence involves access to medications (protected health information), the sentence is classified as implying a confidentiality property. Likewise, since the sentence involves interacting with protected information, the integrity of the data must be maintained. Finally, since the sentence involves a user accessing protected information, the system should keep track of all users who have accessed the data so that they may be held accountable.

For identification/authentication, top keywords include, “authentication”, “login”, “username”, “user”, “authenticate”, and “identify”. While the structure of sentences for confidentiality, integrity, and accountability share a common grammatical pattern, sentences

for identification/authentication share only common keywords that suggest the need to know the identity of a user, or the need to ensure that a user has authenticated into the system so that they can be identified by unique credentials.

Similarly, top keywords for availability include “run”, “availability”, “retain”, “time”, “destroy”, “retention”, and “real-time”. Like identification/authentication, no grammatical pattern exists for availability. Instead, keywords that suggest temporal or data retention/destruction obligations are strong indicators of the presence of an availability security property.

Top keywords for privacy include “consent”, “phi”, “disclosure”, “purpose”, and “privacy”. Again, no grammatical pattern exists in the classified sentences for this property. Instead, common keywords that suggest privacy property include terms that involve a user (patients, in healthcare documents) choosing to give consent, or disclosure of protected information to anyone other than the patient. Disclosure of protected information suggests that a user has consented to disclose given information to a third-party.

### **7.2.5 Effectiveness of Automatically Identifying Security Properties in Natural Language Text**

*RQ3.1: How effectively can security properties be identified and extracted from selected set of documents?*

Once the domain classifier has been created manually, we execute a variety of classifiers (k-NN classifier with custom distance function and from Weka (Hall et al. 2009) – a multinomial naïve Bayes classifier, and a SMO – sequential minimal optimization classifier)

on the document set. We use recall and precision as measures to assess effectiveness of the classifier performance (Section 2.3.2).

In Table 31, we present the results of running the three individual classifiers against the six documents using a ten-fold cross validation. We also created a “Combined” ensemble classifier that uses the results of the k-NN classifier if relatively close sentences were found. Otherwise, the “Combined” classifier uses a majority vote of the three classifiers. The “Combined” classifier demonstrated a slight performance gain over just using the Weka SMO classifier. The k-NN classifier performed equivalently to the SMO classifier. However, the advantage of k-NN classifier comes into play with using the tool in an interactive fashion. The classifier reports the sentences closest to the current sentence under test along with the distance. This allows an analyst to view similar sentences when making choices as to the possible security properties.

Table 31. Ten-Fold Cross Validation – Performance of Classifiers for Healthcare Documents

Classifier	Precision	Recall	$F_1$ Measure
Naïve Bayes	.66	.76	.71
SMO	.81	.76	.78
$k$ -NN ( $k=1$ )	.80	.76	.78
Combined	.82	.79	.80

The reported precision of .82 implies that the tool correctly predicted 82% of all the security properties associated with the sentences it classified. The recall score of .79 means that it found 79% of all of the possible properties. From an error perspective, the precision score implies that 18% of the identified properties an analyst examines would be false positives, and 21% of the possible properties were not found.

We were able to predict the security properties based on the classifier for healthcare domain with an overall precision of 82% and recall of 79%. Based on the results, 18% of the identified properties an analyst examines would be false positives, and 21% of the possible properties were not found.

### 7.2.6 Factors Influencing Classifier Performance

*RQ3.2: Are there factors that potentially influence the classifier performance in identifying security properties?*

In the domain classifier for healthcare domain, some security properties are more prevalent than the others. The performance of the classifiers varied for each security property. For analysis of classifier performance for each security property, we selected the SMO classifier as it gave the best performance among the individual classifiers as shown in Table 31.

We predict the classifications for individual security properties using the domain classifier. The proportion of sentences with and without a particular security property is not equal in the domain classifier and hence the training data for the prediction model is unbalanced. We present a breakdown of the performance of SMO and Naïve Bayes Multinomial classifiers in predicting the individual security properties in Table 32. The properties for which we have more example sentences in the domain classifier (i.e., confidentiality, integrity, and accountability) have higher values for precision, recall and  $F_1$  measure as compared to the properties for which we have fewer example sentences in the domain classifier (availability, identification & authentication, and privacy).

Table 32. Classifier Performance by Security Properties for Healthcare Documents  
–Unbalanced (Original) Training Set

Security Property	Sentences # (%)	Precision	Recall	$F_1$ Measure	Precision	Recall	$F_1$ Measure
		SMO			Naïve Bayes		
Confidentiality (C)	3016 (27%)	.81	.75	.78	.69	.79	.74
Integrity (I)	3289 (30%)	.83	.77	.80	.70	.79	.74
Availability (A)	135 (~1%)	.57	.30	.39	.06	.80	.11
Identification & Authentication (ID)	258 (~2%)	.74	.56	.65	.19	.85	.31
Accountability (AY)	3787 (34%)	.87	.82	.85	.77	.84	.80
Privacy (PR)	204 (2%)	.56	.40	.46	.13	.82	.23

Given that having fewer instances in the training data can impact the classifier’s performance, we employed class balancing such that sentences with and without a security property have equal weights in the training set. We wanted to assess if balancing the training sets will improve the classifier’s performance, specifically for the security properties of availability, identification & authentication, and privacy which have a very small percentage of sentences in the training set. The results are given in Table 33. The precision and recall improved for all the security properties after balancing the classes as compared to the values without balanced classes. Class balancing improved the performance of the classifier indicating that less skewed training set (i.e., where number of sentences classified as having a particular security property is proportional to sentences without the security property) is better at predicting the classifications as compared to an unbalanced training set.

Table 33. Classifier Performance by Security Properties for Healthcare Documents  
– Balanced Training Sets

Security Property	Sentences # (%)	Precision	Recall	$F_1$ Measure	Precision	Recall	$F_1$ Measure
		SMO			Naïve Bayes		
Confidentiality (C)	3016 (27%)	.87	.83	.85	.63	.94	.76
Integrity (I)	3289 (30%)	.87	.85	.86	.65	.93	.77
Availability (A)	135 (~1%)	.95	.47	.63	.67	.56	.61
Identification & Authentication (ID)	258 (~2%)	.98	.71	.82	.76	.82	.79
Accountability (AY)	3787 (34%)	.90	.87	.89	.65	.94	.77
Privacy (PR)	204 (2%)	.96	.55	.70	.69	.74	.71

In Figure 17, we provide the proportion of sentences for each security property (given in parentheses) in the unbalanced training set (domain classifier) and the respective values of precision, recall and  $F_1$  measure before and after balancing the training sets. The improvement in precision was more pronounced for security properties that were more skewed in the unbalanced training set (had the least proportion). For SMO classifier, balanced training sets improved the classifier’s performance across all the three metrics of precision, recall and  $F_1$  measure. However, for the Naïve Bayes classifier, using a balanced training set led to slightly lower precision for properties that are more prevalent and slightly lower recall for properties that are less prevalent. Looking at  $F_1$  measure (giving equal weight to precision and recall), the balanced training set performed better for the Naïve Bayes classifier as well. Overall, using a balanced training set with SMO classifier gave the best performance in predicting the security properties. The performance of balanced Naïve Bayes classifier was slightly lower than SMO for the properties that are more prevalent and

comparable for properties that are less prevalent. Our results indicate that SMO's performance improves with more examples.

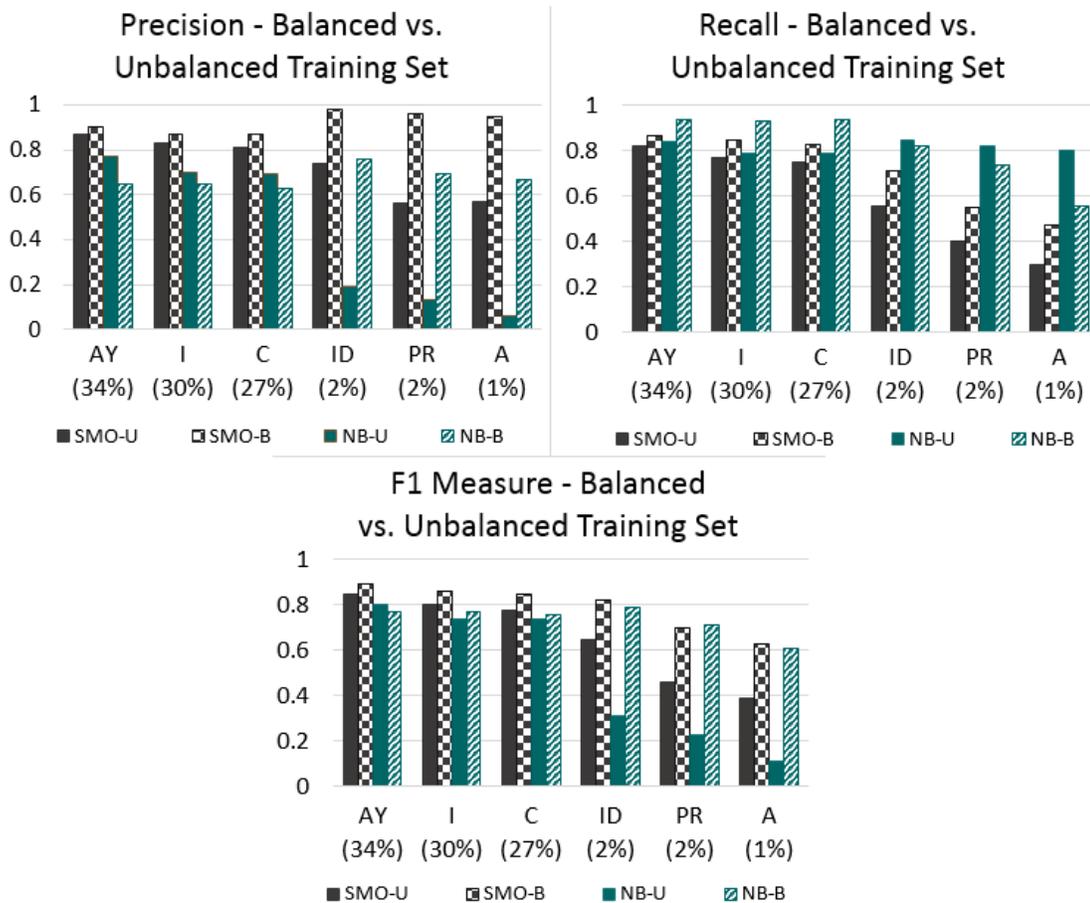


Figure 17. Classifier Performance for Healthcare Documents – Balanced (B) vs. Unbalanced (U) Training Sets

The performance of classifier for predicting security properties varied based on the number of sentences in the input that implied the respective properties. The security properties with the most sentences in the input were also predicted most effectively. Balancing the number of sentences in the training set for security properties tend to improve the classification performance.

### 7.3 Generalizability of Approach to Networks Domain

We have applied our process to multiple documents in the healthcare domain. The results indicate that supervised machine learning can be used to predict the security properties in natural language requirements artifacts with acceptable performance.

To assess the generalizability of our findings to other domains, we selected a random set of 356 sentences from a natural language requirements document created for a networking product by Cisco Systems, Inc. We address the following additional research questions.

***RQ3.3:** How effectively can security properties be identified and extracted from natural language project documents using domain classifier for a different domain?*

#### 7.3.1 Domain Corpus

We developed a corpus of labeled requirements sentences for the networking domain using a set of 356 sentences randomly selected from a total of 1218 sentences in a natural language requirements document for a networking product. We created the domain corpus incrementally, by adding more sentences to the training set to achieve an acceptable level of precision and recall values (Overall  $F_1$  measure  $\geq 0.7$ ). The size of the various training sets and breakdown of security properties in each training set is given in Table 35. For creating the domain corpus, two raters individually assigned the security properties to the sentences and consolidated any differences through discussion. Researchers had good agreement (Koch 1977) on whether a sentence implied a need for security or not (indicated by a kappa score of 0.67). Of the sentences that implied a need for security, we had moderate to good agreement on various security properties (kappa score between 0.52 and 0.8). The agreement was higher

for security properties that were more prevalent in the training set (availability) than those which were less prevalent (confidentiality and privacy).

### **7.3.2 Breakdown of Security Properties**

The breakdown of security properties implied by sentences in the requirements artifacts for healthcare and networks domain is given in Table 27 and Table 35 respectively. In the healthcare domain, we observed that the six selected documents differed in terms of the proportion of various security properties, even within the same domain. However, confidentiality, integrity and accountability were the predominantly implied security properties across the documents. Whereas, for the sentences analyzed from the networks domain, integrity and availability are the most prevalent security properties, as shown in Figure 18. Based on the figure, we see that the inter-domain differences between documents are more pronounced as compared to intra-domain differences. For instance, sentences implying a need for availability are the most prevalent in the document from the networks domain whereas sentences implying a need for availability are the least prevalent in all the documents related to the healthcare domain. Moreover, sentences implying a need for confidentiality are among the least prevalent in the document from the networks domain whereas sentences implying a need for confidentiality are among the most prevalent across all documents in the healthcare domain. However, analysis of further documents from networks is needed to assess if the findings generalize to other documents.

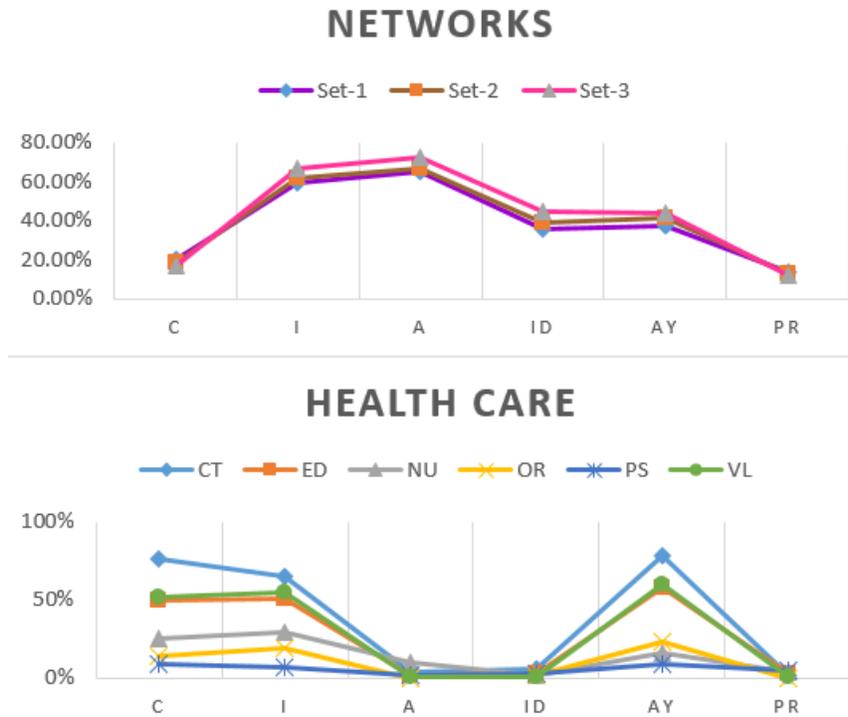


Figure 18. Percentage of Sentences Implying each Security Property – Artifacts from Networks and Healthcare Domains

One of the reasons for lower percentage of sentences with confidentiality and privacy implications in networks domain is that many sentences talked about protocols that operate on the layers where security and privacy concerns are not addressed. Whereas in the healthcare domain, the requirements were mainly for the application layer where such concerns are handled. Moreover, availability of networking infrastructure is a primary concern in networks domain and many sentences implied the need for availability. Whereas for healthcare domain, while availability of services is important, it is not a primary concern implied by a large set of sentences.

### 7.3.3 Similarities Among Sentences

We provide a list of top 20 keywords, ranked based on the information gain value, for each of the security property based on the analysis of sentences from the networks domain in Table 34. A large number of domain specific terminology and protocols are used by the machine learning algorithms to predict the security properties. Properties of integrity and availability often appeared together and the keywords for these properties are also very similar.

Table 34. Top 20 Keywords by Security Properties – Networks Documents Study

Security Property	Keywords
Confidentiality	COGS, security, QoS, metro, available, case, guard, ethernet, ACLs, EMC, mandatory, run, bundles, burden, CDO, CDP, cabinet, CBWFQ, BTS, cellular, 100base, 10g
Integrity	VLAN, note, port, requirement, aperture, price, features, update, switching, 10ge, MAC, depth, competitive, OAM, new, more, COGS, mapping, EVC, layer, routing, transition
Availability	Requirement, VLAN, note, aperture, price, new, depth, product, switching, clock, SFP, wire, 10ge, MAC, multiple, local, mapping, EVC, MPLS, layer, dimension
Identification & Authentication	SFP, MAC, layer, MPLS, front, requirement, 802, 10ge, power, VLAN, port, QinQ, switching, require, COGS, external, protocol, hardware, FCS, OAM, QoS, BFD, hierarchical
Accountability	Switching, SFP, MPLS, front, 802, clock, SFP, 10ge, QinQ, require, COGS, lag, log, hardware, CFM, scheduling, traffic, DSCP, OAM, QoS, OBFL, hierarchical, IGMP, security
Privacy	Security, management, EMC, mandatory, guard, ACLs, schematics, scenarios, change, save, linerate, series, CISPR, CEOPS, catalyst, cellsites, marking, match, research, safety, several

### 7.3.4 Cross-Validation to Analyze Performance of Classifier

We provide results of 10-fold cross-validation for the various training sets in Table 35. We used the SMO and Naïve Bayes multinomial classifiers for the cross-validation. SMO gave the best performance among the classifiers with an overall classification accuracy of around 78%. The precision and recall values increased with the size of the training set indicating that creating a larger domain classifier will lead to improved prediction performance.

Table 35. 10-Fold Cross-Validation of Domain Classifier (Networks)

Security Property	Sentences # (%)	Precision	Recall	F <sub>1</sub> Measure	Accuracy
<b>Set-1 (281 sentences) – SMO Classifier</b>					
C	57 (20%)	0.54	0.25	0.34	80.43%
I	165 (59%)	0.74	0.64	0.69	65.84%
A	183 (65%)	0.80	0.71	0.75	69.40%
ID	102 (36%)	0.68	0.44	0.54	72.24%
AY	105 (37%)	0.57	0.41	0.48	66.55%
PR	40 (14%)	0.46	0.15	0.23	85.41%
	<b>Overall</b>	<b>0.71</b>	<b>0.53</b>	<b>0.60</b>	<b>73.31%</b>
<b>Set-2 (305 sentences) – SMO Classifier</b>					
C	58 (19%)	0.62	0.28	0.38	82.95%
I	188 (62%)	0.74	0.69	0.72	66.23%
A	204 (67%)	0.77	0.79	0.78	69.84%
ID	119 (39%)	0.76	0.50	0.60	74.10%
AY	125 (41%)	0.67	0.45	0.54	68.20%
PR	40 (13%)	0.42	0.13	0.19	86.23%
	<b>Overall</b>	<b>0.73</b>	<b>0.58</b>	<b>0.65</b>	<b>74.59%</b>
<b>Set-3 (356 sentences) – SMO Classifier</b>					
C	62 (17%)	0.50	0.19	0.28	82.58%
I	239 (67%)	0.80	0.83	0.82	75.28%
A	255 (72%)	0.82	0.86	0.84	76.97%
ID	159 (45%)	0.77	0.59	0.67	74.44%
AY	158 (44%)	0.65	0.55	0.60	67.42%
PR	41 (12%)	0.50	0.12	0.20	88.48%
	<b>Overall</b>	<b>0.77</b>	<b>0.67</b>	<b>0.72</b>	<b>77.53%</b>
<b>Set-3 (356 sentences) – Naïve Bayes Multinomial Classifier – default settings</b>					
C	62 (17%)	0.28	0.48	0.36	69.7%
I	239 (67%)	0.81	0.81	0.81	74.7%
A	255 (72%)	0.83	0.78	0.80	73%
ID	159 (45%)	0.64	0.75	0.69	70.2%
AY	158 (44%)	0.65	0.73	0.69	71%
PR	41 (12%)	0.11	0.51	0.18	45.5%
	<b>Overall</b>	<b>0.59</b>	<b>0.74</b>	<b>0.66</b>	<b>67.37%</b>
<b>Set-3 (356 sentences) – Naïve Bayes Multinomial Classifier – custom settings</b>					
C	62 (17%)	0.31	0.53	0.39	71%
I	239 (67%)	0.81	0.78	0.80	73.9%
A	255 (72%)	0.85	0.78	0.81	74.7%
ID	159 (45%)	0.64	0.79	0.71	71.63%
AY	158 (44%)	0.67	0.76	0.72	73.60%
PR	41 (12%)	0.15	0.59	0.24	58.43%
	<b>Overall</b>	<b>0.62</b>	<b>0.75</b>	<b>0.68</b>	<b>70.54%</b>

We also provide the results of Naïve Bayes multinomial classifier using both the default Weka settings and after custom settings (stemming, stop words, and word frequency) for the largest training set in Table 35.

SMO performed the best in predicting security properties that appeared more frequently in the training set (I and A). Whereas Naïve Bayes performed better than SMO for the properties that appeared less often in the training set. We have highlighted the corresponding rows in Table 35. Overall, SMO had 77% precision as compared to 62% precision for Naïve Bayes with custom settings. However, Naïve Bayes provided higher recall values of 75% as compared to 67% recall for SMO. SMO had higher  $F_1$  score and classification accuracy as compared to Naïve Bayes. We plot a comparison of the precision and recall values of different classifiers for each security property based on training Set-3 in Figure 19. We see similar results as earlier that balancing the sets improved classifier performance.

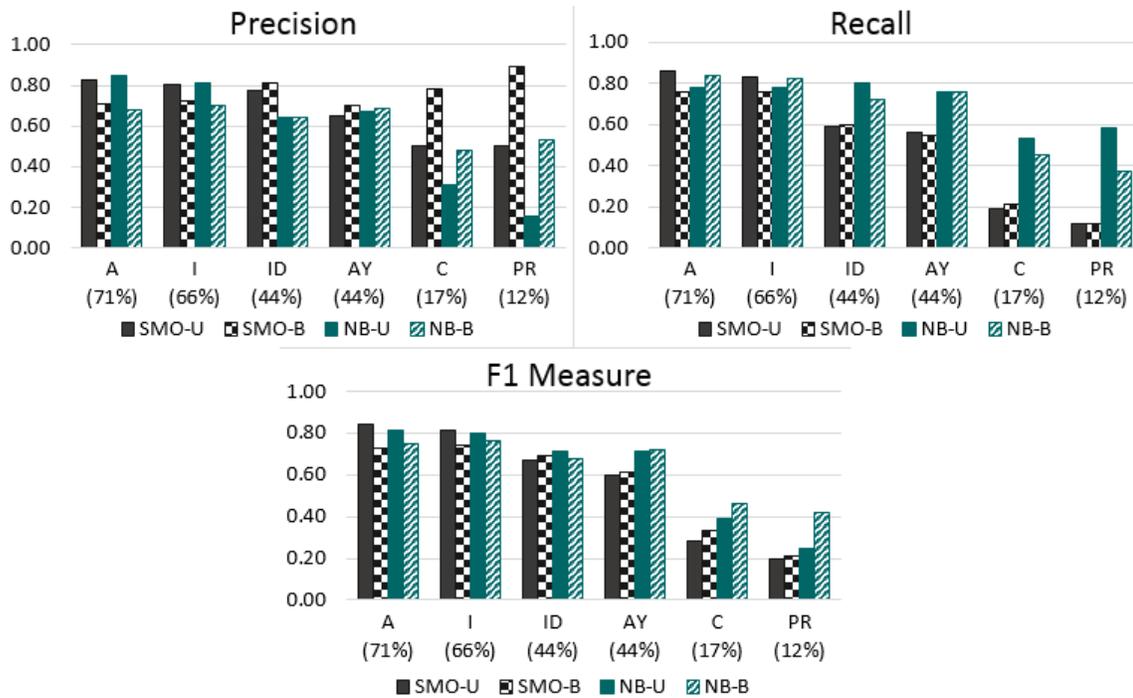


Figure 19. Classifier Performance for Networks Requirements Sentences – Balanced (B) vs. Unbalanced (U) Training Sets

We expect the recall of SMO to improve as the size of the training set increases based on the results in Table 35 and findings in Section 7.2.5. The overall precision and recall values increased with the increase in the size of the training set for SMO classifier, as shown in Table 35. We also observe the trend that precision and recall values for a security property are higher when we have a higher percentage of sentences with that property in the training set, as observed in the healthcare domain (Section 7.2.6). We plot the trends for precision and recall (SMO classifier) for individual security properties based on the size of training set and proportion of respective security properties in the training set in Figure 20.

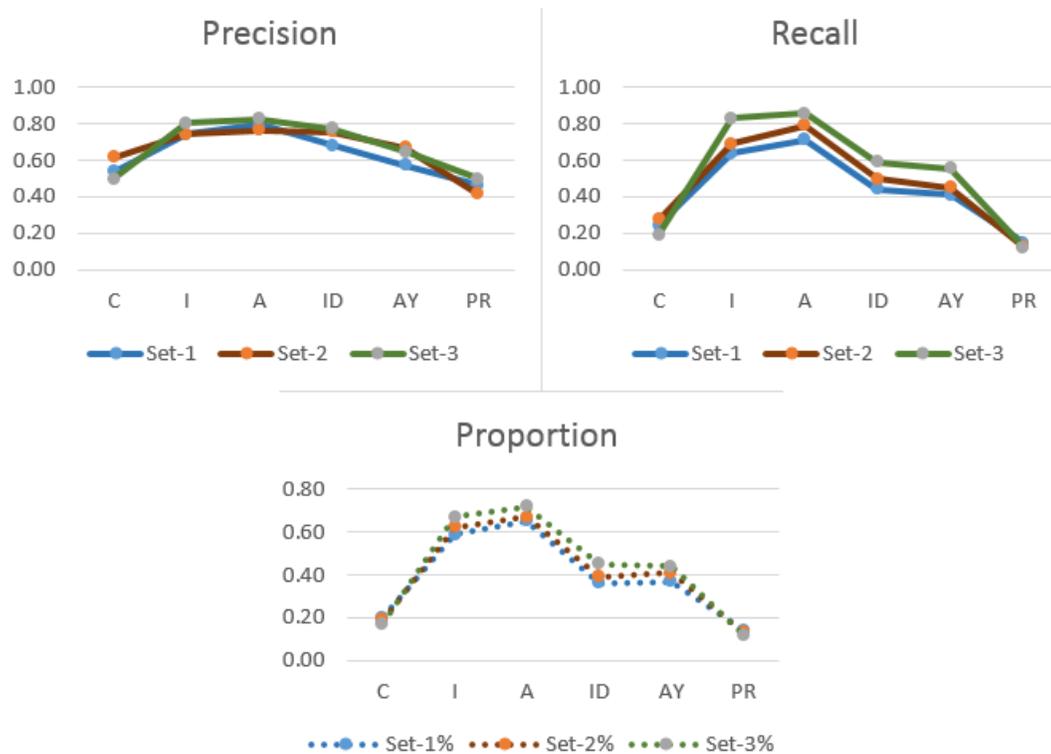


Figure 20. Precision and Recall Comparison across Different Training Sets for SMO – Networks Domain

### 7.3.5 Using Classifier from Healthcare to Predict Security Properties for Networks Domain

*RQ3.3: How effectively can security properties be identified and extracted from natural language project documents using domain classifier for a different domain?*

We used the domain classifier for healthcare domain to predict the security properties for sentences from the networks domain to see the effect on classifier performance. Based on the results in Table 36, the precision and recall values are significantly lower as compared to the values when making predictions for the sentences from the same domain. Naïve Bayes classifier performed slightly better than the SMO classifier as indicated by the highlighted rows in the table. Using a balanced dataset improved the prediction performance to some

extent for a subset of security properties. The balanced Naïve Bayes classifier gave the best performance with an overall precision of 51% and recall of only 13%. These results indicate that we would miss out on identifying 87% of the security properties implied by sentences in the networks domain by using classifier from the healthcare domain.

Table 36. Classifier Performance across Domains – Unbalanced (Un-Bal) vs. Balanced (Bal) Training Sets

			Precision	Recall	F-Measure
<b>Confidentiality</b>	<b>SVM/SMO</b>	Un-Bal	0.00	0.00	0.00
		Bal	0.20	0.03	0.06
	<b>Naïve Bayes</b>	Un-Bal	0.21	0.13	0.16
		Bal	0.21	0.13	0.16
<b>Integrity</b>	<b>SVM/SMO</b>	Un-Bal	0.75	0.03	0.05
		Bal	0.64	0.04	0.07
	<b>Naïve Bayes</b>	Un-Bal	0.75	0.16	0.27
		Bal	0.75	0.17	0.28
<b>Availability</b>	<b>SVM/SMO</b>	Un-Bal	1.00	0.00	0.01
		Bal	0.50	0.02	0.03
	<b>Naïve Bayes</b>	Un-Bal	0.72	0.05	0.10
		Bal	0.73	0.08	0.14
<b>Identification &amp; Authentication</b>	<b>SVM/SMO</b>	Un-Bal	0.00	0.00	0.00
		Bal	0.45	0.03	0.06
	<b>Naïve Bayes</b>	Un-Bal	0.47	0.06	0.10
		Bal	0.47	0.10	0.16
<b>Accountability</b>	<b>SVM/SMO</b>	Un-Bal	0.63	0.06	0.12
		Bal	0.50	0.08	0.13
	<b>Naïve Bayes</b>	Un-Bal	0.51	0.19	0.28
		Bal	0.51	0.19	0.28
<b>Privacy</b>	<b>SVM/SMO</b>	Un-Bal	0.00	0.00	0.00
		Bal	0.00	0.00	0.00
	<b>Naïve Bayes</b>	Un-Bal	0.08	0.02	0.04
		Bal	0.08	0.02	0.04
<b>OVERALL</b>	<b>SVM/SMO</b>	Un-Bal	0.59	0.02	0.04
		Bal	0.48	0.04	0.07
	<b>Naïve Bayes</b>	Un-Bal	0.50	0.11	0.18
		Bal	0.51	0.13	0.20

Based on our analysis of the two domains, the sentence structure and vocabulary across the two domains greatly varied. We see no overlap between the top keywords for various security properties across the two domains (Table 30 and Table 34). Specifically, the sentences from networks domain included a large proportion of abbreviations and protocol names. Many sentences only listed the protocol names and not much else description about the protocol and were often not even phrased as a complete sentence. To enable the cross-domain analysis, we trained the classifiers using features (word vectors) simultaneously created from both domains to create a shared vocabulary.

The spread of security properties across the two domains also varied (Figure 18) further impacting the performance of the classifiers. The input artifacts from healthcare domain contained sentences related to application level functionality of managing user's information versus network and routing functionality for the networks domain. The phrasing of sentences across the two domains also varied with more complete sentences in the healthcare domain versus more abbreviations and protocol identifiers in the networks domain. If two domains are closely related (e.g., information systems managing different type of personal information of users), we expect to see more overlap between the top keywords. However, further studies are needed to establish the cross-domain performance of classifiers in such cases.

For improved classifier performance, we may need to develop classifiers for each problem domain to leverage the vocabulary and domain specific concepts for predicting security properties. The proportion of security properties may also vary across domains further impacting classifier performance.

## 7.4 Threats to Validity

We have considered following threats to validity during our analysis of the documents from healthcare domain:

*Selection of problem domain:* Domain classifier created using documents from one domain may not be generalizable to other domains due to different security properties and domain-specific vocabulary and security concerns. Moreover, assets that need to be protected are well understood in healthcare domain that may facilitate identification of security properties implied by the sentences. Many organizations adopt data classification guides that can be used to help guide our process in other domains.

*Selection of systems and documents:* Security requirements may come from different sources (requirements documents, policy specifications, legislative texts, standards and best practices). Variations may exist between security requirements of software systems, even in the same domain. Thus, selection of documents and the problem domain may influence the type and frequency of security properties identified based on the input sentences.

*Selection of security properties:* We have compiled a list of security properties based on various taxonomies. Our list of security properties may not be complete. To minimize this threat, we have considered multiple sources from security literature to identify the properties. A general consensus on the categorization of security properties minimizes this threat.

*Subjective assessment of security properties:* To develop the domain classifier, we carried out manual classification of sentences, which can be subjective. Misclassification of sentences based on security properties in the oracle may have occurred. To minimize this concern, two researchers independently carried out the classification of each document while

a third researcher consolidated the final classification. Inter-rater reliability ranges between 0.32 to 0.85, lending validity to the process.

*Missed security properties:* We were able to identify between 67% to 79% of security properties based on the classifier predictions indicating that around 33% to 21% of the security properties will be missed by our process. The requirement documents often contain the same asset or resource in multiple sentences. Due to this built-in redundancy, the recall of security properties, when considered in terms of the assets rather than individual sentences, is expected to be much higher.

## **7.5 Summary**

We have evaluated our process on six documents from the electronic healthcare domain, identifying 46% of sentences as implicitly or explicitly related to security. Our classification approach identified security properties with a precision of .82 and recall of .79. We provide an oracle of sentences labeled with relevant security properties as a domain classifier for the healthcare domain<sup>23</sup>.

We evaluated the generalizability of our process by analyzing 356 randomly selected requirements sentences from a networking product requirements specification document. We predicted the security properties with a maximum precision of .77 and maximum recall of .75. The precision and recall values increased with the size of the training set indicating that creating a larger domain classifier will lead to improved prediction performance.

---

<sup>23</sup> <http://go.ncsu.edu/securitydiscoverer/>

The performance of the classifier was impacted by a number of factors including the proportion of security properties in the training set, size of the training set, and whether the training set was balanced or unbalanced. We recommend developing domain specific classifiers and using a balanced training set when possible. We also recommend using Naïve Bayes multinomial classifier if the size of training set or the proportion of security properties in the training set is relatively small and recall is an important consideration. Otherwise, we recommend using SMO classifier as it gives better precision overall and also better recall for relatively larger training sets.

We have demonstrated the feasibility of our process in automatically identifying security properties implied by sentences in natural language requirements artifacts, and predicting the implied security properties with acceptable precision and recall values. Our results indicate that the process is generalizable across different types of requirements sentences from different domains. Our findings also indicate that we need to develop a domain classifier for each domain of interest to get acceptable prediction performance.

## 8 IDENTIFYING THE IMPLIED SECURITY REQUIREMENTS – CONTROLLED EXPERIMENT AND REPLICATIONS ON THE USE OF SECURITY REQUIREMENTS TEMPLATES

In this chapter, we address the following research question:

- *RQ4: What is the effectiveness of automatically-suggesting security requirements patterns based on the implied security properties in supporting the identification and specification of security requirements for a system?*

We conducted a controlled experiment, involving 50 graduate students enrolled in a software security course, to evaluate the last two steps of the SRD process related to patterns suggestion and requirements generation (Riaz, Slankas, et al. 2014). Security requirements templates provide a solution for specifying security requirements to support the goals identified in the initial steps of the SRD process. We divided the participants into treatment (automatically-suggested security requirements templates) and control groups (no templates provided). Based on the findings, automatically-suggested templates helped participants (security non-experts) gain awareness about security implications for the software system and consider more security requirements than they would have otherwise.

We conducted three differentiated replications of the original experiment to examine whether the findings can be replicated in different settings, incorporating lessons learned from the original experiment. Replication studies are beneficial to evaluate the validity of prior study findings, either by reproducing results or by isolating factors that can influence

results and that lead to variations. Based on the objective, we analyzed the following research questions in the original experiment and all subsequent replications:

***RQ4.1:** What is the quality of security requirements elicited through the use of automatically-suggested security requirements templates?*

***RQ4.2:** What is the coverage of security requirements elicited through the use of automatically-suggested security requirements templates?*

***RQ4.3:** How relevant are the security requirements elicited through the use of automatically-suggested security requirements templates?*

***RQ4.4:** How efficient is the process of eliciting security requirements through the use of automatically-suggested security requirements templates?*

For clarity, we use the following codes when referring to various experiments:

- NCSU13: Original study conducted at North Carolina State University (NCSU) in 2013.
- UT14: First replication conducted at University of Trento (UT) in 2014.
- NCSU14: Second replication conducted at NCSU in 2014.
- UCR15: Third replication conducted at University of Costa Rica (UCR) in 2015.

The replications have several differences in terms of context factors such as participants and experimental setting. Some of these differences are inherent to all replications involving a different sample than the original experiment. Other differences were introduced to incorporate lessons learned from the original experiment related to quality, coverage and relevance of the responses. Based on these differences, we qualitatively examine the following additional research questions:

*RQ4.5: Are participants more inclined to fill in the templates when additional support to fill the templates is provided by explicitly indicating subject, action and resource elements in the input requirements?*

*RQ4.6: Can participants differentiate whether a suggested security requirements template is relevant to the given use case scenario?*

*RQ4.7: Are there context factors, such as more time on task, which are conducive to producing better outcomes overall?*

## **8.1 Security Requirements Templates used in the Evaluation**

Security community has established a number of knowledge sources, including security catalogues and controls, that capture security expertise and can support elicitation of security requirements. However, a security non-expert may not readily know how and when to leverage the security knowledge sources in the context of a given system. Security requirements templates are a particular example of a broader class of security knowledge that can be sourced from the community by abstracting out common security requirements. Moreover, providing guidance regarding the applicability of various security requirements patterns based on a system's functionality can support the analysis of security requirements. Empirical evaluation on the use of security requirements patterns can inform how well the requirements templates, documents as part of the pattern solution (Section 6.3.3), support the requirements elicitation process.

We developed an initial pool of 19 security requirements templates to support the Security Requirements Discoverer process in specifying security requirements. The 19 templates capture commonly-used security requirements that support core security properties

and were empirically developed from the analysis of ~11,000 sentences drawn from six different natural language requirements artifacts from the healthcare domain (Riaz, King et al. 2014). The templates generate over 40 different security requirements related to confidentiality, integrity, availability, identification and authentication, accountability and privacy properties. For developing the templates, the researchers used an opportunistic approach to identify commonly occurring security requirements in the requirements artifacts and abstracted the subject, resource and action elements out to form reusable parameterized security requirements. Related set of reusable security requirements were then grouped by security properties to form the templates. The 19 security requirements templates are listed in Table 37. Details of these templates are available at our project website<sup>24</sup>.

Table 37. List of Security Requirements Templates

<b>Security Property</b>	<b>Security Requirements Templates</b>
<i>Confidentiality (C)</i>	<ul style="list-style-type: none"> <li>○ <i>C1</i> – authorized access;</li> <li>○ <i>C2</i> – confidentiality during storage;</li> <li>○ <i>C3</i> – confidentiality during transmission;</li> </ul>
<i>Integrity (I)</i>	<ul style="list-style-type: none"> <li>○ <i>I1</i> – read-type actions;</li> <li>○ <i>I2</i> – write-type actions;</li> <li>○ <i>I3</i> – delete actions;</li> <li>○ <i>I4</i> – unchangeable resources;</li> </ul>
<i>Availability (A)</i>	<ul style="list-style-type: none"> <li>○ <i>A1</i> – maintaining availability of data;</li> <li>○ <i>A2</i> – appropriate response time;</li> <li>○ <i>A3</i> – service availability;</li> <li>○ <i>A4</i> – backup and recovery capabilities;</li> <li>○ <i>A5</i> – capacity and performance;</li> </ul>
<i>Identification &amp; Authentication (IA)</i>	<ul style="list-style-type: none"> <li>○ <i>IA1</i> – select context for roles;</li> <li>○ <i>IA2</i> – unique accounts;</li> <li>○ <i>IA3</i> – authentication;</li> </ul>
<i>Accountability (AY)</i>	<ul style="list-style-type: none"> <li>○ <i>AY1</i> – logging transactions with sensitive data;</li> <li>○ <i>AY2</i> – logging authentication events;</li> <li>○ <i>AY3</i> – logging system events;</li> </ul>
<i>Privacy (PR)</i>	<ul style="list-style-type: none"> <li>○ <i>PR1</i> – usage of personal information;</li> </ul>

<sup>24</sup><http://go.ncsu.edu/secreqtemplatesstudy>

As an example, we provide two security requirement templates associated with the properties of accountability and integrity, respectively, in Table 38. Each template groups a set of related security requirements and can be used to instantiate one or more security requirements for the system. For instance, template AY1 can be filled in with details about subject, resource and action elements from input sentence to instantiate two security requirements as shown in the last column of Table 38. The newly-composed security requirements also contain related security properties themselves. Consider the instantiated security requirements for AY1 in Table 38. These requirements suggest an integrity property to prevent modification of log files (I4). The template for AY1 captures this relationship between accountability and integrity by suggesting the requirements analyst to consider template I4 for integrity when identifying the security requirements for accountability.

Table 38. Example Security Requirements Templates.

<b>Input Sentence:</b> <i>The system should provide a means to view discharge instructions for a particular patient.</i>			
<b>Security Property</b>	<b>Automatically-Suggested Security Requirements Templates</b>		<b>Instantiated Security Requirements</b>
<b>Account-ability</b>	AY1	<b>Logging transactions with sensitive data</b> <i>Given:</i> <subject> = user or role <resource> = sensitive information <action> = create/read/update/delete <b>Add Security Requirements:</b> <ul style="list-style-type: none"> <li>The system shall log every time &lt;subject&gt; [performs the] &lt;action&gt; &lt;on   for&gt; &lt;resource&gt;. [see templates C1, I4]</li> <li>At a minimum, the system shall capture the following information for the log entry: &lt;subject&gt; identification, timestamp, &lt;action&gt;, &lt;resource&gt;, and identification of the owner of &lt;resource&gt;.</li> </ul>	<ul style="list-style-type: none"> <li>The system shall log every time user views discharge instructions for a particular patient.</li> <li>At a minimum, the system shall capture the following information for the log entry: user identification, timestamp, view discharge instructions, patient identification.</li> <li>The system shall not allow modification of the log by any user.</li> </ul>
		<b>Maintaining integrity of unchangeable resources</b> <i>Given:</i> <resource> = write-once information (e.g., log files) <b>Add Security Requirements:</b> <ul style="list-style-type: none"> <li>The system shall not allow modification of &lt;resource&gt; by any user.</li> </ul>	
<b>Integrity</b>	I4		

The original experiment (Riaz, Slankas, et al. 2014), and the subsequent replications reported below, evaluate the last two steps of the SRD process related to patterns suggestion and requirements generation.

## **8.2 Research Methodology**

In the following subsections, we provide details about the methodology for conducting the experiments. We also document the similarities and differences between NCSU13 and the subsequent replications. In all the studies, participants were assigned the task of identifying security requirements based on a given use case scenario. Participants were randomly placed into treatment and control group. The treatment group carried out the task with the support of automatically-suggested security requirements templates whereas the control group did not receive such support.

The motivation for conducting the three differentiated replications of the original experiment is to examine whether the findings can be replicated in different settings (in-class vs. take-home), different level of support (in filling templates), difference in motivation and different problem domain (healthcare vs. mobile banking), incorporating lessons learned from the original experiment (see Table 41).

We document the original experiment in accordance with the reporting guidelines by Jedlitschka et al. (Jedlitschka, Ciolkowski, and Pfahl 2008). Moreover, we follow the initial set of guidelines for reporting experimental replications proposed by Carver (J. Carver 2010). We used the quality checklist for quantitative studies by Kitchenham et al., (B Kitchenham and Charters 2007) to guide our research design and execution.

We have included all the recommended details about the original study as well as the replications. We also detail the results of individual studies (Section 8.3) as well as analysis and synthesis across multiple studies (Sections 8.4 and 8.5 respectively).

### 8.2.1 Goals, Hypotheses and Metrics

In the original experiment, and all subsequent replications, we want to determine whether the use of automatically-suggested security requirements templates leads to efficient and effective elicitation of security requirements when compared to a manual approach based on personal expertise. We test the following null hypotheses to address our research questions RQ4.1-RQ4.4:

*H<sub>01</sub>*: The **quality** of elicited security requirements is unrelated to the use of automatically-suggested security requirements templates. [RQ4.1]

*H<sub>02</sub>*: The **coverage** of elicited security requirements is unrelated to the use of automatically-suggested security requirements templates. [RQ4.2]

*H<sub>03</sub>*: The **relevance** of elicited security requirements is unrelated to the use of automatically-suggested security requirements templates. [RQ4.3]

*H<sub>04</sub>*: The **efficiency** of the requirements elicitation process is unrelated to the use of automatically-suggested security requirements templates. [RQ4.4]

We compute the metrics listed in Table 39 for each participant's response and use the results to test the preceding hypotheses. By using the same criteria and comparable metrics, we support comparison of findings across the original experiment and subsequent replications (B Kitchenham and Charters 2007).

Table 39. Metrics used for evaluating participants' responses.  
*[w.r.t: with respect to]*

<b>Evaluation Criteria</b>	<b>Metric Type</b>	<b>Metrics Used</b>
Quality, <i>of security requirements</i>	Qualitative	Likert-like scale (1-5): lower score indicates lower quality.
Coverage, <i>of security requirements</i>	Quantitative	<ul style="list-style-type: none"> <li>• Recall w.r.t security requirements in the oracle.</li> <li>• Recall w.r.t security requirements templates in the oracle. <i>(for UCRI5)</i></li> </ul>
Relevance, <i>of security requirements</i>	Quantitative	<ul style="list-style-type: none"> <li>• Precision w.r.t security requirements in the oracle.</li> <li>• Precision w.r.t security requirements templates in the oracle. <i>(for UCRI5)</i></li> </ul>
Efficiency, <i>of process for eliciting security requirements</i>	Quantitative	<ul style="list-style-type: none"> <li>• # of security requirements in the oracle identified per minute.</li> <li>• # of security requirements templates in the oracle identified per minute. <i>(for UCRI5)</i></li> </ul>

For computing the metric for quality of identified security requirements, we used a Likert-like scale of 1 to 5 (1: poor; 2: below average; 3: average; 4: above average; 5: good). Two researchers independently assigned quality scores for each participant's response, using following questions as guide:

- Are the requirements too general or too specific?
- Have all necessary elements of the requirements been identified (e.g., subject, resource, action, data to be logged)?
- Are there any logical inconsistencies in the requirements?
- Are different types of security properties considered?
- For the treatment group, have the selected templates been filled-in with appropriate details?

For coverage and relevance, we respectively used the metrics for recall and precision computed based on an oracle of security requirements developed a priori to the conduct of the study (Section 8.2.6). Metrics for quality, coverage and relevance evaluate the security

requirements identified by the participants. The metric for efficiency evaluates the requirements elicitation process.

### **8.2.2 Participants**

In this section, we report the demographic summary of the participants for the original experiment as well as all its replications.

For the original study, NCSU13, participants were graduate students enrolled in a 16-weeks software security course<sup>25</sup> offered at NCSU in Fall 2013. Researchers conducted the study as an online web-based activity during the last week of the course, after students had learned various software security concepts. The task for this study was mandatory for all the students to complete, similar to other class exercises. Based upon the IRB approval obtained for the study, students could opt-out of participating in the study, which would preclude the inclusion of their work in the study results. Of the 54 students enrolled in the course, 50 gave consent to use their responses for the study. Each student received coursework credit for completing the task as a classroom exercise, irrespective of their decision to participate in the study or of the quality of their responses.

The first replication, UT14, was conducted at the University of Trento (UT), Italy within a course in Security Engineering at the Master level offered in Fall 2014. The total enrollment in the course was 35 students, out of which 32 gave consent to participate in the study. Of the 35 students, 13 students were enrolled in a special curriculum on security and privacy while the other students were enrolled in the general Master of Science in Computer

---

<sup>25</sup> <https://sites.google.com/a/ncsu.edu/csc515-software-security/>

Science. The exercise was given after 8 hours of lectures on introduction to security concepts covered over 4 lectures. The concepts include the Confidentiality, Integrity, and Availability (CIA) triad, security controls, security management methodologies (e.g. COBIT<sup>26</sup>) and security risk management (e.g. NIST 800-30<sup>27</sup>, ERM COSO<sup>28</sup>). The exercise was presented in class and the material was given as a take home exercise from Wednesday to the following Tuesday.

The second replication, NCSU14, was conducted by the researchers of the original study at NCSU. The participants were students enrolled in the same course as the original experiment, taught in Fall 2014. Of the 110 enrolled students, 107 agreed to participate in the study. Each student received coursework credit for completing the task as a classroom exercise irrespective of his or her decision to participate in the study. However, in contrast to the original study where all students received a standard participatory grade, the coursework credit given for NCSU14 was based on the quality of each student's response as evaluated by the teaching assistants for the course. The change in incentive was based on findings of the original study to explore whether using differentiated credit based on quality might lead to improved overall quality of the responses (Section 8.5.1).

The third replication, UCR15, was conducted at University of Costa Rica (UCR) in Summer 2015. The participants were first year Master's degree students enrolled in a course on Software Metrics. All the 16 enrolled students participated in the study. The exercise

---

<sup>26</sup> <http://www.isaca.org/cobit/pages/default.aspx>

<sup>27</sup> [http://csrc.nist.gov/publications/nistpubs/800-30-rev1/sp800\\_30\\_r1.pdf](http://csrc.nist.gov/publications/nistpubs/800-30-rev1/sp800_30_r1.pdf)

<sup>28</sup> <http://www.coso.org/ERM-IntegratedFramework.htm>

performed as part of the study was graded and represented a significant percentage (25%) of the final grade. Participants, therefore, had a strong incentive to produce good quality results.

At the end of the task, we asked participants to report their experience in three academic categories (CS: Computer Science, SE: Software Engineering, and Security related education) and three work-related categories (CS, SE, and Security related work experience). Table 40 summarizes the participant's background across the replication studies. Within each category, participants could select one of the four experience categories as listed in the table (>5 years; 3-5 years; 1-2 years; <1 year). In Table 40, we have highlighted the maximum frequency for each study for treatment and control group. Within each study, participants in the treatment and control groups have comparable experience based on the frequency of participants across various experience categories, minimizing potential biases. Since the groups are heterogeneous (to provide a greater power of generalization), we do not make comparison across different replications. Experience is self-reported by participants and the semantics and interpretation of the word "experience" might vary for each participant.

Table 40. Frequency of Participants' Academic and Work Experience across Studies.

(CS: Computer Science; SE: Software Engineering; Sec: Security)

Study	Group	Number of Participants	Experience (yrs)	Academic (A)			Work (W)		
				CS	SE	Sec	CS	SE	Sec
NCSU 13	Treatment	30	> 5 years	16	4	2	0	0	0
			3-5 years	9	11	2	11	6	1
			1-2 years	1	7	7	4	5	2
			<1 year	0	4	15	11	15	23
			Not Responded	4	4	4	4	4	4
	Control	20	> 5 years	11	3	0	0	0	0
			3-5 years	8	10	2	9	7	0
			1-2 years	0	4	5	3	6	5
			<1 year	1	3	13	8	7	15
			Not Responded	0	0	0	0	0	0
UT14	Treatment	17	> 5 years	0	0	0	0	0	0
			3-5 years	10	0	0	3	1	0
			1-2 years	3	5	4	2	1	1
			<1 year	4	12	13	12	15	16
			Not Responded	0	0	0	0	0	0
	Control	15	> 5 years	1	0	0	0	0	0
			3-5 years	10	2	0	0	0	0
			1-2 years	1	5	6	5	1	1
			<1 year	3	8	9	10	14	14
			Not Responded	0	0	0	0	0	0
NCSU 14	Treatment	55	> 5 years	15	1	0	0	0	0
			3-5 years	36	18	0	11	5	0
			1-2 years	2	27	9	31	31	3
			<1 year	2	9	46	13	19	52
			Not Responded	0	0	0	0	0	0
	Control	52	> 5 years	23	2	0	1	1	0
			3-5 years	20	18	1	15	8	0
			1-2 years	9	24	10	24	21	9
			<1 year	0	8	41	12	22	43
			Not Responded	0	0	0	0	0	0
UCR1 5	Treatment	9	> 5 years	5	4	2	2	3	0
			3-5 years	3	2	0	2	1	0
			1-2 years	0	2	1	3	2	2
			<1 year	1	1	6	2	3	7
			Not Responded	0	0	0	0	0	0
	Control	7	> 5 years	2	1	0	1	1	0
			3-5 years	5	4	0	5	4	0
			1-2 years	0	2	2	0	1	1
			<1 year	0	0	5	1	1	6
			Not Responded	0	0	0	0	0	0

### **8.2.3 Study Environment**

In this section, we present the details related to the study environment that were shared among all the experiments. We also provide details in terms of setting that were different across the experiments.

#### **8.2.3.1 Shared Aspects of Study Context**

The original experiment and all subsequent replications were conducted as an online activity. All students received a URL to access the online site for the study. On accessing the site, the students first viewed the consent form. Students then read the consent form and could either allow or deny use of their data in the study results. Next, the system assigned each student an auto-generated random access code. The student was randomly assigned to treatment or control group and shown a screen with instructions for completing the task based on the group the student was assigned to. We provide more details about the grouping in Section 8.2.5 when we discuss the experiment design. Having read the instructions, the student could continue to the task of identifying security requirements.

Students could save the task at any point during the experiment and return to the task by entering their access code at the provided URL. For each participant, we recorded total time spent completing the task and whether the participant submitted the task.

After completing the task, we asked participants to:

- Briefly explain the process used for identifying applicable security requirements (e.g., what information you looked at in the use case or reference material).

We solicited feedback on the security requirements templates from the participants in the treatment group by asking the following additional open-ended questions:

- What is your opinion regarding the use of requirements templates?
- What is your opinion regarding the use of generated requirements?

### **8.2.3.2 Differences in Study Context**

The original experiment and replications differed in terms of certain aspects related to the setting of the study. In Table 41, we summarize the context factors for each experiment. We also indicate the metrics that we expect to be affected by the differences in the context factors.

NCSU13 was conducted as an in-class activity at NCSU. Students were encouraged to work on the task during the 60-minute lecture period in a classroom setting. Participants received related reference material two days prior to the conduct of the study. Researchers provided a five-minute overview of the task at the beginning of the lecture period. In addition to the remaining 55 minutes, students had a total of two days to complete the task and were required to submit the task before the start of the next lecture period. Of the 50 participants, 40 completed the task during the lecture period.

The UT14 study was initially planned as an in-class activity. However, almost 30% of the students did not have a laptop so a last minute change was made to assign the task as a take-home activity. In NCSU13, we found a positive correlation between time on task and number of requirements identified (Riaz, Slankas, et al. 2014). Conducting UT14 as a take-home activity provides opportunity to assess if more time on task leads to improvement in the coverage of identified security requirements. Participants received related reference material two days prior to the conduct of the study, as was done in NCSU13. The measurement of time for UT14 might not be as reliable as NCSU13 or NCSU14. The tool cannot differentiate

whether the task window is merely opened or if the participant is actually working on the task. Though the University of Trento is in Italy, the participants in UT14 attended a Master degree where the language of instruction was English and the audience was mostly international. The participants provided responses in English.

Table 41. Summary of Context Factors across Different Experiments.  
*[\*metrics expected to be affected by differences in study context]*

Study → / Context Factors ↓	Original Study [NCSU13]	Replication-1 [UT14]	Replication-2 [NCSU14]	Replication-3 [UCR15]
<b>Participants</b>	50 graduate students	32 graduate students	107 graduate students	16 graduate students
<b>Setting</b>	In-class Activity, NCSU [60 min]	Take-home Activity, UT [1 week] [*Coverage & Efficiency]	In-class Activity, NCSU [60 min]	In-class Activity, UCR [180 min]
<b>Security Training Provided</b>	<ul style="list-style-type: none"> <li>• Software Security course</li> <li>• 4 page reference material</li> </ul>	<ul style="list-style-type: none"> <li>• Security Engineering course</li> <li>• 4 page reference material</li> </ul>	<ul style="list-style-type: none"> <li>• Software Security course</li> <li>• 4 page reference material</li> <li>• 10 min video on security properties and requirements</li> </ul>	<ul style="list-style-type: none"> <li>• 4 page reference material</li> <li>• 10 min video on security properties and requirements</li> <li>• 15 minutes explanatory presentation in Spanish</li> </ul>
<b>Problem Domain</b>	Healthcare	Healthcare	Healthcare	Mobile banking
<b>Other Changes</b>	NA	No other changes	<ul style="list-style-type: none"> <li>• Suggestion to fill in templates</li> <li>• Feedback on quality of responses [*Quality]</li> <li>• Extraneous Templates [*Relevance]</li> </ul>	<ul style="list-style-type: none"> <li>• Suggestion to fill in templates</li> <li>• Feedback on quality of responses [*Quality]</li> <li>• Extraneous Templates [*Relevance]</li> </ul>

In NCSU14, the students performed the task as an in-class activity during the 60-minute lecture period, like the original experiment. Participants received related reference material two days prior to the conduct of the study. However, instead of the introductory overview at the start of the lecture period, participants watched a 10-minute video introducing the general concept of security properties that are implied by natural language requirements artifacts

before the lecture. The video did not include details related to the security requirements templates, and only treatment group had the knowledge of the templates during the conduct of the study, as with all other studies.

In UCR15, the students performed the task as an in-class activity during a 180-minute lecture period. Participants received related reference material, including the 10-minute video presentation given to participants in NCSU14, one week prior to the conduct of the study. Researchers expected the participants to have at least read the provided material (see Section 8.2.4). Participants also received a review of the reference material in the form of a 15-minute presentation in Spanish before the start of the study. Participants in UCR15 had access to the research paper in which the findings of the original experiment were reported (Riaz et al. 2014) however we do not know if they read the paper before the experiment. Participants in UCR15 were mostly native Spanish speakers. Almost half of the participants in the control group provided responses in Spanish. Researchers at UCR translated the responses to English. The templates suggested to the treatment group in NCSU14 and UCR15 included some extraneous suggestions as well, however participants were not aware that some of the suggestions may be extraneous.

#### **8.2.4 Experiment Artifacts**

All participants received four pages of reference material containing a description of software security properties as well as textual clues that can indicate an implied security property. Reference material also contained a total of 40 example security requirements grouped by security properties. We provided this standard reference material to participants prior to the start of the experiment. During the experiment, the control group had access to

the same reference material online. However, for the treatment group, we presented example security requirements in two forms: i) reusable security requirements templates grouped by security properties; and ii) concrete example security requirements (also available to control group) that were generated from the templates. The reference material, use cases and other study documents are available on our project website<sup>29</sup>.

As part of the task, participants identified security requirements based on a given use case scenario. We used the following criteria to select the use cases:

- The use case must focus on a single unit of functionality, such that participants could easily understand the scope of the requirements.
- Understanding the use case shall require no understanding of domain-specific taxonomies.
- The use case shall imply at least four different types of security properties.
- The use case specifications shall be openly accessible.

For the NCSU13, we selected two use cases for participants to identify security requirements for, both from the electronic healthcare domain that met our specified criteria. First use case (UC1 - Document office visit) is from the iTrust<sup>30</sup> electronic health record (EHR) system (Meneely, Smith et al. 2012), an open-source system developed by students at NCSU. The second use case (UC2 - Retrieve exam results by patient ID) is based on a user story<sup>31</sup> from Virtual Lifetime Electronic Record (VLER), a business and technology initiative that allows secure and standardized electronic exchange of health and benefits

---

<sup>29</sup> <http://go.ncsu.edu/secreqtemplatesstudy>

<sup>30</sup> <http://agile.csc.ncsu.edu/iTrust/wiki/doku.php?id=requirements>

<sup>31</sup> [http://www.va.gov/vler/vlerdocs\\_userstories.asp](http://www.va.gov/vler/vlerdocs_userstories.asp)

information for United States Veterans and Service members. Participants in UT14 and NCSU14 identified security requirements related to the same use cases as NCSU13.

For UCR15, to assess generalizability of findings across domains, we selected two use cases from the Cyclos<sup>32</sup> mobile payment software. Cyclos offers a complete mobile banking platform including SMS banking and Mobile application. The first use case (UC1- Make payment) is related to the functionality of making online payment to another member via SMS. The second use case (UC2- Retrieve account information) is related to the functionality for querying account information, such as current account balance, via SMS. Both use cases have a similar number of sentences and readability scores as the use cases in the original study.

We did not introduce any differences in terms of the experiment material given to participants between NCSU13 and UT14. However, since participants in NCSU13 and UT14 were enrolled in different courses, they may differ in terms of knowledge and experience related to security requirements. For the second and third replications, NCSU14 and UCR15, we provided a 10-minute video to participants introducing the concept of security properties and requirements. UCR15 also got separate introductions in Spanish. Participants in UCR15 were also instructed to find as many security requirements as they could.

We also introduced two differences in terms of the support provided to the treatment group as compared to NCSU13. Firstly, participants in the treatment group for NCSU14 and UCR15 received additional details for filling in templates such as the subject, action and resource elements in the use case sentences. Participants in NCSU14 and UCR15 also

---

<sup>32</sup> <http://www.cyclos.org/mobilebanking/>

received course work grade based on the quality of responses. With the additional support and strong incentive on quality, participants were expected to provide better quality responses as compared to participants in NCSU13 where one-third of the participants did not fill in the templates, impacting the quality score. Secondly, we intentionally suggested some extraneous templates to participants in NCSU14 and UCR15 that were not relevant to the given use case sentence. By suggesting extraneous templates, we wanted to assess whether the participants randomly select any suggested template or if they can differentiate between applicable and extraneous templates. The second change can have an impact on the relevance of the requirements identified by the participants. We have summarized these differences in Table 41.

### **8.2.5 Experiment Design**

We used a 2x2 between-subjects design (Lane 2011) for the original study and all subsequent replications. We automatically assigned study participants (students who agreed to participate in the study) to one of four groups in a round-robin fashion based on the process used for identifying requirements and the use case assigned (UC1 or UC2). All groups were given the same task of identifying security requirements. We provided specific sets of instructions, reference material, and task screens depending on the process (treatment vs. control) and the use case (UC1/UC2). To minimize potential bias, participants did not know about the existence of different groups or use cases. They were just informed that they will be given a use case scenario and will have to identify security requirements in accordance with the instructions. Participants could be assigned to one of the following processes for identifying security requirements:

- Treatment (T): automatically-suggested security requirements templates for identifying security requirements.
- Control (C): no templates, manual identification of security requirements.

In Table 42, we document the number of participants in each group for the original and replicated experiments. We recorded no personally identifiable information about the participants (e.g., name, student identifier).

Table 42. Number of Participants in Each Group – Security Requirements Templates Study.

Experiment	Treatment		Control		Total
	UC1	UC2	UC1	UC2	
NCSU13	16	14	10	10	50
UT14	9	8	9	6	32
NCSU14	29	26	25	27	107
UCR15	5	4	4	3	16
Overall	59	52	48	46	205
	111		94		

In Figure 21, we provide the task screen for treatment group for UCR15 (only showing two sentences from the use cases for brevity). We indicate the subject, resource and action elements at the end of each sentence in the use case to help with filling in the security requirements templates.

Security Requirements Discovery
Access Code: 4934482345

---

Instructions

Security Objectives

Example Requirements

Task started at: 2016-04-13 17:22:51.744

Task: Direct payment via SMS

**Context:**  
Cyclos is a secure and scalable payment software. It offers a complete mobile banking platform, including SMS banking and Mobile app.

**Main Flow:**

- <sup>1</sup>A member can make a payment to another member by sending just one SMS to the organization number/short code. *[Subject: member; Action: make, send; Resource: payment, SMS;]*
- <sup>2</sup>If the payment has been processed successfully the payer and receiver will receive a confirmation notification by SMS. *[Subject: payer and receiver; Action: receive; Resource: confirmation notification;]*

**Security objectives associated with statement 1: [Confidentiality, Integrity, Identification and Authentication, Accountability]**

Select pattern to add:

[1 - Confidentiality: Data]  
The system shall enforce access privileges that <enable|prevent> <subject> to <action> <resource>.  
The system shall encrypt <resource> and store <resource> in encrypted format using an industry approved encryption algorithm.  
The system shall transmit <resource> data in encrypted format to and from the authorized <subject>.  
The system shall monitor the status and location of system components that may contain unencrypted <resource> data.  
[1]

Figure 21. Task screen for treatment group in UCR15.

The task screen for control group is similar to treatment group but there are no suggestions of applicable security properties or templates for individual sentences in the use case. Participants have to manually identify applicable security requirements and enter into the text area at the bottom half of the page. For traceability, participants entered the security requirements in the text area followed by sentence number(s) from use case scenario to which the security requirement relates.

### 8.2.6 Evaluation Methodology

In this section, we present the methodology for evaluating the participants' responses to compute the metrics.

### 8.2.6.1 Oracle of Security Requirements

Five software security researchers, including the first three authors, created an oracle of the security requirements for each use case to evaluate the coverage and relevance of security requirements identified by the participants. In Table 43, we provide a summary of the templates associated with each of the use case sentences in the oracle for the use cases selected from the domains of healthcare (used in NCSU13, UT14, NCSU14) and mobile banking (used in UCR15). The templates are selected from the list of templates given in Table 37. The manual steps for creating the oracle are similar to the steps of SD process, as listed below:

- For each sentence in the use case, identify the security properties associated with the sentence.
- For each identified property, select the security requirements templates that are applicable.
- For each applicable security requirements template, instantiate the templates by filling-in contextual details from the original sentence to generate concrete security requirements.
- Remove duplicate or redundant requirements.

We used the same process for creating the solution oracle for all the use cases. Three of the five researchers who participated in the creation of study oracle for the original study, NCSU13, were involved in the creation of the oracle for UCR15. For consistency, we used the same classification guide for UCR15 as for NCSU13 when deciding on applicable properties and templates. The classification guide lists textual clues that can indicate an

implied security property and the guide was provided to participants as part of the reference material as well (see Table 26).

Table 43. Security requirements templates associated with use case sentences in the oracle.

Use Case	Security Requirements Template	Sentences in Use Case Implying the Template
<b>UC1-Health</b> <i>[NCSU13, UT14, NCSU14]</i>	C1: Confidentiality – Data	3, 4, 5, 7, 8, 9, 10
	I1: Integrity – Read-type actions	9, 10
	I2: Integrity – Write-type actions	3, 4, 5, 6, 7, 8
	A1: Availability – Maintaining availability of data	9
	IA2: Identification and authentication – Unique accounts	1, 2
	IA3: Identification and authentication – User authentication	2
	AY1: Accountability – Logging transactions of sensitive data	3, 4, 5, 6, 7, 8, 9, 10
	AY3: Accountability – Logging system events	2
<b>UC2-Health</b> <i>[NCSU13, UT14, NCSU14]</i>	PR1: Privacy – Usage of personal information	3, 4, 5, 7, 8, 9
	C1: Confidentiality – Data	1, 3, 4, 5, 6, 8, 9
	A1: Availability – Maintaining availability of data	9
	AY1: Accountability – Logging transactions of sensitive data	2, 3, 4, 5, 6, 8, 9
<b>UC1-Mobile</b> <i>[UCR15]</i>	PR1: Privacy – Usage of personal information	1, 3, 4, 5, 8, 9
	C1: Confidentiality – Data	1, 2, 3, 4, 5, 6, 7, 8, 9
	I1: Integrity – Read-type actions	6, 7
	I2: Integrity – Write-type actions	1, 3, 4, 5
	IA2: Identification and authentication – Unique accounts	1, 2, 3, 4, 5
	AY1: Accountability – Logging transactions of sensitive data	1, 2, 3, 4, 5, 8, 9
<b>UC2-Mobile</b> <i>[UCR15]</i>	AY3: Accountability – Logging system events	6, 7
	C1: Confidentiality – Data	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
	I1: Integrity – Read-type actions	5, 6, 7
	I2: Integrity – Write-type actions	1, 2, 3, 4, 10
	IA2: Identification and authentication – Unique accounts	1, 2, 3, 4, 8, 9, 10
	AY1: Accountability – Logging transactions of sensitive data	1, 4, 8, 9, 10
	AY3: Accountability – Logging system events	5, 6, 7

### 8.2.6.2 Mapping Responses to the Oracle

The security requirements in the participants’ responses were mapped to the requirements in the oracle to compute the metrics for coverage and relevance. For each participant’s response, if a requirement in a response could be mapped to a requirement in the oracle, it was considered as a true positive (TP). If a requirement in a response could not be mapped to a requirement in the oracle, we considered two cases: a) the requirement is not related to the given use case and is thus a false positive (FP); and b) the requirement is related to the given use case scenario and should be marked as true positive (TP). In the latter case, we would

also update the oracle to include the newly identified requirement. However, none of the participants identified any security requirement that was relevant to the scenario but not in the oracle already. Lastly, the requirements in the oracle not identified by a participant would be considered false negatives (FN).

For the treatment group, the requirements in the responses could be directly mapped to the requirements in the oracle as the requirements were generated using the same templates in both cases (TP). If a requirement was generated using an extraneous template, the requirement was marked as FP. Control group did not have the templates however they had example security requirements (one example requirement generated using each template) available with them. A number of participants in the control group used those examples as guide when specifying security requirements. Participants in the control group could also use their own words to phrase security requirements in which case we would not have a direct mapping to requirements in the oracle. In such cases, we mapped the requirements in the participants' response to one or more of the closest matching requirements in the oracle (TP). For instance, if a participant in the control group specified a general confidentiality requirement (e.g., all data should be encrypted during transmission), we mapped it to all the requirements for confidentiality during transmission in the oracle (e.g., encrypt passwords during transmission, encrypt health records during transmission) to minimize the potential advantage that treatment group had through the availability of templates. A participant in the treatment group would have to select the corresponding template (Confidentiality during transmission) for sentences related to each individual resource (e.g., passwords, health records) to get a similar mapping. In some cases, a requirement in the response for control

group would partially map to a requirement in the oracle, in which case we still counted the requirement to be identified (TP). If a requirement in the response did not map to any of the requirements in the oracle, and was not relevant to the given scenario, we marked the requirement as FP.

For UCR15, we defined the oracle in terms of the templates (i.e., did not instantiate the templates to generate individual security requirements in the oracle). Control group in UCR15 explicitly mentioned which security property the requirement was related to and we mapped the requirement to the corresponding template (e.g., if the participant mentioned a requirement related to integrity, we mapped it to the template related to the property of integrity). Essentially, we are comparing which security properties each participant considered for a given sentence in the use case for treatment and control groups in UCR15. We may not get as granular mapping as other studies, but we still found significant differences in coverage between treatment and control groups in UCR15 indicating that participants in the treatment group considered significantly more security properties and corresponding templates.

The metrics for coverage and relevance provide an assessment of the participant's performance in terms of how many requirements in the oracle a participant identified as well as how much of the effort was spent in identifying relevant requirements (TP) versus irrelevant ones (FP). We can have cases where for high relevance score, the participant has low coverage (e.g., participant only identified a few requirements and those requirements were in the oracle) and vice versa (e.g., participant identified a large number of requirements in the oracle, but also identified a large number of irrelevant requirements). We may also

have cases where for the same relevance score, participants have different coverage scores (e.g., participant A identifies only 5% requirements in the oracle and no irrelevant requirement while participant B identifies 80% requirements in the oracle and no irrelevant ones – both have 100% relevance score) and vice versa. In such cases, we may see no relation between the two metrics (Menzies et al. 2007).

### **8.3 Results Based on Individual Experiments**

We present the results from each replication below and discuss whether the results support the hypotheses give in Section 8.2.1. We have used 2x2 ANOVA for unbalanced groups, adjusting for multiple comparisons (Tukey), to test the four null hypotheses, as in the original study. We used SAS version 9.4 for the statistical analysis. The two factors for grouping are: i) requirements process (tgroup) which can be either treatment or control; and ii) use case (ucid) which can be either 1 or 2. Using the analysis, we determine whether the factors or the interaction between the factors leads to significantly different group means for the four metrics. Results with  $p < 0.05$  are considered significant for our analysis. Our data meets the ANOVA assumptions of independence, normality and homogeneity of variance (Levene's test) for the treatment and control groups across the studies in general. However, for UT14 and NCSU14, the homogeneity of variance assumption doesn't hold for the metric of relevance due to the large variations in the relevance scores for the control group as compared to the treatment group. For efficiency, the distribution of treatment group is slightly skewed due to a couple of outliers with high efficiency scores however the outliers do not affect the significance of results. Quality is a likert-like scale and the quality scores are normally distributed. Considering quality as an interval scale (assuming difference

between scores 1 and 2 is similar to difference between scores 2 and 3), ANOVA can be applicable given that the parametric assumptions are satisfied (McCrum-Gardner 2008).

Two raters individually evaluated the responses and consolidated the evaluation through discussion. The two raters also assigned the quality scores based on a pre-specified criterion for assessing quality. For the metric of quality, we applied weighted kappa (Viera 2005) using linear weights to assess how far apart the two raters were in assigning the quality scores. The weighted kappa scores<sup>33</sup> are 0.689 (good agreement), 0.948 (very good agreement) and 0.848 (very good agreement) for UT14, NCSU14 and UCR15 respectively.

We provide overall mean scores for all metrics across studies in Table 44 for a high-level overview of the findings across studies. We discuss the results for each study in the following subsections. The results discussed in this section provide insights into the answers for research questions RQ4.1 to RQ4.4.

Table 44. Overall mean scores for all metrics across studies.  
[\*scaled for comparison]

Study	Time available for completing the task	Mean time on task	Mean Quality (1-5)	Mean Coverage (0-1)	Mean Relevance (0-1)	Mean Efficiency (req./min)
<b>NCSU13</b>	60 minutes in-class	~20 minutes	2.88	0.31	0.88	1.14
<b>UT14</b>	One week to complete at home	~47 minutes	2.70	0.36	0.77	1.07
<b>NCSU14</b>	60 minutes in-class	~25 minutes	2.66	0.37	0.73	1.01
<b>UCR15</b>	180 minutes in-class	~102 minutes	3.69	0.51	0.66	0.64*

### 8.3.1 NCSU13: Original Experiment at NCSU

A summary of findings from the original experiment, NCSU13, are listed in Table 45.

We discuss results based on each of the four metrics below.

---

<sup>33</sup> <http://graphpad.com/quickcalcs/kappa1/?K=5>

We found no statistically significant difference between the quality of requirements identified by the treatment vs. control groups ( $p\text{-value}=0.928$ ) or between either of the use cases ( $p\text{-value}=0.891$ ) at  $p<0.05$ . Thus, we fail to reject the null hypothesis  $H_{01}$  and cannot state that the quality of identified requirements is affected by the use of security requirements templates. Overall, neither group produced quality requirements. Although the treatment group was able to identify the applicable requirements templates, one-third of the participants did not fill-in the templates (8 out of 16 for UC1; 2 out of 14 for UC2). In addition to the classroom time, participants had two days to complete the exercise. Only one participant who started the task in class submitted it later, after briefly reviewing the response. Since study responses were collected anonymously, participants were required to submit a separate sheet of paper to the instructor to receive credit for participation in the classroom exercise (irrespective of their participation in the study). Anonymity, combined with a lack of a stronger incentive, might have reduced participant motivation for producing a top quality assignment. Only one participant in the treatment group did not use the automatically-suggested templates. Conversely, many participants in the control group incorporated quality security requirements by leveraging the example requirements provided as part of the reference material. The control group participants who did not use the reference material often documented requirements that were too general (e.g., "Send the results over a secure channel") or too specific, discussing security mechanisms (e.g., "Avoid URL jumping by using HTTP referrer fields...").

We found both the requirements process (treatment vs. control,  $p\text{-value}=<0.0001$ ) and use case (UC1 or UC2,  $p\text{-value}=0.0002$ ) to be significant factors in determining the coverage

of security requirements at  $p < 0.05$ . The interaction between the requirements process and use case was also significant ( $p\text{-value} = 0.0054$ ). Thus, we reject the null hypothesis  $H_{02}$  and determine that the use of automatically-suggested security requirements templates helped in identifying significantly more requirements when compared to the control group. Overall participants identified only 31% of all the security requirements in the oracle. Participants in the treatment group identified significantly more requirements when compared to the participants in the control group. For UC1, participants in treatment group identified twice as many requirements as control group. For UC2, treatment group identified three times the requirements identified by the control group. We found significant differences in the coverage of identified requirements in the treatment versus control groups, as well as for the two use cases given to the participants. Automatically-suggested templates helped participants in the treatment group to consider more security requirements for the system. Participants in the control group did not know about the templates. Although reference material for the control group contained information about the security properties and various textual clues that could imply a security property, participants had to figure out applicable properties on their own. Participants in the control group were not able to incorporate requirements to meet the various security properties of the system indicating that they were not able to identify and consider all the security properties.

We did not find statistically significant difference in the ratio of relevant requirements in the response between treatment and control groups ( $p\text{-value} = 0.34$ ) at  $p < 0.05$ . Thus, we fail to reject the null hypothesis  $H_{03}$  that ratio of relevant requirements to total requirements identified is unrelated to the use of security requirements templates. Although we did not find

a statistically significant difference in the ratio of relevant requirements identified between various groups, participants in the treatment group performed slightly better (90% of the identified requirements were relevant) than the control group (85% of the identified requirements were relevant).

Table 45. Results of 2x2 ANOVA for NCSU13.

Factor ↓ / Metric →		Quality (1-5)		Coverage (0-1)		Relevance (0-1)		Efficiency (req/min)	
		Means	p-value	Means	p-value	Means	p-value	Means	p-value
Requirements Process (tgroup)	Treatment	2.87	0.9283	0.43	<0.0001	0.90	0.3410	1.35	0.0381
	Control	2.9		0.16		0.85		0.77	
Use case (ucid)	1	2.87	0.8909	0.18	0.0002	0.88	0.8677	1.43	0.0102
	2	2.91		0.40		0.87		0.7	

Participants in the treatment group received suggestions related to the security requirements templates that might be applicable based on the security properties implied by sentences in the given use case. However, not all the suggested templates may be applicable for a given functional requirement. Some of the participants in the treatment group selected all suggested templates increasing the number of irrelevant requirements identified. Overall, participants in both treatment and control group listed mostly relevant security requirements (over 88% of all the identified requirements were relevant)

We found both the requirements process (treatment vs. control, p-value=<0.038) and use case (UC1 or UC2, p-value=0.01) to be significant factors in determining the efficiency at p<0.05. Thus, we reject the null hypothesis  $H_{04}$  and determine that the use of our automatically-suggested templates significantly improved the efficiency of the requirements elicitation process. Participants in the treatment group spent an average of six additional minutes eliciting security requirements (22 minutes for treatment vs. 16 minutes for control).

Participants spent an average of 17 minutes for UC1 vs. 21 minutes for UC2. On average, participants spent 20 minutes on the task and differences in mean time on task between various groups are not statistically significant. We found a strong linear relation between task time and requirements identified (Pearson correlation coefficient [4] = 0.44). Participants who spent more time on the task identified more security requirements in general. Participants in the treatment group identified twice as many security requirements per unit of time when compared to the control group. Participants working on UC1 were also twice as efficient in identifying security requirements as compared to UC2. This can be attributed to the fact that UC1 has three-times as many security requirements as UC2 so there were more requirements to be considered. The lack of significant difference in mean time on task between various groups can be attributed to the observation that the participants perceived the study as a classroom activity and did not use extra time to work on the task.

### **8.3.2 UT14: Replication at UT**

Based on the results of UT14 as listed in Table 46, we found the requirements process (treatment vs. control,  $p\text{-value} < 0.0001$ ) to be a significant factor in determining the relevance of identified security requirements. Thus, we reject the null hypothesis  $H_{03}$  that ratio of relevant requirements to total requirements identified is unrelated to the use of security requirements templates. Almost 90% of the security requirements identified by the participants using the templates were relevant (88% for UC1, 92% for UC2). Only 61% of the security requirements identified by participants in the control group were relevant (66% for UC1, 53% for UC2) on average. We did not find any other significant differences based on the requirements process or use cases. Thus, we fail to reject the null hypotheses  $H_{01}$ ,  $H_{02}$

and H<sub>04</sub> for UT14. The difference in coverage of security requirements between treatment and control group (41% vs. ~30%) is not significant at  $p < 0.05$  but it is significant at  $p < 0.1$ . The coverage scores are also close to the corresponding values for NCSU14. The interaction between requirements process and use case is not significant for any of the metrics.

Table 46. Results of 2x2 ANOVA for UT14.

Factor ↓ / Metric →		Quality (1-5)		Coverage (0-1)		Relevance (0-1)		Efficiency (req/min)	
		Means	p-value	Means	p-value	Means	p-value	Means	p-value
Requirements Process (tgroup)	Treatment	2.91	0.1230	0.41	0.0782	0.90	<0.0001	1.37	0.2731
	Control	2.47		0.30		0.61		0.74	
Use case (ucid)	1	2.75	0.6220	0.30	0.0836	0.77	0.4674	1.33	0.2988
	2	2.64		0.43		0.76		0.74	

We provide box-plots capturing group means and variance for each requirements process (tgroup) and use case (ucid) in Figure 22. On average, participants in the treatment group performed better than participants in the control group for each use case across all the four metrics.

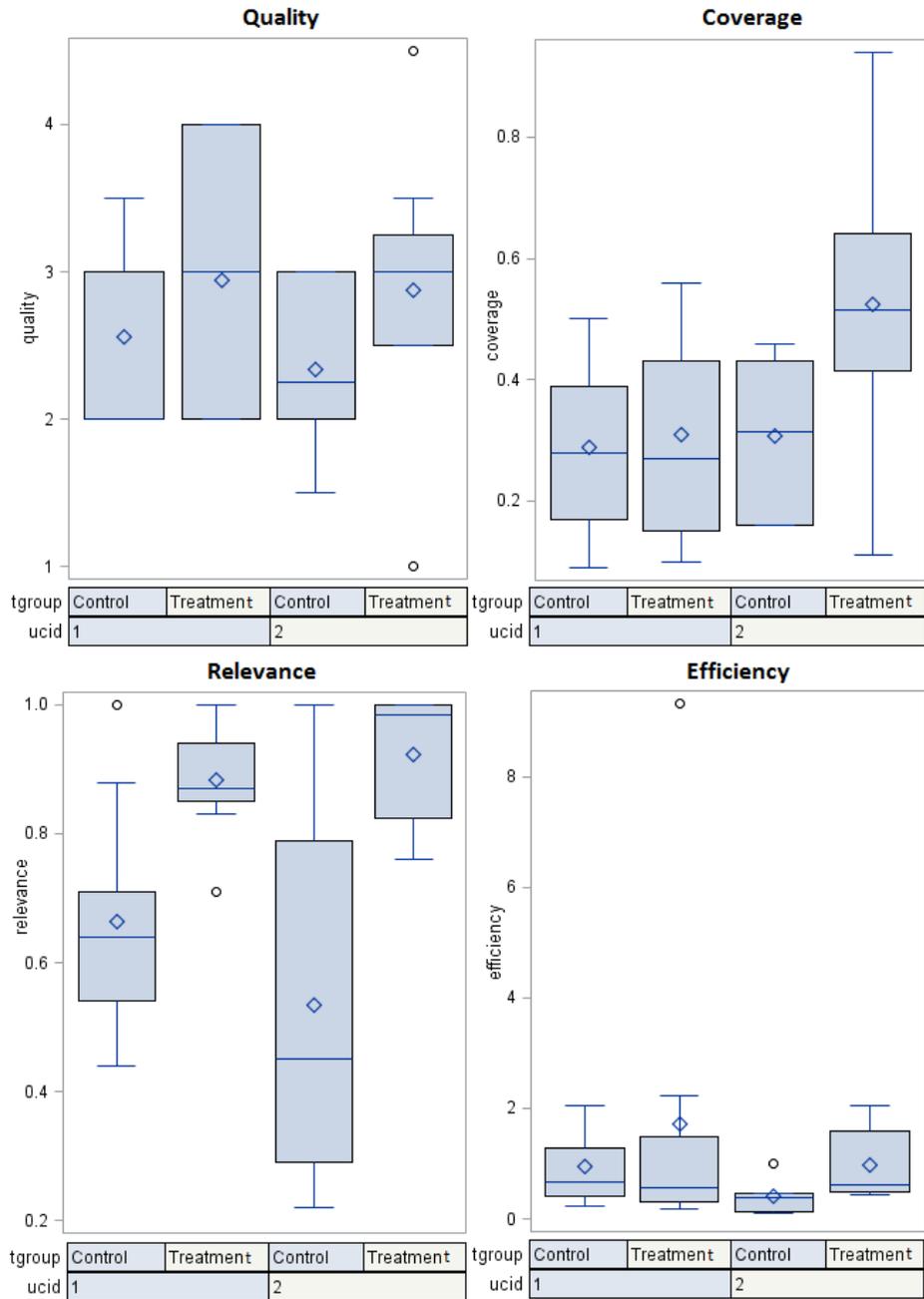


Figure 22. Box-plots with results from UT14 across each factor and metric

### 8.3.3 NCSU14: Replication at NCSU

Based on the results of NCSU14 as listed in Table 47, we found that participants in the treatment group performed significantly better than the control group across all four metrics. Thus, we reject the null hypotheses  $H_{01}$ ,  $H_{02}$ ,  $H_{03}$  and  $H_{04}$  that the performance of participants based on metrics for quality, coverage, relevance and efficiency is unrelated to the use of security requirements templates. Participants in the treatment group produced significantly better quality requirements (3.33 versus 1.95 for control group). Participants in the treatment group also identified significantly more requirements (~46% vs. ~26% for control group) and the identified requirements were more often relevant (~91% vs. ~54% for control group). The participants in the treatment group were also 40% more efficient as compared to the control group overall. This relative efficiency translates to the identification of an additional requirement every three minutes for the treatment group and may not provide any practical significance in terms of the efficiency of the process.

Table 47. Results of 2x2 ANOVA for NCSU14.

Factor ↓ / Metric →		Quality (1-5)		Coverage (0-1)		Relevance (0-1)		Efficiency (req/min)	
		Means	p-value	Means	p-value	Means	p-value	Means	p-value
Requirements Process (tgroup)	Treatment	3.33	<0.0001	0.46	<0.0001	0.91	<0.0001	1.18	0.0221
	Control	1.95		0.26		0.54		0.84	
Use case (ucid)	1	2.71	0.7706	0.27	<0.0001	0.75	0.3932	1.28	0.0003
	2	2.60		0.46		0.70		0.74	

We also found significant differences between the coverage (p-value <0.0001) of security requirements identified for each use case. The difference in efficiency (p-value=0.0003) is also significant. We have a total of 110 security requirements for UC1 versus 35 security requirements for UC2 in the oracle. Although the participants working on UC1 were more

efficient, as they identified more total security requirements per unit of time, the percentage of identified requirements for UC1 was less compared to the percentage of requirements identified for UC2.

The interaction between requirements process and use case was not significant for any of the metrics at  $p\text{-value} < 0.05$ . We provide box-plots capturing group means and variance for each requirements process (tgroup) and use case (ucid) in Figure 23. Participants in the treatment group performed better than participants in the control group for each use case across all the four metrics.

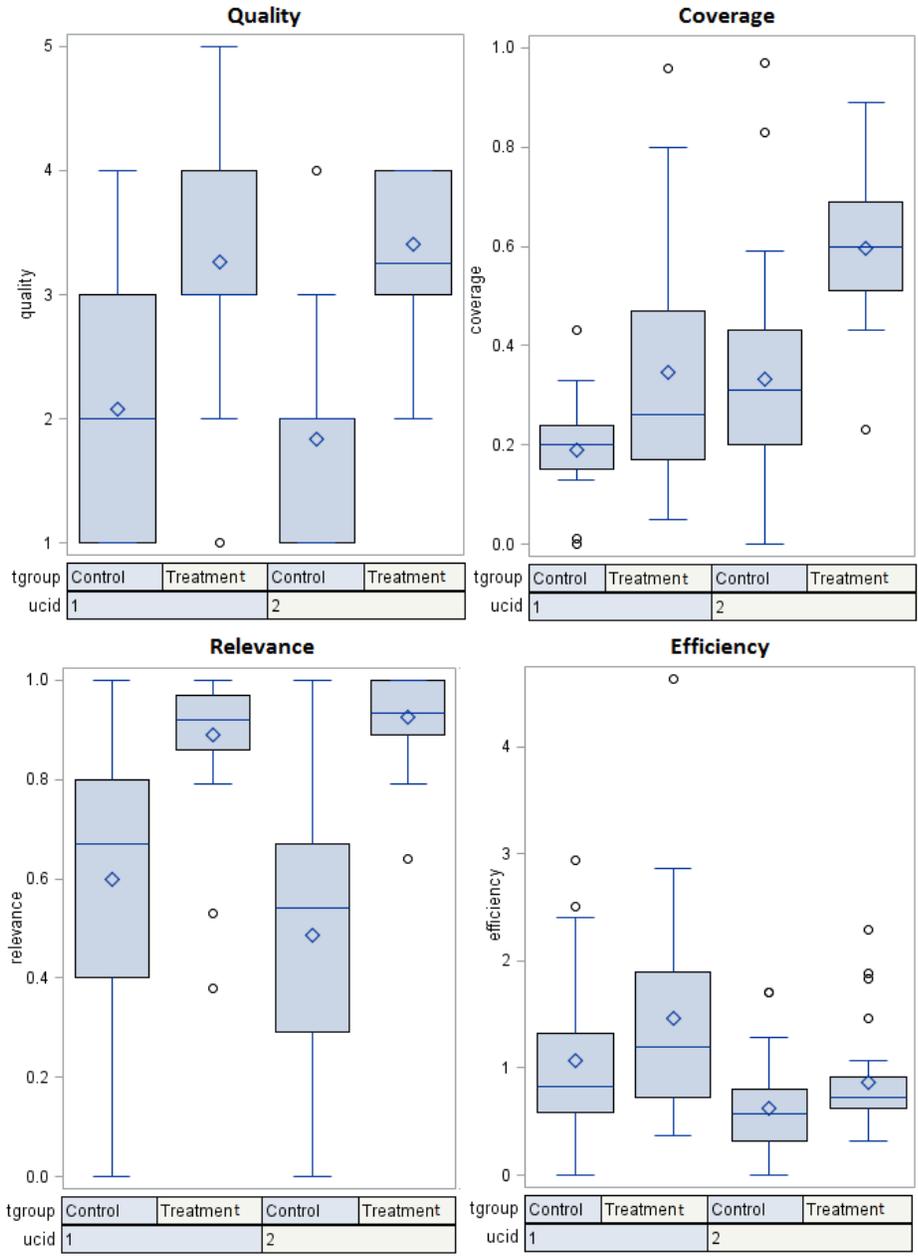


Figure 23. Box-plots with results from NCSU14 across each factor and metric

### 8.3.4 UCR15: Replication at UCR

Based on the results of UCR15 as listed in Table 48, we found the requirements process (treatment vs. control) to be a significant factor in determining the metrics for coverage (p-value=0.0162) and efficiency (p-value=0.013). Thus, we reject the null hypotheses  $H_{02}$  and  $H_{04}$  that the performance of participants based on metrics for coverage and efficiency is unrelated to the use of security requirements templates. Participants in the treatment group identified almost 63% of all the requirements in the oracle (65% for UC1, 60.5% for UC2). In comparison, participants in the control group identified only 36% of all the requirements in the oracle (40.5% for UC1, 30% for UC2) on average.

Participants in the treatment group were also 77% more efficient as compared to control group, although the efficiency scores for both the groups is much lower as compared to other experiments. The reason for this difference is primarily that, for UCR15, we computed the metrics based on the number of security requirements templates identified per minutes instead of the number of individual security requirements identified per minute. Each template generates multiple security requirements as discussed earlier.

Table 48. Results of 2x2 ANOVA for UCR15.

Factor ↓ / Metric →		Quality (1-5)		Coverage (0-1)		Relevance (0-1)		Efficiency (templates/min)	
		Means	p-value	Means	p-value	Means	p-value	Means	p-value
Requirements Process (tgroup)	Treatment	3.94	0.1565	0.63	<b>0.0162</b>	0.71	0.0631	0.20	<b>0.0130</b>
	Control	3.36		0.36		0.59		0.11	
Use case (ucid)	1	3.83	0.3998	0.54	0.4650	0.66	0.9695	0.14	0.2679
	2	3.50		0.47		0.66		0.18	

We did not find any significant differences for the metrics of quality and relevance based on the requirements process or use cases. Thus, we fail to reject the null hypotheses  $H_{01}$  and

H03 for UCR15. The difference in relevance of security requirements between treatment and control group (71% vs. ~59%) is not significant at  $p < 0.05$  but is significant at  $p < 0.1$ . Although participants in the treatment group received suggestion for extraneous templates, the requirements identified are still more relevant as compared to the control group. The overall quality of responses in UCR15 was better than all other replications at 3.7 compared to a range of 2.7 to 2.9 for other studies. The difference might be due to the change in use cases for the study or the fact that participants spent the most time on task, two to five times more than other studies. Interaction between the requirements process and use case is not significant for any of the metrics.

We provide box-plots capturing group means and variance for each requirements process (tgroup) and use case (ucid) in Figure 24. Participants in the treatment group performed better than participants in the control group for each use case across all the four metrics.

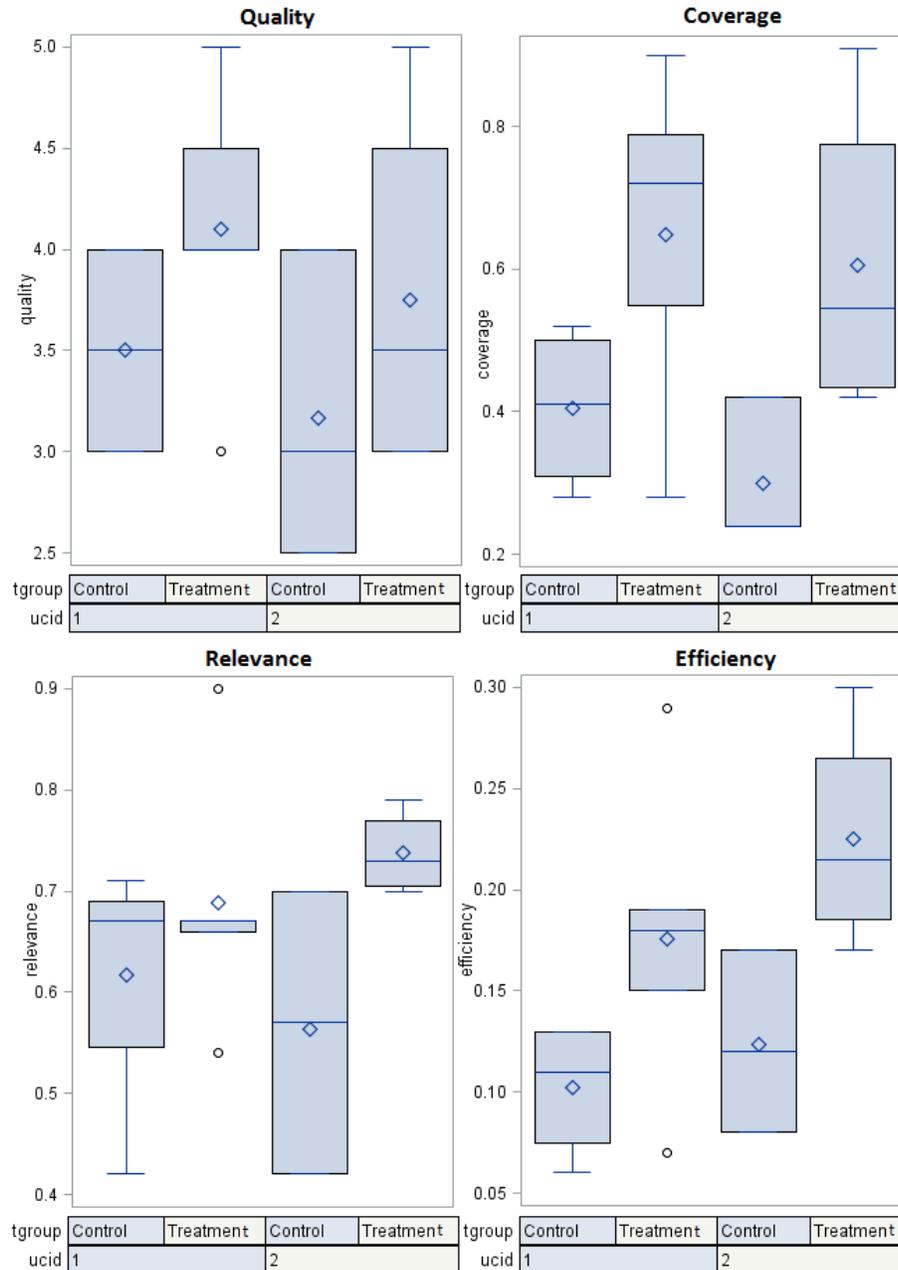


Figure 24. Box-plots with results from UCR15 across each factor and metric

### 8.3.5 Summary of Findings from Individual Studies

We compare the performance of participants in the treatment and control groups across the studies in terms of the four metrics for quality, coverage, relevance and efficiency to

address research questions RQ4.1-RQ4.4. In Table 49, we provide a study-wise summary of differences in mean scores between treatment and control groups for each of the four metrics (treatment mean – control mean). We also indicate whether the difference is significant or not. We plot the mean scores for the four metrics across studies for the treatment and control groups in Figure 25. The actual mean scores for each study are provided in previous sections.

Table 49. Difference between treatment and control group means across studies.

[\*Significant at  $p$ -value < 0.1; \*\*Significant at  $p$ -value < 0.05]

Study	$\Delta$ Quality (1-5)	$\Delta$ Coverage (0-1)	$\Delta$ Relevance (0-1)	$\Delta$ Efficiency (req./min)
NCSU13	-0.03	0.27 **	0.05	0.58 **
UT14	0.44	0.11 *	0.29 **	0.63
NCSU14	1.38 **	0.20 **	0.37 **	0.34 **
UCR15	0.58	0.27 **	0.12 *	0.36 **

Participants in the treatment group performed better than participants in the control group across all the four metrics in all the studies for both use cases. The differences were most evident for the metrics of coverage and efficiency as shown in Figure 25. We address RQ4.1-RQ4.4 based on the findings below.

**RQ4.1:** *What is the quality of security requirements elicited through the use of automatically-suggested security requirements templates?*

We found the least differences in the treatment and control groups in terms of the quality of the identified security requirements. The overall quality of the treatment group was 3.2 compared to 2.3 for the control group. Participants in the control group performed slightly better in terms of the quality of the identified security requirements for one of the use cases (UC1) in the original study NCSU13 (2.95 vs. 2.78), providing the only result where control group performed better. However, the difference is not statistically significant. In NCSU13, almost half of the participants in the treatment group for UC1 did not fill in the templates

which negatively impacted the overall quality score. For the replication studies, participants in the treatment group produced better quality requirements as compared to the control group. The difference is significant for NCSU14 with the largest number of participants (107). Participants in the treatment group for NCSU14 had additional support in filling the templates which could have positively impacted the quality scores.

***RQ4.2:** What is the coverage of security requirements elicited through the use of automatically-suggested security requirements templates?*

In all the studies, requirements coverage of the treatment group is better than the control group. The treatment and control groups differ significantly (at  $p\text{-value} < 0.05$ ) in terms of the coverage of the identified requirements in three of the four studies. If we consider a significance level of 0.1, treatment and control group differ significantly in all the studies for the metric of coverage. Participants in the control group identified 25% of the security requirements in the oracle overall whereas participants in the treatment group identified almost 46% of the security requirements in the oracle.

Missing requirements is a common problem in requirements engineering (Walia and Carver 2009) and we identified that participants in our studies also had missing security requirements, indicated by the low coverage scores. Overall, between 49 to 69% of the relevant security requirements in the oracle were not identified by the participants across studies. To some extent, this lack of security requirements coverage may be due to limited security expertise of the participants, limited resources and time constraints, and to the fact that no one individual may identify all applicable security requirements. One of the most commonly cited reasons for missing requirements is the lack of knowledge or expertise.

Given that participants in our studies were not security experts and not involved in the on-going development of the systems they analyzed, they may miss more requirements as compared to an individual with additional security expertise and domain knowledge.

***RQ4.3:** How relevant are the security requirements elicited through the use of automatically-suggested security requirements templates?*

The requirements identified by treatment group were more relevant as compared to the control group in all the studies. The difference was significant in two of the four studies at p-value  $< 0.05$ . If we consider significance level of 0.1, three of the four studies differed significantly in terms of the relevance of identified security requirements. In three studies, the relevance of treatment group is over 90% whereas in UCR15, with the smallest sample size, the relevance is the lowest at 71%. Overall relevance of treatment group is 89% compared to 62% of the control group.

***RQ4.4:** How efficient is the process of eliciting security requirements through the use of automatically-suggested security requirements templates?*

In all the studies, participants in the treatment group, using the automatically-suggested templates, were more efficient as compared to the control group. Treatment and control group differ significantly in terms of the efficiency of the requirements elicitation process in three of the four studies. Participants using the templates (treatment group) to elicit security requirements were also 57% more efficient as compared to the control group overall (1.23 vs 0.78).

For study UCR15, we selected use cases from the domain of online banking instead of healthcare. The results for UCR15 are similar to the findings of the NCSU13 where the

treatment group is significantly better than the control group for the metrics of coverage and efficiency. Security requirements templates support elicitation of applicable security requirements in the selected use cases for both healthcare and mobile banking domains. The automatically-suggested templates capture the security knowledge of multiple experts and can support the security requirements elicitation process as indicated by the results.

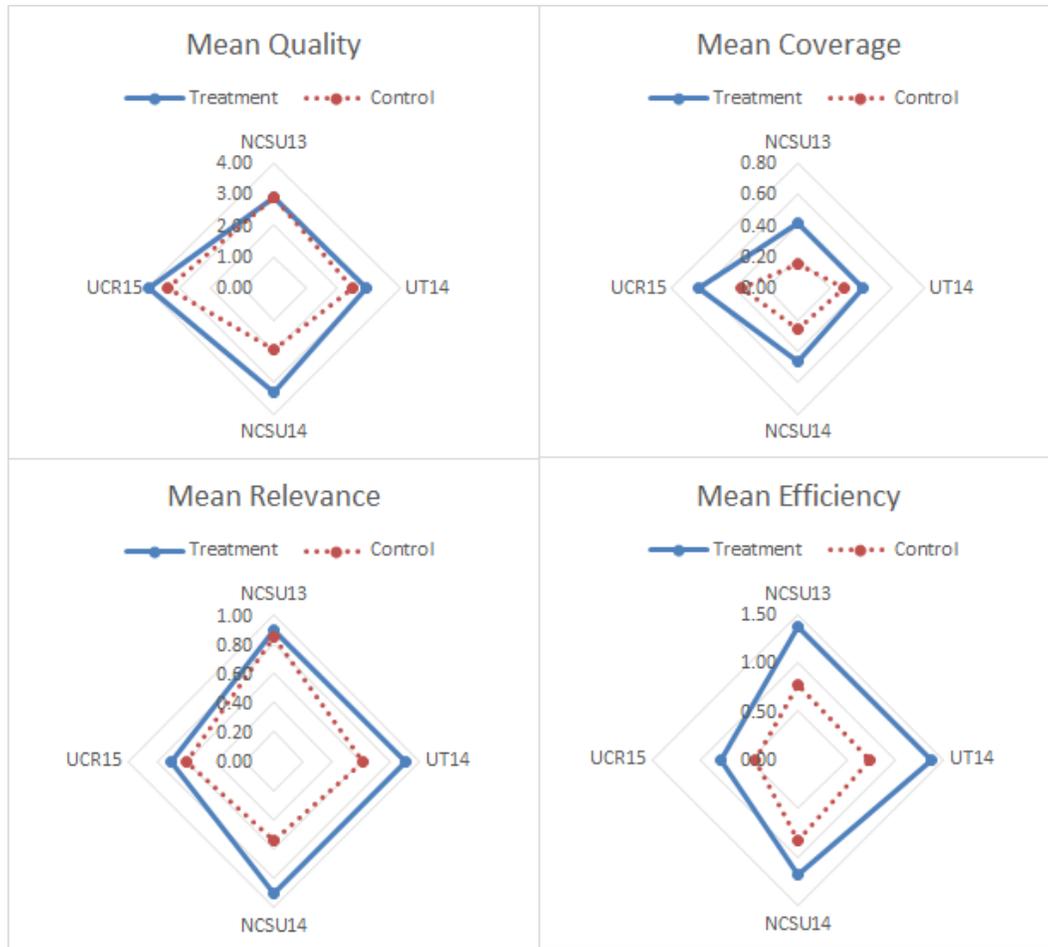


Figure 25. Mean scores for treatment and control groups across studies

## 8.4 Results Based on Analysis across Studies

We provide a comparative analysis of studies in terms of the metrics for quality, coverage, relevance and efficiency. We have raw data available from all the studies. Despite some differences across studies, we use the same or comparable metrics for computing the results. We perform a three-way ANOVA for unbalanced groups, adjusting for multiple comparisons (Tukey), to test the four null hypotheses for the combined data from all four studies. The third factor, in addition to the group and use case, is the study itself. Results with  $p < 0.05$  are considered significant for our analysis.

We combined the raw data from all the studies to perform the combined analysis of variance across the three factors: requirements process (treatment, control), use case (1-health, 2-health, 1-mobile, 2-mobile) and study (NCSU13, UT14, NCSU14, UCR15). The combined analysis is equivalent to performing an aggregate analysis of the individual studies, assigning weights based on the number of participants in each study. The combined results are most affected by the performance of participants in NCSU14 since NCSU14 has the largest number of participants, almost half of the total participants. We scaled the metric for efficiency for UCR15 by a factor of four, converting from templates per minute to requirements per minute, to make it comparable with other studies. Based on the results of combined analysis of data using a three-way ANOVA, as listed in Table 50, we found that participants in the treatment group performed significantly better than the control group across all the four metrics for quality, coverage, relevance and efficiency of the identified security requirements. Thus, we reject the null hypotheses H01, H02, H03 and H04 that the performance of participants in eliciting security requirements in terms of quality, coverage,

relevance and efficiency metrics is unrelated to the use of security requirements templates. Participants in the treatment group produced significantly better quality requirements (3.2 vs 2.3 for control group). Participants in the treatment group also identified significantly more requirements (~46% vs. ~25% for control group) and the identified requirements were more often relevant (~89% vs. ~62% for control group). Participants in the treatment group were also more efficient in identifying the security requirements (1.23 vs 0.78 for control group).

Table 50. Results of 3-way ANOVA for combined data from all studies.  
 [\* scaled for comparison]

Factor ↓ / Metric →		# of Obs.	Quality (1-5)		Coverage (0-1)		Relevance (0-1)		Efficiency (req./min)	
			Mean	p-value	Mean	p-value	Mean	p-value	Mean	P-value
Requirements Process (tgroup)	Treatment	111	3.2	<b>0.0014</b>	0.46	<b>&lt;0.0001</b>	0.89	<b>&lt;0.0001</b>	1.23	<b>0.0168</b>
	Control	94	2.3		0.25		0.62		0.78	
Use case (ucid)	1-health	98	2.76	0.7214	0.26	<b>&lt;0.0001</b>	0.79	0.6723	1.34	<b>0.0005</b>
	2-health	91	2.69		0.45		0.76		0.74	
	1-mobile	09	3.83		0.54		0.66		0.57	
	2-mobile	07	3.5		0.47		0.66		0.73	
Study (study)	NCSU13	50	2.88	0.3075	0.31	0.0808	0.88	<b>0.0002</b>	1.14	0.9372
	UT14	32	2.70		0.36		0.77		1.07	
	NCSU14	107	2.66		0.37		0.73		1.01	
	UCR15	16	3.69		0.51		0.66		0.64*	
<b>Significant Interactions (p &lt; 0.05)</b>			tgroup*study (<0.0001)		tgroup*ucid (0.0064)		tgroup*study (0.0001)		NONE	

We found significant differences between the two use cases from the healthcare domain for the metrics of coverage and efficiency. Due to the large number of total requirements in the oracle for UC1 in the healthcare domain, the coverage percentage is lower, but efficiency is higher as there are more requirements to be identified overall for UC1. Results also indicate significant differences between the original study NCSU13 and NCSU14 (the largest study) for the metrics of relevance. These differences are due to the lower relevance scores of

the control group in NCSU14 as compared to NCSU13. No other differences between studies are significant.

Considering interactions between the factors, we found significant interactions between requirements process and study for the metrics of quality and relevance as shown in Table 50. Specifically, we observed large variations in the quality of responses by both control and treatment groups across the studies. The relevance of responses also varied across the studies for both control and treatment groups. In terms of coverage, we found significant interaction between the requirements process and use case. No other interactions were significant at p-value  $<0.05$ . We provide group means and variance for each metric across all the groups in the studies in Table 51.

Participants in the treatment group produced significantly better quality requirements as compared to the control group overall. Participants in the treatment group also identified significantly more requirements, almost twice as many as control, and the identified requirements were more often relevant. Participants in the treatment group were also more efficient in identifying the security requirements with the support of automatically-suggested templates.

Table 51. Group means and variances for the combined data from all the studies.

Study	Use case	Metric	Control			Treatment		
			# of Obs.	Mean	Std. Dev	# of Obs.	Mean	Std. Dev
NCSU13	1	Quality	10	2.95	1.09	16	2.78	0.97
		Coverage	10	0.12	0.06	16	0.24	0.13
		Relevance	10	0.86	0.31	16	0.90	0.06
		Efficiency	10	1.15	0.68	16	1.70	1.52
	2	Quality	10	2.85	0.82	14	2.96	1.22
		Coverage	10	0.19	0.07	14	0.62	0.31
		Relevance	10	0.84	0.24	14	0.91	0.16
		Efficiency	10	0.40	0.13	14	1.00	0.39
UT14	1	Quality	9	2.56	0.68	9	2.94	0.88
		Coverage	9	0.29	0.14	9	0.31	0.18
		Relevance	9	0.66	0.18	9	0.88	0.09
		Efficiency	9	0.95	0.70	9	1.71	2.94
	2	Quality	6	2.33	0.61	8	2.88	0.99
		Coverage	6	0.31	0.14	8	0.52	0.24
		Relevance	6	0.53	0.30	8	0.92	0.10
		Efficiency	6	0.41	0.33	8	0.99	0.69
NCSU14	1	Quality	25	2.08	1.00	29	3.26	0.99
		Coverage	25	0.19	0.11	29	0.35	0.25
		Relevance	25	0.60	0.32	29	0.89	0.14
		Efficiency	25	1.07	0.81	29	1.46	0.95
	2	Quality	27	1.83	0.91	26	3.40	0.63
		Coverage	27	0.33	0.23	26	0.60	0.17
		Relevance	27	0.49	0.29	26	0.93	0.09
		Efficiency	27	0.62	0.46	26	0.87	0.47
UCR15	1	Quality	4	3.50	0.58	5	4.10	0.74
		Coverage	4	0.41	0.11	5	0.65	0.24
		Relevance	4	0.62	0.13	5	0.69	0.13
		Efficiency	4	0.41	0.14	5	0.70	0.32
	2	Quality	3	3.17	0.76	4	3.75	0.96
		Coverage	3	0.30	0.10	4	0.61	0.23
		Relevance	3	0.56	0.14	4	0.74	0.04
		Efficiency	3	0.49	0.18	4	0.90	0.22

## 8.5 Synthesis based on Differences Introduced among Studies

Studies differed in terms of participants, empirical setting, time and support available to participants in performing the task. We have summarized these differences in the context

factors in Table 41. We can assess if these factors impacted the results to address the additional research questions RQ4.5-RQ4.7. We present the synthesis of results from the original experiment and subsequent replications below.

### **8.5.1 Filling the Details in Security Requirements Templates**

***RQ4.5:** Are participants more inclined to fill in the templates when additional support to fill the templates is provided by explicitly indicating subject, action and resource elements in the input requirements?*

We hypothesize that providing additional support in filling templates by explicitly indicating subject, action and resource elements in the input requirements will lead to a higher percentage of participants filling in the templates.

Participants in the treatment group for NCSU14 and UCR15 received additional support in filling templates as we explicitly indicated the subject, action and resource for each of the input sentences. Participants in both studies also received course work credit based on the quality of their responses as opposed to just a participatory grade in the original study, NCSU13. For NCSU14, the course work credit was minimal whereas for UCR15, it was a significant percentage of the final grade. Participants in UCR15 also had the most dedicated time for the task.

We looked at the percentage of participants who filled the templates in the treatment group for different studies. In the original study, NCSU13, only 62% of the participants filled in the templates. In UT14, only 59% of the participants filled in the templates. In both cases, participants did not have support in filling the templates. In NCSU14, 78% of the participants filled in the templates which is an increase of ~26% as compared to NCSU13. In UCR15,

100% of the participants filled in the templates. These results suggest that participants might be more inclined to fill in the templates with the additional support. However, participants had stronger motivation for NCSU14 and UCR15 as compared to NCSU13 and UT14.

To control for motivation, we ran a confirmatory experiment at UT in October 2015, UT15, as an in class exercise with compulsory participation but no impact on grading. The goal of UT15 was to see whether participants in the treatment group fill-in the templates if we provide support in filling the templates, but not a lot of time or incentive. The experiment was setup similar to NCSU14 and UCR15 and was only intended to investigate RQ4.5 further. Participants in UT15 had support in filling templates, similar to NCSU14 and UCR15. However, the participants had limited time and did not have strong motivation. Using the same metrics that we have used for the other studies, we found that 15 of the 18 participants (83%) in the treatment group filled the templates despite lack of strong motivation and time on task.

The mean quality score of the treatment group is impacted positively if more participants fill in the templates. The mean quality of participants in the treatment group in NCSU14 was significantly better than the control group (3.33 vs. 1.95 of control) and also better than the treatment group in NCSU13 (2.86). The overall quality of responses in UCR15 was better than all other studies as shown in Table 44. The improved quality can be attributed to a number of factors including the additional support and considerable motivation to perform well, different problem domain, or the fact that participants spent the most time on task, two to five times more than other studies.

Strong motivation and more time on task are only partial drivers to increase the rate of filling in the templates. Providing additional support in filling templates by explicitly indicating subject, action and resource elements in the input requirements leads to a higher percentage of participants filling in the templates.

### **8.5.2 Differentiating between Relevant and Extraneous Templates**

*RQ4.6: Can participants differentiate whether a suggested security requirements template is relevant to the given use case scenario?*

We hypothesize that participants will consider whether a suggested template is relevant to the given use case scenario when selecting applicable templates.

In addition to the relevant security requirements templates, participants in the treatment group for NCSU14 and UCR15 were intentionally suggested extraneous templates. We can qualitatively examine if the participants selected relevant templates or extraneous ones as well. Based on the relevance scores given in Table 44, participants in NCSU14 and UCR15 identified less relevant requirements overall. Looking at the relevance scores for only the treatment groups, 90% of the security requirements identified in the NCSU13 were relevant as compared to 91% for NCSU14 and 71% for UCR15. Participants in NCSU14 and NCSU13 identified almost the same percentage of relevant requirements indicating that participants did not randomly select the suggested templates in NCSU14 and that participants may be able to differentiate between good and bad suggestions. The relevance of requirements is lower for UCR15 compared to NCSU13. In addition to the lack of knowledge about presence of extraneous templates, this difference can be attributed to the

different problem domain, more time on task and the instructions to identify as many security requirements as possible.

We also looked at the counts of extraneous templates selected by the participants in both studies. The four use cases analyzed in the studies have between 21 and 34 relevant templates, as shown in Table 43. In NCSU14, 26 of the 55 participants (47%) in the treatment group selected at least one extraneous template. The 26 participants selected 4 extraneous templates on average (~2 extraneous templates per participant for the treatment group overall). In UCR15, all 9 participants in the treatment group selected at least one extraneous template with an average of 8 extraneous templates selected by each participant. Participants in UCR15 selected four-times the number of extraneous templates on average as compared to NCSU14 and spent thrice the time on task. Consequently, the relevance score for UCR15 was the least (71%) whereas relevance score in all other studies is over 90% for the treatment group.

Based on the feedback from the participants, as discussed in Section 8.7, participants considered the templates to be a good starting point for identifying security requirements. Although participants were asked to select the templates they thought were applicable, some participants may consider extraneous templates as valid suggestions if they found most of the other suggested templates relevant.

Participants may be able to differentiate whether a suggested security requirement template is relevant or extraneous to the given use case scenario. However, extraneous suggested templates may confuse the participants if they consider them a good starting point and always potentially correct suggestions. Spending more time on task may also lead to identifying some extraneous security requirements.

### **8.5.3 Impact of Task Time on Study Outcomes**

*RQ4.7: Are there context factors, such as more time on task, which are conducive to producing better outcomes overall?*

We hypothesize that differences in context factors, such as more time on task, account for some of the differences in findings across the studies.

UT14 was conducted as a take-home assignment with one week to complete. NCSU14 was conducted as an in-class activity of 60 minutes as was NCSU13, but participants had slightly stronger motivation to produce better outcomes as compared to NCSU13. UCR15 was conducted as an in-class activity, however participants had 180 minutes to work on the task as compared to 60 minutes in the original study and had strong motivation to produce good quality outcomes.

In UT14 and NCSU14, participants identified almost 36% and 38% of all the requirements in the oracle respectively, slightly more than the 31% identified for NCSU13 as shown in Table 44. Participants in UCR15 identified 51% of all the requirements on average, which is an increase of almost 64% as compared to the NCSU13. Overall, participants in UCR15 spent the most time on task, 102 minutes on average, which is almost five times that

of NCSU13. The significant increase in the coverage of security requirements identified in UCR15 could be due to other factors as well, such as different problem domain or strong motivation to perform well.

Participants may use the available time as cue for the expected time they ought to spend on the task. Another factor that may influence the time spent on task is the expected credit or reward. Participants may look at both the available time and expected credit to decide how much of the available time to spend on task, especially if the time available is not dedicated time for task. With comparable credit for the task in NCSU13 and UT14, participants in UT14 spent more time on task on average (20 vs 47 minutes) even if the time was not a dedicated slot for the task. However, time spent in case of UT14 was still less than time spent in UCR15 where availability of more dedicated time coincided with higher credit for the task as well.

Considering the overall mean relevance and coverage scores for the studies, with more time spent on task, the mean coverage scores increased but the mean relevance scores went down as shown in Table 44. For studies UT14 and NCSU14, the decrease in relevance can be attributed to the performance of the control group as treatment group had high relevance scores (over 90%). However, for UCR15, the decrease in relevance is also due to the performance of the treatment group where relevance is 71%. This may be attributed to the availability of extraneous templates and availability of more time on task. We provided extraneous templates to participants in NCSU14 as well, but the relevance scores didn't go down. Under time constraints, as in NCSU14, participants may select fewer templates that seem the most relevant and thus irrelevant templates are mostly left out as well as some

relevant templates. Whereas when ample time is available, as in UCR15, once the participants have identified most relevant security requirements, they may try to improve the response by selecting additional templates, consequently increasing the chance of selecting irrelevant templates as well. This can explain the simultaneous increase in coverage (63%, whereas other studies have between 41 – 46%) and decrease in relevance (71%, whereas other studies have over 90%) observed in the treatment group in UCR15. However, we don't have a reliable way of knowing the order in which requirements were identified by each participant.

In terms of efficiency, participants in NCSU13 were more efficient as compared to participants in the subsequent replications as shown in Table 44. The results indicate that although participants in UT14, NCSU14 and UCR15 identified relatively more requirements than NCSU13, they were not more efficient and identified some extraneous requirements.

We plot the relation between tasktime (in minutes) and the metrics for quality, coverage and relevance in Figure 26 for the combined data of 205 participants across the four studies. By definition, tasktime and efficiency are negatively correlated. Tasktime was much larger for UCR15 as compared to other studies as visible in Figure 26. For NCSU14, the study with the largest number of participants, most of the responses are clustered towards higher relevance and lower coverage. If we plotted the figure grouped by treatment and control groups instead, we would see that treatment group participants are clustered towards higher relevance and control group participants are clustered towards lower coverage.

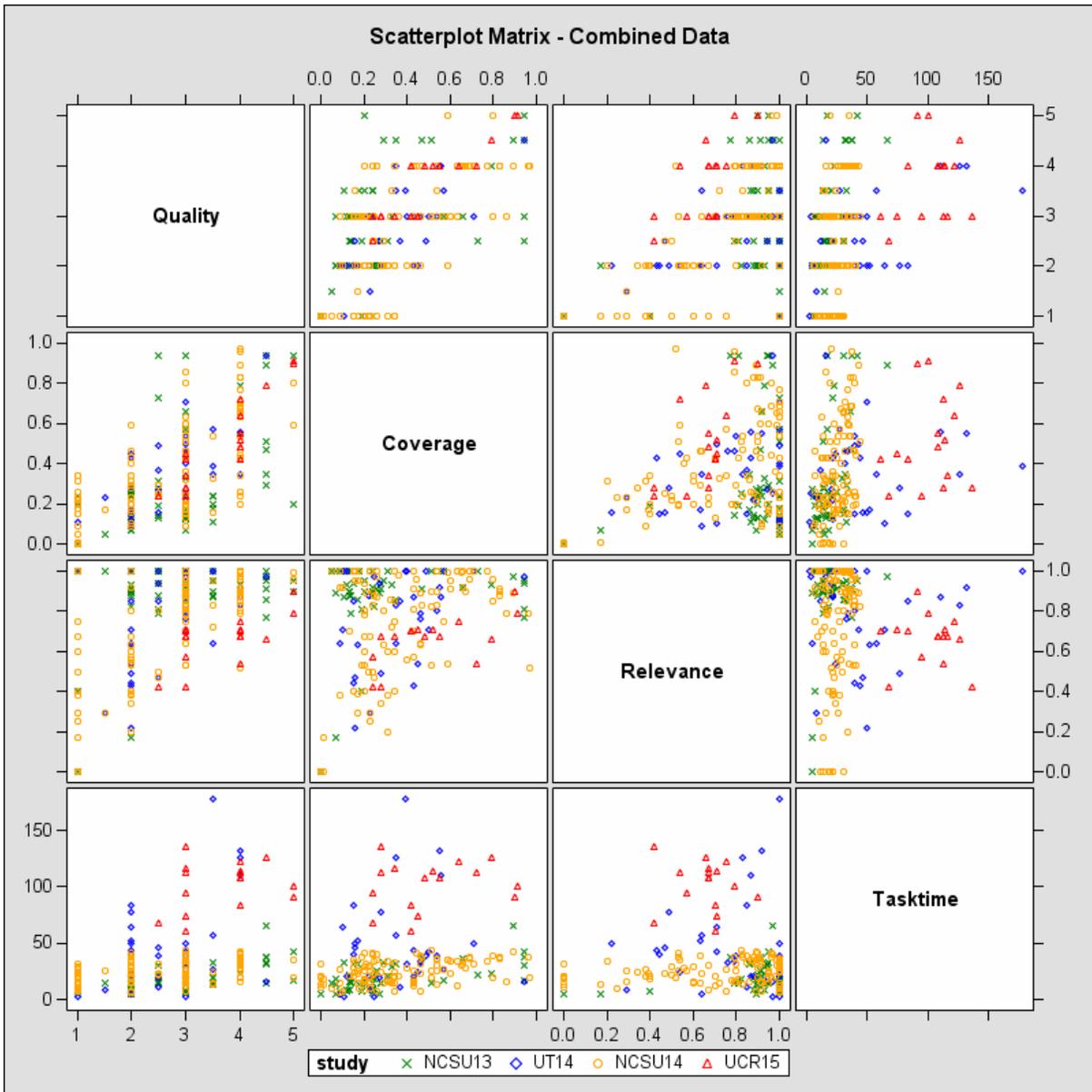


Figure 26. Relation among tasktime and metrics of quality, coverage and relevance for all participants

We identified a significant positive correlation between tasktime and quality of identified requirements overall (Pearson correlation coefficient of 0.39 at p-value < 0.0001). We also found a significant positive correlation between tasktime and coverage of identified requirements overall (Pearson correlation coefficient of 0.29 at p-value < 0.0001), similar to

our findings in NCSU13. We did not identify any significant correlation between tasktime and relevance (Pearson correlation coefficient of -0.04, p-value = 0.5756). In terms of relation among the metrics, quality is positively correlated with coverage (Pearson correlation coefficient of 0.64 at p-value < 0.0001) and relevance (Pearson correlation coefficient of 0.48 at p-value < 0.0001). Coverage and relevance metrics are also positively correlated (Pearson correlation coefficient of 0.32 at p-value < 0.0001), indicating that the participants who performed well, performed well across multiple metrics. There is no apparent correlation between other data points.

Participants who performed well, performed well across multiple metrics. Participants may look at both the available time and expected credit to decide how to approach the task. Under time constraints, participants may select fewer templates that seem the most relevant. Spending more time on task can lead to identifying more relevant security requirements. However, some participants may spend the additional time in improving the quality of the identified requirements (filling in templates, using reference material) or in identifying some irrelevant requirements.

## **8.6 Breakdown of Identified Security Requirements**

We group security requirements by the properties that will be supported if a system satisfies the security requirement. For example, security requirements related to ensuring access control over sensitive resources support the property of confidentiality. Similarly, security requirements related to logging user activities will support the property of accountability. The first three studies, NCSU13, UT14 and NCSU14, involved identifying

implied security requirements based on use cases selected from the domain of healthcare. The fourth study, UCR15, involved identifying implied security requirements based on use cases selected from the domain of mobile banking. For all the four use cases, we computed the number of requirements in the oracle (Table 43) for each security property. We also computed how many of the requirements in the oracle were identified by the participants on average in the treatment and control groups for each property as shown in Figure 27.

The treatment group not only identified more security requirements overall, but also more security requirements for each security property as compared to the control group in general. The security requirements related to the properties of confidentiality and accountability were the most frequently identified by both treatment and control groups and were also the most common requirements in the oracle for all the use cases as shown in Figure 27. Looking at the different types of confidentiality requirements identified, majority of the participants identified requirements related to ‘enforcing access privileges’ in both groups. Majority of the participants in the treatment group also considered requirements for confidentiality during storage and transmission as well as monitoring activity of locations where sensitive information is stored. A very small number of participants in the control group considered these additional confidentiality requirements. We observed similar trend for the accountability requirements where almost all of the participants in both groups identified requirements to log one or more of the user actions (e.g., documenting office visit, updating office visit). Moreover, almost everyone in the treatment group considered requirements for integrity of log files whereas almost everyone in the control group ignored these requirements. These findings indicate that participants in the control group may consider

only the most obvious security requirements (e.g., access control, logging of activities) for a given security property whereas participants in the treatment group are able to consider additional requirements as well.

For the healthcare domain, the requirements related to privacy were identified by the treatment group, but not as often by the control group. The oracle for UC1 had a fairly large number of integrity requirements but only a small fraction of these requirements were identified by the participants in both groups. We had the least number of requirements related to identification and authentication and availability in the oracle for UC1-Health and UC2-Health. Almost everyone identified some requirements related to identification and authentication. Whereas almost no one identified requirements related to availability.

We also provide the breakdown of identified requirements for the mobile banking domain in Figure 27. The requirements identified by the participants in treatment and control groups mirror the breakdown of requirements in the oracle. Treatment group identified more security requirements for each security property except for UC2-Mobile where control group identified slightly more requirements for integrity as compared to the treatment group.

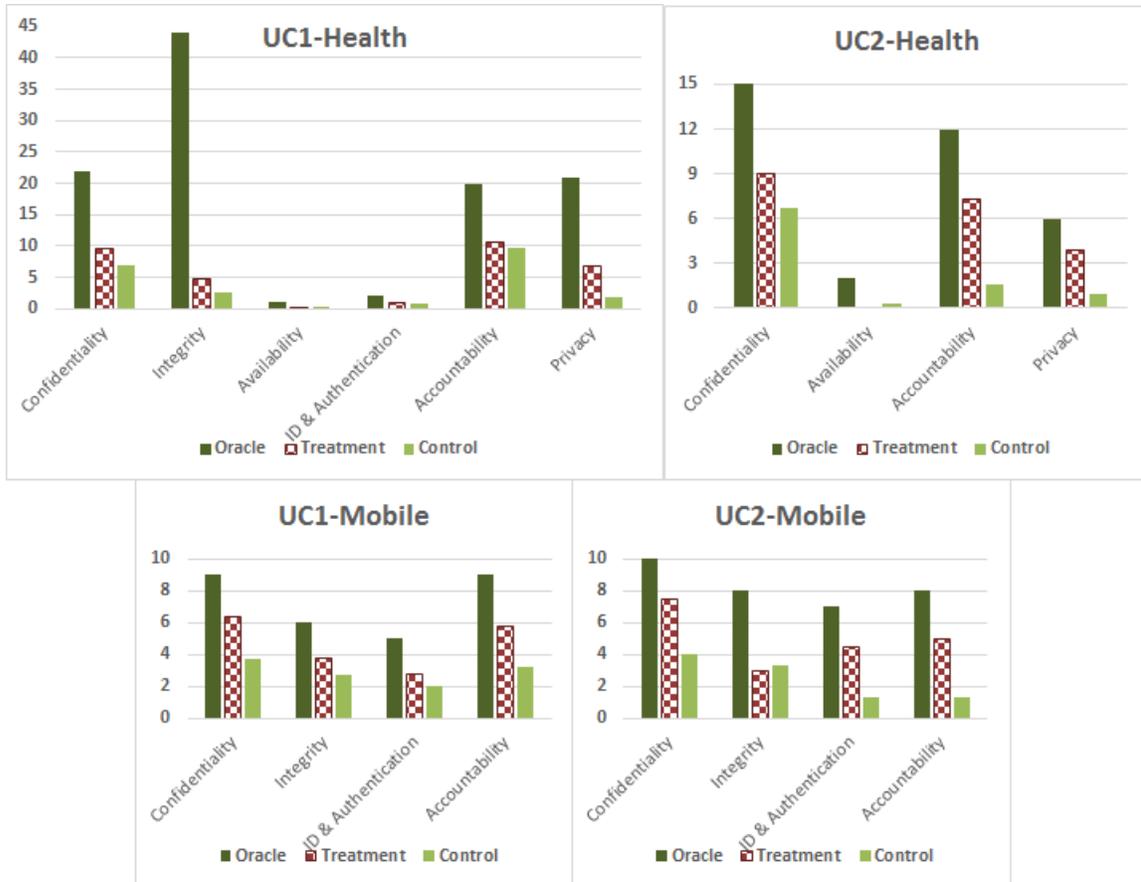


Figure 27. Requirements in the oracle identified per security property

The treatment group not only identified more security requirements overall, but also more security requirements for each security property as compared to the control group in general. Participants in the control group may consider only the most obvious security requirements (e.g., access control, logging of activities) for a given security property whereas participants in the treatment group are able to consider additional requirements as well. We observed a focus on identifying security requirements related to the properties of confidentiality and accountability in both healthcare and mobile banking domains.

## 8.7 Feedback from Participants

We solicited feedback from participants in the treatment group on the use of the security requirements templates and the generated security requirements in a post-task survey. The feedback was voluntary for all the studies.

Overall, almost 80% of the 111 participants in the treatment group provided a favorable opinion related to the use of security requirements templates. We did not get feedback regarding templates from 8 participants (7% of 111). Of these, six participants did not provide answers for the optional feedback whereas one participant in UT14 and NCSU14 each did not use templates. The participant in UT14 not using templates still copied the example security requirements generated from templates provided as part of the reference material. The participant in NCSU14 who chose not to use the templates identified security requirements based on keywords in the sentence according to survey response. The participant was the least efficient in the treatment group but not influential on the results. The participant did not provide a reason for not using templates.

Based on the feedback from participants related to the security requirements templates, we identified the following response categories. We provide the frequency of responses for each category in Figure 28.

- *Provide a good starting point (28%)*: provide a direction for developing secure systems; can use the templates and add additional requirements if needed; easy to start defining security requirements with the templates;
- *Good coverage and applicability (12%)*: classification based on security properties is comprehensive and covers all main areas of concern; holistic and ensure that system

is analyzed from all perspectives related to security; easy to apply to the given system;

- *Help in thinking about security (22%)*: allows users with minimal knowledge to use and understand the templates; helped in thinking about context of the sentences in the use case; helped in considering more security requirements than would have otherwise;
- *Help in phrasing requirements (18%)*: saves time by making it easy to type and edit requirements; provided reusable requirements; help in thinking how to go about writing the requirements; helped put forward ideas in formal manner; better if can be auto-filled (e.g., input field for resource, subject, action as these fields are same within a template);
- *More templates and support (6%)*: more template choices would be even helpful; good but not exhaustive; auto-filling the templates to generate security requirements will be good;
- *Apply with caution (7%)*: not all requirements in the template may be relevant; should not be considered as an exhaustive list; might lead to choosing templates arbitrarily, without carefully thinking about security requirements;
- *Not used / answered (7%)*: participant did not provide feedback related to templates.

The templates helped participants think about security and provided a starting point to identify relevant security requirements. Participants considered the six security properties, which are used to classify the templates, to be holistic in supporting a comprehensive analysis of security concerns. However, the templates should not be viewed as an exhaustive

list, as pointed out by some of the participants. Some participants also indicated the need to apply the templates with caution as all templates may not be relevant and availability of templates may lead an individual to choose the templates arbitrarily.

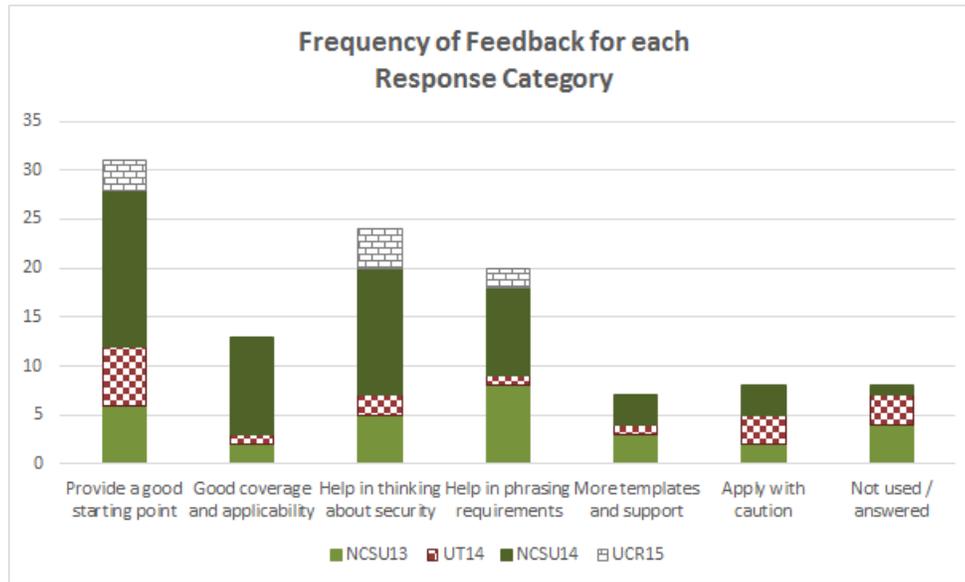


Figure 28. Feedback related to the use of security requirements templates

We asked the participants to summarize the approach they used for identifying security requirements in a post-task survey.

In the treatment group, all the participants used suggested templates to formulate the security requirements, with one exception. The only participant who did not use the templates indicated a methodology based on identifying possible attacks. Participants in the treatment group reported following approaches to formulating security requirements:

- *Considered suggested templates*: think how templates map to scenario; identify keywords and key phrases; identify resources, action and users; fill in the templates;

- *Considered security properties*: keep properties at the back of mind; see if property might be violated somewhere;
- *Individual expertise*: use intuition to guide the analysis; familiarity with the given scenario; consider potential risks in that scenario; consider assets that need to be protected; think like an attacker, what an attacker can gain;

In terms of individual expertise, one participant reported using an approach based on intuition to identify security requirements. Two participants reported that they were familiar with the system from which the use case scenario was used in the study, so they leveraged their background and experience with the system. However, the two participants who were familiar with the system identified less than the average number of security requirements in their group, indicating that experience did not help.

Participants in the control group reported more diverse approaches to identify security requirements, as listed below:

- *Keywords*: identify keywords from the sentences that indicate a need for security (e.g., logging, sends data, process data, identification and authentication)
- *Identify security properties*: identify applicable security properties from the given scenario;
- *Example security requirements*: identify relevant examples of security requirements from the reference material; try to map example security requirements to the given scenario;
- *Individual expertise*: identify standard issues; security requirements and methodologies taught in the course (e.g., misuse cases, secure entry and exit points, secure the data);

Many participants in the control group seemed to manually iterate through steps similar to the automated process provided for the treatment group, such as identifying security properties from sentences and trying to map relevant example security requirements to the given scenario. Many participants also relied on their background knowledge and material covered during the course to help guide their analysis.

## **8.8 Lessons Learned Conducting the Experiments**

We present some of the lessons learned as part of conducting the original experiment and subsequent replications.

### **8.8.1 Encouraging Participation and Performance**

In all the studies, the task given to students was mandatory to complete for class participation, but participation in the study was not mandatory. Almost 93% of the students agreed to participate in the case of NCSU13. Similarly, almost all the participants agreed to allow the use of their responses for the study in subsequent replications. Students may be more willing to participate in a study when it is a relevant part of the educational experience.

In the original study, to keep the responses anonymous, we distributed a separate sheet of paper to all students asking: *Briefly explain what you learned as part of this classroom exercise.* All students who submitted the sheet to the teaching staff received participation credit for the class exercise, irrespective of their participation in the study or of the quality of their responses. The lack of a detailed grade on the exercise might have affected their motivation to spend more time and effort in identifying high-quality security requirements. To ensure that participants diligently work on a task, they need further incentive than just a participatory grade. Either the experiment setting needs to be changed to give dedicated time

during a lab or lecture to completely perform the task, or the participants need some sort of grades, scoring, or personalized feedback to place more focus on producing quality responses. We incorporated these lessons in the subsequent replications.

Further, we recommend that to ensure task completeness, participants either need a step-driven (i.e., wizard) approach to perform the task, or progress indicators to effectively identify remaining work. Such clues might enable participants to think more about the task and how much effort is expected of them.

One-third of the participants in the treatment group did not completely fill-in the templates to produce concretized security requirements. We improved the support available to participants in filling-in the templates which resulted in a higher percentage of participants who filled-in the templates in subsequent replications. We may also check that participants have provided missing information (such as resources and actions) before they can add the requirements suggested by the template to the final set of identified security requirements

### **8.8.2 Communicating with the Original Experimenters**

The replications were conducted in close collaboration with the original experimenters. All the experiment material and online tool setup was provided by the original researchers. Moreover, the first two authors evaluated the results for UT14 and NCSU14 replications, as was done for NCSU13. Conversely, one of the original evaluators and researchers from UCR evaluated the responses for UCR15. The first author maintained communication with the researchers conducting the replications over email and voice calls throughout the planning, conducting, evaluating and reporting stages of the experiment. Two of the original researchers, including the principal investigator of the original experiment, also met in person

with the researchers conducting the replications prior to the conduct of the replications to discuss the research effort behind the original experiment.

Based on our experience, maintaining close communication links is important for successful conduct of replication studies. The researchers need to communicate the context factors associated with the original experiment in detail as well as the tacit knowledge gained by conducting the original experiment. Details about experiment design, context factors and evaluation process should also be reported in research publications. Experiment material and artifacts should be made available for any other researchers who are interested in performing subsequent replications. In addition to communicating the details of the original experiment, feedback and observations from the researchers conducting the replications is valuable in understanding the findings.

### **8.8.3 Minimizing Technical Setup**

Minimizing the effort needed to setup the experiment supports the replication effort. As observed by Riaz et al. (Riaz, Breaux, and Williams 2015), a replication package, such as the online tool and experiment material provided by the original researchers, is conducive to the conduct of subsequent replications. In our case, the researchers conducting the replications required no technical setup at their end. Each participant needed a computer or laptop to access the online site for participating in the study. On the other hand, if a tool is used to conduct a study, the reliability of the tool is important for the successful conduct of the study. Technical support should be at hand in case problems arise. Moreover, contingency plans, such as availability of offline options to complete the task, should be put in place in case the tool is not available.

#### **8.8.4 Managing and Reporting Emerging Contexts**

Replicating a study, even with all the experimental material and tools available, requires the acquisition of a considerable amount of knowledge. In our case, before conducting the replications, the researchers studied the original experiment, read the additional reference material and tested the tool to familiarize with the environment prior to the conduct of the replications. Despite all the preparation, unexpected events and situations may still arise during the conduct of the study which should be managed and reported.

The replication at UT was initially planned as an in-class activity. However, almost half of the students did not have access to a computer as observed by the instructor at the start of the class. The researchers at UT decided to conduct the study as a take home activity instead. The findings can be interpreted in the context of a take home activity where participants may have more time overall to work on the task but not dedicated time for the task.

For the replication at UCR, almost half of the participants in the control group provided responses in Spanish. Researchers at UCR translated the responses and also participated in evaluating the responses. Researchers conducting replications should be prepared to handle such emerging situations and report the potential impact of changes in the experimental context on the findings of the study.

#### **8.8.5 Developing Shared Insights**

Availability of data from multiple studies with small differences in context factors provides an opportunity to develop insights through qualitative analysis of the combined data, looking at each study's findings in the particular context. Participants in the studies were enrolled in three different graduate courses related to software security and metrics. All

the studies were well-integrated with the coursework requirements (J. C. Carver, Jaccheri, and Morasca 2010) however participants had varying degrees of motivation to perform well. In addition to different participants and courses, we modified the context of the studies in terms of the support in filling the templates as well as the presence of extraneous templates. We were also able to explore the effect of time on task on various metrics. Each group of researchers had unique insights into the study they conducted. By sharing these insights, we addressed additional research questions (RQ4.5-RQ4.7) to explore potential reasons for similarities and differences observed across studies.

#### **8.8.6 Working with Diverse Groups of Participants**

Participants in the studies were graduate students from three different countries, enrolled in difference courses, and speaking different languages. Different sets of contextual issues, mainly background, previous knowledge and cultural issues, may exist which we did not factor into our results. For instance, due to language differences, researchers at UCR prepared equivalent instructions in Spanish to explain the purpose of the study in the native language to the participants. Moreover, some participants in the control group responded in Spanish and also provided feedback in Spanish which was translated by the researchers at UCR.

As observed by researchers at UCR, students may feel a sense of competition if they are aware that other students in different countries have performed the task of identifying security requirements. Some of the students at UCR also showed interest in knowing about the results across universities at the end of the study. Moreover, participants in larger classes may behave differently than participants in smaller classes. For instance, in NCSU14 with

the largest class size, the researchers observed that many participants were confused about the user interface despite the availability of written instructions addressing the questions asked by the participants. Whereas we did not observe such trend in NCSU13.

## **8.9 Threats to Validity**

We report various threats to validity (Wohlin et al. 2000) that we considered or mitigated during the design and execution of the four studies.

### **8.9.1 Internal Validity**

*Selection:* The effect of natural variation in human performance can influence the study outcomes. In all the four studies, we used the round-robin assignment approach to randomly assign participants to treatment and control groups, as well as to one of the use cases. Unbalanced groups in terms of participant expertise in the given task could result. However, based on the background information of participants, treatment and control groups were evenly balanced in terms of expertise across all studies.

*Instrumentation:* The effect of experiment artifacts can influence the study outcomes. We observed significant differences in the mean coverage scores for the use cases from healthcare domain (UC1, UC2) used in studies NCSU13, UT14 and NCSU14. UC1 has 110 unique security requirements in the oracle compared to 35 unique security requirements for UC2. Participants identifying security requirements for UC1 would have found a smaller percentage of the total security requirements in the oracle even if they found the same absolute number of security requirements as UC2. However, participants in the treatment group identified significantly more security requirements as compared to the control group,

independent of the use cases, in three of the studies as well as in combined analysis, indicating that our findings related to coverage of security requirements (RQ4.2) still hold.

*Diffusion or imitation of treatment:* Three of the four studies were conducted as an in-class activity where participants did not have the opportunity to share information about various treatments so this threat is minimal overall. Moreover, participants were not aware that they will be assigned to different groups (treatment and control) to further minimize the risk of treatment diffusion and biased responses. However, study UT14 was conducted as a take-home activity and control group participants could have gotten the templates from fellow treatment group participants. We examined the responses by control group in UT14 to see if they resembled closely with treatment responses. However, we did not find any evidence of treatment diffusion across the groups. Some participants in the control group specified security requirements based on the examples available to them while others used their own words, similar to our observation in other studies.

*Testing and training:* As part of the reference material, we provided example security requirements to the control group generated from the same requirements templates available to the treatment group. Many participants in the control group used the example requirements as a basis for specifying their own security requirements and the examples might have introduced a bias by priming the participants towards certain security requirements. We provided the same examples to the treatment group as well (in addition to the templates), so any potential bias is similar for both groups. Moreover, the process for creating the oracle is similar to the process for identifying security requirements used by the treatment group which may introduce a bias. To minimize the potential bias, we provided control group with

examples of security requirements generated from the same templates used to create the oracle.

### **8.9.2 External Validity**

*Representativeness of sample population:* The sample population should be representative of the population for which we want to draw conclusions based on the study outcomes. Participants in the study were enrolled in four different graduate courses across three different universities in two different continents. In three studies, participants had been exposed to concepts related to security principles, practices and tools. In the fourth study, participants were enrolled in a non-security related course. Participants are fairly representative of the graduate students in computer science. Based on the feedback, about 72% of the participants had less than 1 year of academic experience related to security and about 85% had less than 1 year of work experience related to security. Rest of the participants had between 1-2 years of academic and work experience with only a handful of candidates having more than 3 years security experience. Participants can be considered representative of entry-level, non-expert software and security practitioners accordingly.

*Task representativeness:* The task should be representative of how security requirements are identified in practice. Each participant identified security requirements based on a single use case scenario. Additional context for the system and problem domain may help in considering additional security requirements.

*Templates representativeness:* The templates should be representative of how security requirements are specified for different systems. Through the replications, we have demonstrated the applicability of the templates in identifying security requirements for four

different use case scenarios selected from two different domains. Our findings indicate that we may use the templates to identify security requirements for other scenarios and potentially other application domains that have similar security properties as covered by the templates.

*Experimental constraints that limit realism:* Participants used a limited amount of time to complete the task, which may affect the quality and coverage of identified security requirements. Moreover, participants work individually and may not be able to identify all the applicable requirements in a limited amount of time.

### **8.9.3 Construct Validity**

*Hypothesis guessing:* Participants may try to guess the purpose of the experiment which could bias their performance. In the first three studies, participants were not aware of the existence of treatment versus control groups or whether they belonged to different groups. Participants were told only that they were supposed to perform the task of identifying security requirements based on a given use case. Thus single blinding was used to minimize biases towards viewing one requirements elicitation process favorably as compared to the other. However, participants in UCR15 had access to the research paper documenting the original experiment and they may have read the paper prior to the experiment. The problem domain for UCR15 was different than the original experiment so the participants could not have known the answers (i.e., which templates are in the oracle) however they may have known about the hypothesis. This threat is limited to UCR15.

### **8.9.4 Conclusion Validity**

*Reliability of measures:* Reliability of measures is an important consideration to draw valid conclusions about the outcomes. When measuring time spent on the task, we

automatically recorded every time a participant saved or resumed the task. The recorded timestamps helped us assess the actual time spent on the task at a more granular level, compared to self-reporting by participants. However, the measurement of time for UT14 might not be as accurate as other studies. In UT14, participants performed the task as a take-home activity. We record time based on when a participant opened the task screen and when the participant saved or submitted the task. If a participant opened the task and then switched to something else, we may get a lower efficiency score due to higher time on task recorded. Consequently, if a participant performed the task offline and then copied the responses back to the web page, we may get a higher efficiency score due to lower time on task recorded. We noticed one response in UT14 where efficiency was unusually high. However, removing that response from the analysis does not affect the significance of results for UT14 or the combined analysis. For take-home activities, a better approach would be to use a combination of recording time with the tool and comparing with the self-reported time by the participants. This threat is limited to the assessment of efficiency (RQ4.4) for UT14.

*Fishing and the error rate:* Fishing pertains to the threat of experimenters looking for a specific outcome. For instance, we might actively look for results that support the use of security requirements templates. Ideally, the evaluator should be blind to whether they are evaluating the responses for a participant in the treatment group or control group. However, as is often the case with experiments in software engineering, we could not employ double-blinding when reviewing the participants' responses. Determining whether a participant belonged to the treatment or control group was obvious, since participants in the treatment group used security requirements templates with standardized wording. In contrast,

participants in the control group specified requirements in their own words. Care was taken to minimize biases during the evaluation of the responses by devising quantitative measures whenever possible, having multiple independent evaluators and using a standard oracle created beforehand. Some participants in the control group for UCR15 provided response in Spanish that were translated by the researchers to English for the purpose of evaluation. Assessment of quality of responses might be impacted due to the translation (RQ4.1). However, native Spanish speakers took part in evaluating the responses using the same criteria as the other studies. This threat is limited to the control group in UCR15 and is minimal given the high inter-rater agreement between evaluators using Spanish response and corresponding English translation. In terms of the error rate, we adjusted the error rate for multiple comparison to maintain the desired significance level of results ( $p\text{-value} < 0.05$ ).

*Violated assumptions of statistical tests:* For UT14 and NCSU14, one of the ANOVA assumption related to homogeneity of variance doesn't hold for the metric of relevance (RQ4.3) due to the large variations in the relevance scores for the control group as compared to the treatment group. In both studies, we found the relevance scores of treatment group to be significantly better than the control group. However, the results for the relevance metric in these two studies may not be considered as reliable as other findings where ANOVA assumptions are met.

*Number of participants:* For UCR15, we have 16 participants divided in four groups and we may not be able to draw reliable conclusions due to limited number of participants. For instance, we have three participants in the control group for UC2 from the mobile banking domain. If one participant makes a mistake, that amounts to ~33% of the participants making

the mistake. However, the threat is limited to UCR15 and the findings of UCR15 are similar to other studies discussed in this paper having a larger number of participants.

### **8.10 Discussion and Considerations for Future Evaluation**

We have conducted three differentiated replications of a controlled experiment to evaluate the use of automatically-suggested templates in identifying implicit security requirements as compared to a manual approach without the guidance of templates. We presented information about the security properties implied by sentences in the given use case and suggested security requirements templates to consider when identifying applicable security requirements. Participants in the treatment group performed significantly better than participants in the control group in terms of the coverage of the identified requirements and efficiency of requirements elicitation process in three of the four studies. Participants in the treatment group also performed significantly better for the metric of relevance in two studies and metric of quality in one study. In the combined analysis of all the studies, participants in the treatment group performed significantly better than the control group across all the four metrics. We did not find a case where participants in the control group performed significantly better. Overall, participants using templates identified 84% more requirements and were 57% more efficient as compared to the control group. Almost 80% of the participants in the treatment group provided a favorable opinion related to the use of security requirements templates. Providing more dedicated time on task and strong motivation, as in UCR15, led to improvement in the overall quality of elicited requirements. The findings hold for the four different scenarios selected from the domains of healthcare and mobile banking,

indicating that the results may be generalizable across other scenarios, potentially from different domains, where security is an important consideration.

### **8.10.1 Differences in Use of Knowledge Sources based on Expertise**

An open issue is whether such results from students' experiments would also be broadly applicable to practitioners. A key observation is that security patterns or similar forms of knowledge sourcing from the community (such as security catalogues or guidelines) are mostly useful to people who are not experts. For example, Gray and Meister's survey of practitioners shows that knowledge sourcing is most sought and most effective when needed in conditions of high intellectual demand with respect to users' actual expertise (Gray and Meister 2004). De Gramatica et al. experiments with practitioners show that security catalogues can indeed equalize security experts, without a catalogue, to non-security experts, with a security catalogue (De Gramatica et al. 2015). Therefore, our experiments capture the case of most interest: knowledge sourcing does improve performance of non-experts such as students or junior practitioners in the field. The question remains open whether the performance of experts could be significantly improved by using the security requirements templates. The qualitative evidence from De Gramatica et al. seems to imply that experts and non-expert use knowledge sourcing in essentially different ways. Non-experts may use additional knowledge for finding information while experts may use it as a checklist for noting what may otherwise be forgotten. More experiments would be needed in this respect to develop further insights.

The automatically-suggested templates capture the security knowledge of multiple experts and can support the security requirements elicitation process as indicated by the

results. However, between 49 to 69% of the relevant security requirements in the oracle were not identified by the participants across studies. Despite the relatively low coverage scores, requirements coverage of the treatment group is better than the control group across all the studies. The coverage is significantly better in three out of the four studies as well as in the combined analysis. The overall lack of security requirements coverage may be due to limited security expertise of the participants, time and resource constraints, and to the fact that no one individual may identify all applicable security requirements. A few participants in the treatment group also voiced the concern that the templates should not be considered an exhaustive list, as reported in Section 8.7. Participants may tend to over-rely on the technique and overlook security requirements that are not part of the templates.

#### **8.10.2 Role of Additional Context and Domain Knowledge**

We only provided a use case scenario as input to the participants as a starting point for identifying the security requirements. However, additional resources such as security policies can also be provided as input to the participants and may guide the identification of applicable security requirements. However, current approach is based on assessing how well participants can identify security requirements only given the knowledge of the functional requirements (or use case) and security requirements templates as compared to a control group. We have conducted an industrial case study to evaluate the coverage of security requirements identified by our process for a software system, in comparison to a proprietary approach. The results, documented in subsequent chapters, provide evidence on how the process generalizes when applied with the help of security analysts without the time and

other experimental constraints. Moreover, using our process as complementary to other existing approaches is another direction for future work.

Based on our experience, maintaining close communication links between the original researchers and the researchers conducting the replications is important for successful conduct of the replication studies. Researchers need to communicate the context factors associated with the original experiment in detail as well as the tacit knowledge gained by conducting the original experiment. Moreover, researchers conducting the replications should be prepared to handle emerging situations and discuss the potential impact of changes in the experimental context on the findings of the study.

## 9 INDUSTRIAL CASE STUDY IN THE NETWORKS DOMAIN

In this chapter, we address the following research question:

- **RQ5:** *How effectively can our process be used to identify and specify security requirements for a software system?*

We conducted a case study of requirements for a real-world system from the networks domain to carry out an evaluation of SRD process in identifying security requirements for the software system. With the analysis, we can assess the comprehensiveness of our security requirements patterns in reference to the selected case and carry out gap analysis of the security requirements generated through our process in reference to the selected case. Consequently, we breakdown the overall research question in to the following two research questions:

**RQ5.1:** *How well do the security requirements, identified through a proprietary process, align with the security requirements patterns that we have identified?*

**RQ5.2:** *How well do the security requirements, identified through a proprietary process, align with the security requirements identified via our semi-automated process?*

### 9.1 Requirements Artifacts

The requirements artifacts for this study were provided by Cisco Systems, Inc. We received the following two types of artifacts from the industry, as listed in Table 52:

- Product requirements document (PRD) including the functional requirements for the system. We used this document as input to our semi-automated process for identifying security requirements.

- Product security baselines (PSB) document outlining the security requirements for the system. We used this document to validate our security requirements patterns and compare the output of our process against to carry out a gap analysis.

Table 52. Requirements Artifacts – Industrial Case Study

<b>Artifact Type</b>	<b>Size</b>
Product requirements document (PRD)	1218 sentences
Product security baseline (PSB)	108 security requirements

For the PSB, Cisco also provided a mapping of the security requirements to applicable NIST controls. We have developed our security requirements patterns from NIST controls as well and can leverage the mapping provided by Cisco in our analysis.

## **9.2 Coverage of Security Requirements Patterns in PSB**

***RQ5.1:** How well do the security requirements, identified through a proprietary process, align with the security requirements patterns that we have identified?*

We want to evaluate the comprehensiveness of our security requirements patterns for prevention, detection, and response and identify areas for which our patterns do not provide coverage of security requirements in the PSB document. We have developed the security requirements patterns by systematically analyzing and synthesizing NIST controls that support same security goals (Section 6.3). Cisco also provided a mapping between security requirements in the PSB and NIST controls. Each requirement could map to 0 or more NIST controls. Of the 108 security requirements in the PSB, 96 requirements generated 145 mappings to NIST controls whereas 12 of the requirements did not map to any NIST control based on the information available to us. Thus we have a total of 157 mappings.

We deferred to the mapping between PSB and NIST controls provided by Cisco whenever applicable. We leveraged the provided mapping to identify matching security requirements patterns during the initial mapping of security requirements in PSB to our security requirements patterns. We assign the security requirements in the PSB one of the following codes. If a requirement in PSB does not map to any of the NIST controls, we assign 'NA' to the requirement in the initial mapping (NA). If a requirement in the PSB matched to multiple NIST controls, and each of those controls is included in at least one of our patterns, then the patterns completely cover that security requirement (X). If the controls included in our patterns cover a subset of the controls to which a requirement maps, then the patterns partially cover the security requirement (P). Otherwise our patterns do not cover that security requirement (N).

After the initial mapping, we manually looked at the security requirements in the PSB that have not been completely covered by the patterns (P, N and NA). As a result, we manually analyzed 43 of the 157 total mappings. If during manual analysis, we found one or more applicable security requirements pattern, then we updated the code of the requirements accordingly. One researcher carried out the manual mapping at first and a second researcher reviewed the mapping resulting in 22 updates of the 43 manually analyzed mappings.

We provide a breakdown of the coverage of security requirements in the PSB based on our initial mapping as well as after the additional manual mapping in Table 53. Based on the initial analysis, 70 of the 108 security requirements were marked as covered by our security requirements patterns. After manual analysis, we identified 105 of the 108 security

requirements to be covered by our patterns. We identified one requirement related to the use of a specific operating system to be partially covered.

Table 53. Coverage of Security Requirements Patterns for Security Requirements in PSB.

	<b>Initial Mapping</b>	<b>First Manual Mapping</b>	<b>Second Manual Mapping</b>
<b>X:</b> Covered	70 (65%)	103 (~95%)	105 (97%)
<b>P:</b> Partially Covered	11 (10%)	0 (0%)	1 (~1%)
<b>N:</b> Not Covered	15 (14%)	5 (~5%)	2 (~2%)
<b>NA:</b> Not Applicable	12 (11%)	0 (0%)	0 (0%)
<b>TOTAL</b>	<b>108</b>	<b>108</b>	<b>108</b>

We provide a breakdown of the security requirements patterns that one or more of the security requirements in the PSB mapped to after the final mapping in Table 54. The requirements mapped to 31 different security requirements patterns related to preventing, detecting and responding to security breaches. Most of the requirements were related to preventing a breach of security with almost 27% of the requirements related to preventing unauthorized access to information (P-C\_I\_PR-1). Another 16% were related to managing the security of the identifiers and authenticators themselves (P-C\_I\_ID-1). For detecting security breaches, the most commonly mapped pattern was detecting malicious system modification attempts (D-I-1). For responding to security breaches, the most commonly mapped pattern was denial of service prevention and response (R-A-2). Some of the requirements mapped to multiple patterns so the sum of counts in the last column of Table 54 is not meaningful.

Table 54. Breakdown of Security Requirements Patterns that cover Security Requirements in PSB

Security Action	Scope of Pattern	Pattern Identifier and Name	Requirements mapped in PSB
<b>Prevention</b>	Information	P-C_I_PR-1: Prevent unauthorized access to information	29
	Security	P-C_I_ID-1: Manage security of identifiers and authenticators.	17
	Security	P-C_I_ID-2: Manage security of cryptographic functions.	13
	System	P-I_A-1: Limit unnecessary exposure of system functionality.	10
	Security / Communication	P-C_I_ID-3: Manage security of communication session.	9
	System	P-AY-2: Generate audit records and reports.	8
	System	P-I-1: Protect system integrity at runtime.	7
	Information	P-C_I_PR-3: Restrict access to media.	6
	User	P-ID-1: Identify and authenticate system users.	6
	System	P-ID-2: Establish authenticity of system components.	6
	Information	P-C_PR-1: Prevent information leakage.	5
	System	P-AY-1: Enable auditing of events.	4
	Communication	P-CIA_PR-1: Manage secure internal and external connections.	4
	Information	P-C_I_PR-2: Limit authorized access to information.	3
	Security	P-AY-3: Protect confidentiality and integrity of audit records.	2
System	P-ALL-1: Enable continuous monitoring.	1	
System	P-ALL-3: Deceive persistent attackers.	1	
<b>Detection</b>	System	D-I-1: Detect malicious system modification attempts.	6
	System	D-ALL-2: Analyze audit records to detect malicious activity.	3
	System	D-I-2: Detect vulnerabilities in the system.	2
	Security	D-I-3: Detect problems with security functions.	2
	System	D-ALL-1: Monitor for security incidents.	2
	User	D-ID-1: Detect malicious activity during authentication.	2
	Information	D-C_PR-1: Detect unauthorized disclosure of information.	1
<b>Response</b>	System	R-A-2: Prevent and respond to denial of service.	11
	Information	R-A-1: Setup backup and recovery procedures.	2
	Security	R-SEC-1: Respond to threats to security functions.	2
	Security	R-AY-1: Respond to failures in accountability.	1
	System	R-CIA-1: Respond to system failure conditions.	1
	User	R-ID-1: Respond to malicious activity during authentication.	1

The results indicate that our patterns have high coverage for the security requirements in the analyzed PSB. The five requirements that did not map to any of the patterns include requirements related to training of developers and testers, use of specific software testing techniques and the need to install a specific operating system. Systematically developing additional patterns based on NIST controls related to policy, management and configuration can support coverage of these types of requirements.

### **9.3 Gap Analysis**

*RQ5.2: How well do the security requirements, identified through a proprietary process, align with the security requirements identified via our semi-automated process?*

We evaluate the similarities and differences between the security requirements identified from the PRD using our process (SRD) versus the security requirements in the PSB identified through a proprietary process.

#### **9.3.1 Security Properties Identification**

We converted the PRD into a text-only format to predict the security properties implied by the sentences in the PRD using supervised machine learning. We created a domain classifier for the networks domain by randomly selecting one-third of the sentences in the PRD as explained in Section 7.3. Based on the domain classifier, we predicted the implied security properties. The recall values ranged between 53% (privacy) to 86% (availability) for different security properties (Table 35) depending on how prevalent a particular property is in the training set. Consequently, we might miss out on identifying a large percentage of sentences related to confidentiality (recall: 53%) and privacy (recall: 59%). However, we are able to predict a large percentage of sentences that imply the need for integrity (recall: 83%)

and availability (recall: 86%). We can predict a reasonable percentage of the sentences related to identification and authentication (recall: 79%) and accountability (recall: 76%). Given that many sentences talk about the same assets or functionality, the actual recall in terms of the assets and system functionality that needs to be protected will be higher depending on the level of redundancy in the input natural language artifacts.

### **9.3.2 Suggested Security Requirements Patterns**

Based on the security properties implied by sentences in the PRD, we created security goals for preventing, detecting and responding to the potential breaches for each security property. We also identified any implied goals and selected security requirements patterns to meet the identified security goals.

All of the security requirements patterns that mapped to security requirements in the PSB (Section 9.2) are identified by our process indicating that we have an overlap of 105 out of the 108 security requirements. We identified 1 requirement related to the use of a particular operating system partially. We could not identify 2 of the non-technical requirements related to personal training and management in the PSB. Moreover, we identified the following additional security requirements patterns that may be applicable, as shown in Table 55. Of the additionally suggested patterns, we did not find any equivalent requirements in the PSB. However, PSB does address point-solutions related to the generic categories spelled out by these patterns which were missed during the mapping process. For example, the pattern R-I-1 related to responding to vulnerabilities in the system is not a PSB requirement, but there are a number of items in the PSB that do suggest what to do for very specific vulnerabilities. The pattern D-I-2 related to detecting vulnerabilities in the system is considered a useful

suggestion based on the current version of the PSB. Requirements in PSB related to testing and static analysis already support parts of the D-I-2 pattern. In a soon-to-be-published update to the PSB, D-I-2 is already being incorporated.

Table 55. Additional Security Requirements Patterns based on PRD

Security Action	Scope of Pattern	Pattern Identifier and Name
Prevention	System	P-ALL-2: Limit system exposure to persistent attackers.
	Communication	P-C_PR-2: Prevent unauthorized remote sensing and activation.
	Communication	P-I_A_ID-1: Setup secure name and address resolution functions.
Detection	System	D-I-2: Detect vulnerabilities in the system.
Response	Information	R-C_PR-1: Respond to unauthorized disclosure of information.
	System	R-I-1: Respond to vulnerabilities in the system.

### 9.3.3 Limitations

The PRD document provided for analysis included a number of images depicting various network configuration. Our process could not take into account the information provided in those images and this is a general limitation of SRD process.

Although our patterns mapped to requirements in the PSB as both the requirement and the patterns address comparable security concerns, the requirements in PSB and the requirements that would be generated by the patterns are not equivalent in many cases. Specifically, the requirements in the PSB talk about particular system components, protocols or services whereas requirements templates are stated in terms of generic system components or services. Looking at the original sentences in the input, that implied the need for a particular pattern, may provide some context but not to the level of details in the PSB. Furthermore, in some cases, all the necessary detail may not be available in the input

sentence and would need an expert to assess how to instantiate the elements within the templates. The patterns still provide a useful starting point in such cases.

#### **9.4 Discussion and Lessons Learned**

Based on the feedback from analysts at Cisco, the initial applicability of any particular security requirements template to generate security requirements for the system appears to be dependent on how much the template and the target documents match in language to begin with. For example, applying templates based on natural language constructs associated with information system domain may not be immediately applicable to documents associated with technical requirements for data networking. For instance, healthcare documents are often written -as a matter of course- with natural language phrases associated with Confidentiality, Integrity, Availability, Identification and Authentication, Accountability, and Privacy, as such constructs are required due to HIPPA and other regulations. Technical networking documents, on the other hand, are not yet always written with security in mind from the get-go. Hence, it appears that to achieve immediate applicability of a set of security templates, either those templates ought to be customized for the target subject field or - more likely - the subject documents ought to be written to include more security language constructs (or implied constructs) or both. The SRD process shows promise for requirements processing and is worth additional experimentation for additional domains. Organizations may also customize the SRD process based on their internal security categorization, if applicable.

## 10 CONCLUSION

We have developed a semi-automated process, Security Requirements Discoverer (SRD), for identifying candidate security requirements for a system based on natural language requirements artifacts. We provide a visual map of various research activities involved in the process in Figure 29. We summarize our findings and contributions in the following sections.

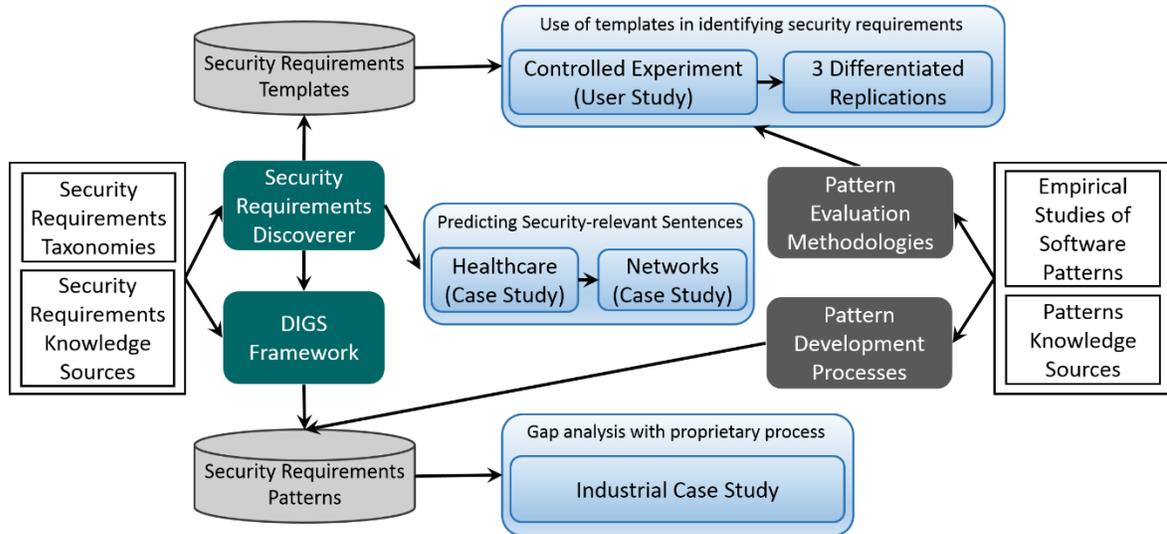


Figure 29. An Overview of Research Contributions

### 10.1 Summary

Our process takes as input existing natural language requirements artifacts and produces a set of candidate security requirements for the system as output. From a user's perspective, an analyst can import the artifacts they want to analyze into our process. Our process will automatically identify security properties implied by sentences in the input and suggest applicable security requirements patterns. The analyst will then be able to select and instantiate appropriate patterns to specify security requirements for the system.

To enable the automatic identification of security properties, we manually developed a labeled corpus of requirements sentences and the security properties implied by those sentences (such as confidentiality, integrity, availability, accountability, and privacy) using a classification guide. Once the domain corpus is available, it can be used to train machine learning classifiers and predict security properties across other documents in the domain. The labeled corpus can be developed incrementally over time as additional labeled requirement sentences are added into the system. We evaluated the performance of the machine learning classifiers in identifying security properties through a case study of six documents from the electronic healthcare domain. The domain corpus contains ~11,000 sentences, of which 13% explicitly state security requirements and an additional 33% sentences indicate the need for implied security requirements. Our classification approach identified security properties with a precision of .82 and recall of .79. We evaluated the generalizability of our process by analyzing over 350 randomly selected requirements sentences from a networking product requirements specification document. We predicted the security properties with a maximum precision of .77 and maximum recall of .75.

To support the specification of security requirements based on the implied security properties, we have empirically developed a set of 35 security requirements patterns based upon an analysis of NIST security and privacy controls. Each pattern is cataloged in terms of one or more security properties and supports the goals for preventing, detecting or responding to a breach of the corresponding security properties. The solution part of the pattern provides a group of related security requirements templates, with a combined set of

131 different templates in the 35 patterns. The templates can be instantiated to specify security requirements to support the goals related to each security property.

We conducted multiple controlled experiments involving 205 graduate students to evaluate the use of automatically-suggested security requirements templates in identifying and specifying security requirements. Participants in the treatment group, using the templates, identified almost twice as many relevant security requirements (46% vs. 25%) in the same amount of time as compared to the control group. The requirements identified by the treatment group were also more relevant (89% vs. 62%) and of better quality (3.2 vs. 2.3 on a scale of 1-5). Our results indicate that our process can support an analyst in considering multiple security properties and identifying a larger set of security requirements when compared to a control group.

We conducted an evaluation of our process in identifying security requirements as compared to a propriety process for a product from the networks domain. Using our process, we identified security requirements comparable to 105 of the 108 security requirements identified using the proprietary process. We missed 2 non-technical security requirements related to personal training, and partially identified requirements related to the use of a specific operating system. We also proposed 6 additional security requirements patterns for detecting and responding to vulnerabilities and failure conditions, and limiting system's exposure to persistent attackers.

We contribute an empirically validated systematic process for identifying a candidate set of implied security requirements from natural language requirements artifacts. To support automatic identification of security properties, we create domain classifiers for healthcare

and networks domains. To support specification of security requirements related to the identified properties, we develop a framework for systematically discovering security goals and empirically developing security requirements patterns to meet the goals. Our research can help mitigate errors of omissions in security requirements by identifying the implied security requirements early on that may otherwise be overlooked.

## 10.2 Contributions

We contribute the following with this thesis:

- An empirically evaluated framework, DIGS, for systematically discovering security goals for a software system.
- A systematic process for identifying security requirements patterns from existing security requirements knowledge sources.
- A catalog of 35 security requirements patterns based on NIST Special Publication 800-53<sup>34</sup> to generate security requirements operationalizing security goals.
- A semi-automated process, Security Requirements Discoverer (SRD), for identifying and classifying security properties implied by sentences in natural language requirements artifacts to identify a set of candidate implied security requirements for the system.
- Empirical evaluation of SRD in classifying sentences in natural language requirements artifacts in terms of the implied security properties through multiple case studies of documents from different domains.

---

<sup>34</sup><http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf>

- Empirical evaluation of SRD related to the use of security requirements templates for identifying and specifying security requirements implied by natural language requirements artifacts through multiple controlled experiments.
- An empirical evaluation of SRD through an industrial case study.
- A systematic map of empirical literature that draws together research evaluating software pattern application in problem solving.
  - A classification of measures used to evaluate success when applying software patterns and approaches to minimize subjectivity during evaluation.
  - Compilation of reported threats to validity of studies evaluating software pattern application.
  - Considerations to support replication and comparability of study findings.

### **10.3 Future Directions**

We have presented a semi-automated process for identifying security requirements from natural language requirements artifacts. In future, further automation can be introduced in various steps of the process. For instance, we can leverage natural language processing to automatically extract assets and actions from the input artifacts. Combining this information with the already identified security properties of the assets can lead to automatic identification of security goals and can also support filling in relevant parts of the suggested templates for requirements specification.

Natural language requirements often have redundant information. For instance, multiple sentences may talk about the same assets and actions on the assets. We have not analyzed

how the precision and recall would be affected if we factor in redundant information in the natural language artifacts however it merits further exploration.

We create a set of security goals for the system as an intermediate step of the overall SRD process. It might be useful to create a mapping between the security goals and later stages of software development, such as creating test cases to assess how well the requirements identified by our process have been integrated into the system. The use of SRD or similar process for identifying and specifying other types of requirements may be another research area.

## REFERENCES

- A. Toval, J. Nicolar, B. Moros, and F Garcia. 2002. "Requirements Reuse for Improving Information Systems Security: A Practitioner's Approach." *Requirements Engineering* 6 (4): 205–19. doi:10.1007/PL00010360.
- A.I. Antón, J.B. Earp, and R A Carter. 2003. "Precluding Incongruous Behavior by Aligning Software Requirements with Security and Privacy Policies." *Information and Software Technology, Elsevier* 45 (14): 967–77.
- Adolph, S, W Hall, and P Kruchten. 2008. "A Methodological Leg to Stand on: Lessons Learned Using Grounded Theory to Study Software Development." In *Conference of the Center for Advanced Studies on Collaborative Research (CASCON)*, 166–78. New York, NY, USA.
- Alexander, Ian. 2003. "Misuse Cases: Use Cases with Hostile Intent." *IEEE Software* 20 (1): 58–66.
- Ali, M., and M.O. Elish. 2013. "A Comparative Literature Survey of Design Patterns Impact on Software Quality." In *International Conference on Information Science and Applications (ICISA)*, 1–7. Suwon.
- Ali, Raian, Fabiano Dalpiaz, and Paolo Giorgini. 2009. "A Goal Modeling Framework for Self-Contextualizable Software." In *Lecture Notes in Business Information Processing*, 29:326–38. doi:10.1007/978-3-642-01862-6\_27.
- Alshazly, Amira A., Ahmed M. Elfatraty, and Mohamed S. Abougabal. 2014. "Detecting Defects in Software Requirements Specification." *Alexandria Engineering Journal* 53 (3). Faculty of Engineering, Alexandria University: 513–27. doi:10.1016/j.aej.2014.06.001.
- Ampatzoglou, A., S. Charalampidou, and I. Stamelos. 2013. "Research State of the Art on GoF Design Patterns: A Mapping Study." *Journal of Systems and Software* 86 (7): 1945–64.
- Anton, A, and C Potts. 1998. "The Use of Goals to Surface Requirements for Evolving Systems." In *Proceedings of the 20th International Conference on Software Engineering*, 157–66. Washington, DC: IEEE.
- Asnar, Yudistira, Paolo Giorgini, Fabio Massacci, and Nicola Zannone. 2007. "From Trust to Dependability through Risk Analysis." In *Proceedings - Second International Conference on Availability, Reliability and Security, ARES 2007*, 19–26. Vienna. doi:10.1109/ARES.2007.93.
- B. Arkin, S. Stender, and G McGraw. 2005. "Software Penetration Testing." *IEEE Security*

& Privacy.

- B. Ramesh, and M Jarke. 2001. "Toward Reference Models for Requirements Traceability." *IEEE Transactions on Software Engineering* 27 (1): 58–93.
- Baharudin, Baharum, Lam Hong Lee, and Khairullah Khan. 2010. "A Review of Machine Learning Algorithms for Text-Documents Classification." *Journal of Advances in Information Technology* 1 (1): 4–20. doi:10.4304/jait.1.1.4-20.
- Basili, V. 1992. "Software Modeling and Measurement: The Goal/Question/Metric Paradigm." College Park, MD.
- Basin, David, and J Doser. 2006. "Model Driven Security: From UML Models to Access Control Infrastructures." *ACM Transactions on Software* 15 (1): 39–91. doi:10.1145/1125808.1125810.
- Berczuk, S P. 2003. *Software Configuration Management Patterns: Effective Teamwork, Practical Integration*. Addison-Wesley Professional.
- Bloom, Benjamin S, and David R Krathwohl. 1956. *Taxonomy of Educational Objectives: The Classification of Educational Goals. Handbook I, Cognitive Domain*. Longmans, Green.
- Boehm, B, and R Turner. 2003. *Balancing Agility and Discipline: A Guide for the Perplexed*. Addison-Wesley.
- Breaux, T D, A I Antón, K Boucher, and M Dorfman. 2008. "Legal Requirements, Compliance and Practice: An Industry Case Study in Accessibility." In *IEEE 16th International Requirements Engineering Conference*, 10. Catalunya, Spain.
- Breaux, Travis D, Hanan Hibshi, and Ashwini Rao. 2014. "Eddy, A Formal Language for Specifying and Analyzing Data Flow Specifications for Conflicting Privacy Requirements." *Requirements Engineering Journal* 19 (3): 281–307.
- Budgen, D, M Turner, P Brereton, and B Kitchenham. 2008. "Using Mapping Studies in Software Engineering." In *20th Annual Psychology of Programming Interest Group Conference (PPIG)*, 195–204. Lancaster University, UK.
- Carver, J. 2010. "Towards Reporting Guidelines for Experimental Replications: A Proposal." In *1st International Workshop on Replication in Empirical Software Engineering Research (RESER) [Held during ICSE 2010]*, 4. Cape Town, South Africa.
- Carver, J.C., L. Jaccheri, and S. Morasca. 2010. "A Checklist for Integrating Student Empirical Studies with Research and Teaching Goals." *Empirical Software Engineering* 15 (1): 35–59.

- Charness, Neil, and Michael Tuffiash. 2008. "The Role of Expertise Research and Human Factors in Capturing, Explaining, and Producing Superior Performance." *Human Factors: The Journal of the Human Factors and Ergonomics Society* 50 (3): 427–32.
- Chung, Lawrence, Barbara Paech, Liping Zhao, Lin Liu, and Sam Supakkul. 2012. "RePa Requirements Pattern Template." In *International Workshop on Requirements Patterns (RePa'12)*, 7–10. Chicago. [http://www.utdallas.edu/~supakkul/rep12/RePaRequirementsPatternTemplate v1.0.1.pdf](http://www.utdallas.edu/~supakkul/rep12/RePaRequirementsPatternTemplatev1.0.1.pdf).
- Cohen, J. 1988. *Statistical Power Analysis for the Behavioral Sciences*. Hillsdale, N.J.: Lawrence Erlbaum Associates.
- "Common Criteria for Information Technology Security Evaluation, Version 3.1. Release 4." 2012. <http://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R4.pdf>.
- Cooper, G, and J Sweller. 1987. "Effects of Schema Acquisition and Rule Automation on Mathematical Problem-Solving Transfer." *Journal of Educational Psychology* 79 (4): 347–62.
- Creswell, J W. 2003. *Research Design: Qualitative, Quantitative and Mixed Methods Approaches*. 2nd ed. Sage Publications.
- D. C. Schmidt, M. Fayad, and R E Johnson. 1996. "Software Patterns." *Communications of the ACM*.
- D. Mellado, C. Blanco, L. E. Sánchez, and E Fernández-Medina. 2010. "A Systematic Review of Security Requirements Engineering." *Computer Standards and Interfaces* 32 (4): 153–65. doi:10.1016/j.csi.2010.01.006.
- Daramola, Olawande, Guttorm Sindre, and Tor Stalhane. 2012. "Pattern-Based Security Requirements Specification Using Ontologies and Boilerplates." In *2012 2nd IEEE International Workshop on Requirements Patterns, RePa 2012 - Proceedings*, 54–59. Chicago. doi:10.1109/RePa.2012.6359973.
- De Gramatica, M., K. Labunets, F. Massacci, F. Paci, and A. Tedeschi. 2015. "The Role of Catalogues of Threats and Security Controls in Security Risk Assessment: An Empirical Study with ATM Professionals." In *21st International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ2015)*, 98–114. Essen, Germany: Springer Verlag.
- Dieste, O., A. Grimán, and N. Juristo. 2009. "Developing Search Strategies for Detecting Relevant Experiments." *Empirical Software Engineering* 14 (5): 513–39.
- Dong, J, Y Zhao, and T Peng. 2009. "A Review of Design Pattern Mining Techniques." *International Journal of Software Engineering and Knowledge Engineering (IJSEKE)*

19 (6): 823–55.

- Fabian, Benjamin, Seda Gürses, Maritta Heisel, Thomas Santen, and Holger Schmidt. 2010. “A Comparison of Security Requirements Engineering Methods.” *Requirements Engineering - Special Issue on RE'09: Security Requirements Engineering* 15 (1): 7–40.
- “Federal Information Security Management Act.” 2002. <http://csrc.nist.gov/drivers/documents/FISMA-final.pdf>.
- Firesmith, D G. 2003. “Engineering Security Requirements.” *J. Object Technology* 2 (1): 53–68.
- Firesmith, D G. 2004. “Specifying Reusable Security Requirements.” *Journal of Object Technology* 3 (1): 61–75.
- Fowler, M. 1997. *Analysis Patterns: Reusable Object Models*. Edited by Grady; Jacobson Booch Ivar; Rumbaugh, James. *Object Technology Series*. Menlo Park, CA: Addison Wesley Longman, Inc.
- Fowler, M. 2002. *Patterns of Enterprise Application Architecture*. Reading, MA: Addison-Wesley Professional.
- Franqueira, V. N. L., T. T. Tun, Y. Yu, R. Wieringa, and B. Nuseibeh. 2011. “Risk and Argument: A Risk-Based Argumentation Method for Practical Security.” In *IEEE International Requirements Engineering Conference*, 239–48. Trento.
- Galster, M., and P. Avgeriou. 2012. “Qualitative Analysis of the Impact of SOA Patterns on Quality Attributes.” In *12th International Conference on Quality Software*, 167–70. Xi’an, Shaanxi.
- Gamma, E, R Helm, R Johnson, and J Vlissides. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, Massachusetts: Addison-Wesley Publishing Company.
- Gao, X, and M P Singh. 2014. “Extracting Normative Relationships from Business Contracts.” In *International Conference on Autonomous Agents and Multi-Agent Systems*, 101–8. Paris.
- Gervasi, Vincenzo, and Didar Zowghi. 2005. “Reasoning About Inconsistencies in Natural Language Requirements.” *Transactions on Software Engineering and Methodology* 14 (3): 277–330.
- Giactalone, Matteo, Federica Paci, Rocco Mammoliti, Rodolfo Perugino, Fabio Massacci, and Claudio Selli. 2014. “Security Triage: An Industrial Case Study on the Effectiveness of a Lean Methodology to Identify Security Requirements.” In *Proceedings of the 8th*

*ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 1–8. Torino. doi:10.1145/2652524.2652585.

Giorgini, P., H. Mouratidis, and N. Zannone. 2006. “Modelling Security and Trust with Secure Tropos.” In *Integrating Security and Software Engineering: Advances and Future Visions*. Idea Group Publishing.

Gómez, Omar S, Natalia Juristo, and Sira Vegas. 2010. “Replication, Reproduction and Re-Analysis: Three Ways for Verifying Experimental Findings.” In *1st International Workshop on Replication in Empirical Software Engineering Research, RESER*, 4. Cape Town, South Africa: ACM.

Gray, P. H., and D.B. Meister. 2004. “Knowledge Sourcing Effectiveness.” *Management Science* 50 (6): 821–34.

Greenspan, S., J. Mylopoulos, and Alex Borgida. 1982. “Capturing More World Knowledge in the Requirements Specification.” In *6th International Conference on Software Engineering*, 225–34. Los Alamitos, CA.

Haley, Charles B, Robin Laney, Jonathan D Moffett, and Bashar Nuseibeh. 2008. “Security Requirements Engineering: A Framework for Representation and Analysis.” *IEEE Transactions on Software Engineering* 34 (1): 133–53.

Hall, M., H. National, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. 2009. “The WEKA Data Mining Software: An Update.” *ACM Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD Explorations)* 11 (1): 10–18.

Han, J., M. Kamber, and J. Pei. 2011. *Data Mining: Concepts and Techniques*. 3rd ed. The Morgan Kaufmann Series in Data Management Systems.

Harville, D. A. 1977. “Maximum Likelihood Approaches to Variance Component Estimation and to Related Problems.” *Journal of the American Statistical Association* 72: 320–40.

Heckman, S., and L. Williams. 2008. “On Establishing a Benchmark for Evaluating Static Analysis Alert Prioritization and Classification Techniques.” In *2nd International Conference on Empirical Software Engineering and Measurement (ESEM 2008)*, 41–50. Kaiserslautern, Germany.

Hibshi, Hanan, Travis Breaux, Maria Riaz, and Laurie Williams. 2014. “Towards a Framework to Measure Security Expertise in Requirements Analysis.” In *1st International Workshop on Evolving Security and Privacy Requirements Engineering (ESPRES)*, 13–18. Sweden.

Höst, Martin, and Björn Regnell. 2000. “Using Students as Subjects—A Comparative Study of Students and Professionals in Lead-Time Impact Assessment.” *Empirical Software*

*Engineering* 5 (3): 201–14.

- I.A. Tondel, M.G. Jaatun, and P H Meland. 2008. “Security Requirements for the Rest of Us: A Survey.” *IEEE Software*.
- “IEEE Standard Glossary of Software Engineering Terminology.” 1990. <http://standards.ieee.org/findstds/standard/610.12-1990.html>.
- Ito, Yurina, Hironori Washizaki, Masatoshi Yoshizawa, Yoshiaki Fukazawa, Takao Okubo, Haruhiko Kaiya, Atsuo Hazeyama, Nobukazu Yoshioka, and Eduardo B. Fernandez. 2015. “Systematic Mapping of Security Patterns Research.” In *Pattern Languages of Programs (PLoP)*. Pittsburgh.
- Jackson, M. 2005. “Problem Frames and Software Engineering.” *Information and Software Technology* 47 (14): 903–12.
- Jalil, M.A., and S.A.M. Noah. 2007. “The Difficulties of Using Design Patterns among Novices: An Exploratory Study.” In *5th International Conference on Computational Science & Applications (ICCSA’07)*, 97–103. Kuala Lumpur, Malaysia.
- Jedlitschka, Andreas, Marcus Ciolkowski, and Dietmar Pfahl. 2008. “Reporting Experiments in Software Engineering.” In *Guide to Advanced Empirical Software Engineering*, 201–28. Springer London.
- Joachims, Thorsten. 1998. “Text Categorization with Support Vector Machines: Learning with Many Relevant Features.” In *European Conference on Machine Learning (ECML)*, 137–42. Chemnitz, Germany.
- Karpati, Peter, Andreas L. Opdahl, and Guttorm Sindre. 2015. “Investigating Security Threats in Architectural Context: Experimental Evaluations of Misuse Case Maps.” *Journal of Systems and Software* 104. Elsevier Ltd.: 90–111. doi:10.1016/j.jss.2015.02.040.
- Kerth, Norman L., and Ward Cunningham. 1997. “Using Patterns to Improve Our Architectural Vision.” *IEEE Software* 14 (1): 53–59.
- Khomh, F., and Y.-G. Guéhéneuc. 2008. “Do Design Patterns Impact Software Quality Positively?” In *12th European Conference on Software Maintenance and Reengineering*, 274–78. Athens.
- Kitchenham, B A, D Budgen, and O Pearl Brereton. 2011. “Using Mapping Studies as the Basis for Further Research—A Participant-Observer Case Study.” *Information & Software Technology, Special Section from EASE* 53 (4): 638–51.
- Kitchenham, B, and S Charters. 2007. “Guidelines for Performing Systematic Literature

Reviews in Software Engineering.” Technical Report EBSE-2007-01 School of Computer Science and Mathematics, Keele University.

Kitchenham, BA A., Tore Dybå, and M. Jørgensen. 2004. “Evidence-Based Software Engineering.” In *26th International Conference on Software Engineering (ICSE)*, 273–81. Scotland. doi:10.1109/ICSE.2004.1317449.

Kitchenham, Barbara, Hiyam Al-Khilidar, Muhammed Ali Babar, Mike Berry, Karl Cox, Jacky Keung, Felicia Kurniawati, Mark Staples, He Zhang, and Liming Zhu. 2008. “Evaluating Guidelines for Reporting Empirical Software Engineering Studies.” *Empirical Software Engineering* 13 (1): 97–121. doi:10.1007/s10664-007-9053-5.

Kitchenham, Barbara, Dag I K Sjøberg, Tore Dybå, Pearl Brereton, David Budgen, Martin Höst, and Per Runeson. 2012. “Trends in the Quality of Human-Centric Software Engineering Experiments – A Quasi-Experiment.” *IEEE Transactions on Software Engineering* 39 (7): 1002–17.

Kleinschmager, S., and S. Hanenberg. 2011. “How to Rate Programming Skills in Programming Experiments?: A Preliminary, Exploratory, Study Based on University Marks, Pretests, and Self-Estimation.” In *3rd ACM SIGPLAN Workshop on Evaluation and Usability of Programming Languages and Tools (PLATEAU '11)*, 15–24. Portland, OR.

Koch, G G. 1977. “The Measurement of Observer Agreement for Categorical Data.” *Biometrics* 33 (1): 159–74.

Kohls, C., and K. Scheiter. 2008. “The Relation between Design Patterns and Schema Theory.” In *15th Conference on Pattern Languages of Programs (PLoP)*, 1–14. Nashville, Tennessee.

Kolfschoten, G, S Lukosch, A Verbraeck, and Edwin Valentin. 2010. “Cognitive Learning Efficiency through the Use of Design Patterns in Teaching.” *Computers and Education* 54 (3): 652–60.

Kurt Schneider, Eric Knauss, Siv Houmb, and Shareeful Islam. 2012. “Enhancing Security Requirements Engineering by Organizational Learning.” *Requirements Engineering* 17 (1): 35–56.

Lamsweerde, A.v. 2004. “Elaborating Security Requirements by Construction of Intentional Anti-Models.” In *26th International Conference on Software Engineering (ICSE)*, 148–57. Edinburgh, Scotland.

Lane, David M. 2011. “Research Design.” *Online Statistics Education: An Interactive Multimedia Course of Study*. Rice University. [http://onlinestatbook.com/2/research\\_design/designs.html](http://onlinestatbook.com/2/research_design/designs.html).

- LaPiedra, James. 2002. "The Information Security Process: Prevention, Detection and Response." *Style (DeKalb, IL)*.
- Levenshtein, V. I. 1966. "Binary Codes Capable of Correcting Deletions, Insertions, and Reversals." *Soviet Physics Doklady* 10: 707–10.
- Lindsay, R.M., and A.S.C. Ehrenberg. 1993. "The Design of Replicated Studies." *The American Statistician* 47 (3): 217–28.
- Massacci, Fabio, John Mylopoulos, and Nicola Zannone. 2010. "Security Requirements Engineering: The SI\* Modeling Language and the Secure Tropos Methodology." *Advances in Intelligent Information Systems* 265: 147–74. doi:10.1007/978-3-642-05183-8\_6.
- Massacci, Fabio, and Nicola Zannone. 2006. "Detecting Conflicts between Functional and Security Requirements with Secure Tropos: John Rusnak and the Allied Irish Bank." In *Social Modeling for Requirements Engineering*, 337–62.
- Massey, Aaron K., Jacob Eisenstein, Annie I. Anton, and Peter P. Swire. 2013. "Automated Text Mining for Requirements Analysis of Policy Documents." In *2013 21st IEEE International Requirements Engineering Conference, RE 2013 - Proceedings*, 4–13. Rio de Janeiro. doi:10.1109/RE.2013.6636700.
- McCrum-Gardner, Evie. 2008. "Which Is the Correct Statistical Test to Use?" *British Journal of Oral and Maxillofacial Surgery* 46 (1): 38–41. doi:10.1016/j.bjoms.2007.09.002.
- McDermott, J., and C. Fox. 1999. "Using Abuse Case Models for Security Requirements Analysis." In *Computer Security Applications Conference*, 55–64. Phoenix, AZ.
- Mead, N R, E D Houg, and T R Stehney. 2005. "Security Quality Requirements Engineering (SQUARE) Methodology." Carnegie Mellon University: Software Engineering Institute.
- Mellado, Daniel, Eduardo Fernández-Medina, and Mario Piattini. 2007. "A Common Criteria Based Security Requirements Engineering Process for the Development of Secure Information Systems." *Computer Standards & Interfaces* 29 (2): 244–53. doi:10.1016/j.csi.2006.04.002.
- Menzies, T., A. Dekhtyar, J. Distefano, and J. Greenwald. 2007. "Problems with Precision: A Response to 'Comments on "Data Mining Static Code Attributes to Learn Defect Predictors"'" *IEEE Transactions on Software Engineering* 33 (9): 637–40.
- Meszaros, G, and J Doble. 1996. "Metapatterns: A Pattern Language for Pattern Writing." In *The 3rd Pattern Languages of Programming Conference*, 4–6. Monticello, Illinois,

USA.

- “Minimum Security Requirements for Federal Information and Information Systems .” 2006. Federal Information Processing Standards. <http://csrc.nist.gov/publications/fips/fips200/FIPS-200-final-march.pdf>.
- Montgomery, Dough. 2012. “Nested and Split-Plot Designs.” In *Design and Analysis of Experiments*, 8th ed., 604–41.
- N. Yoshioka, H. Washizaki, and K Maruyama. 2008. “A Survey on Security Patterns.” *Progress in Informatics, Special Issue: The Future of Software Engineering for Security and Privacy*, no. 5: 35–47.
- Nuseibeh, Bashar, and Steve Easterbrook. 2000. “Requirements Engineering: A Roadmap.” In *Proceedings of the Conference on The Future of Software Engineering - ICSE '00*, 1:35–46. Limerick, Ireland. doi:10.1145/336512.336523.
- Pandita, Rahul, Xusheng Xiao, Hao Zhong, Tao Xie, and Stephen Oney. 2012. “Inferring Method Specifications from Natural Language API Descriptions.” In *34th International Conference on Software Engineering (ICSE 2012)*, 815–25. Zurich, Switzerland.
- Petersen, K., R. Feldt, S. Mujtaba, and M. Mattsson. 2008. “Systematic Mapping Studies in Software Engineering.” In *12th International Conference on Evaluation and Assessment in Software Engineering (EASE'08)*, 68–77. University of Bari, Italy.
- Quinlan, J. R. 1986. “Induction of Decision Trees.” *Machine Learning* 1: 81–106.
- Riaz, Maria, Travis Breaux, and Laurie Williams. 2015. “How Have We Evaluated Software Pattern Application? A Systematic Mapping Study of Research Design Practices.” *Information and Software Technology* 65: 14–38. doi:10.1016/j.infsof.2015.04.002.
- Riaz, Maria, Jason King, John Slankas, and Laurie Williams. 2014. “Hidden in Plain Sight: Automatically Identifying Security Requirements from Natural Language Artifacts.” In *22nd International Requirements Engineering Conference (RE)*, 183–92. Karlskrona: IEEE. doi:10.1109/RE.2014.6912260.
- Riaz, Maria, John Slankas, Jason King, and Laurie Williams. 2014. “Using Templates to Elicit Implied Security Requirements from Functional Requirements - A Controlled Experiment.” In *8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 1–10. Torino: ACM Press. <http://dl.acm.org/citation.cfm?doid=2652524.2652532>.
- Riaz, Maria, Jonathan Stallings, Munindar P. Singh, John Slankas, and Laurie Williams. 2016. “DIGS – A Framework for Discovering Goals for Security Requirements Engineering.” In *Empirical Software Engineering and Measurement (ESEM) (to*

*Appear*). Ciudad Real.

- Riaz, Maria, and Laurie Williams. 2012. "Security Requirements Patterns: Understanding the Science behind the Art of Pattern Writing." In *Proceedings of the 2nd IEEE International Workshop on Requirements Patterns (RePa)*, 29–34. Chicago. doi:10.1109/RePa.2012.6359977.
- Robillard, Pierre N. 1999. "The Role of Knowledge in Software Development." *Communications of the ACM*.
- Salini, P, and S Kanmani. 2012. "Survey and Analysis on Security Requirements Engineering." *Computers and Electrical Engineering* 38 (6): 1785–97.
- Saltzer, J H, and M D Schroeder. 1974. "The Protection of Information in Computer Systems." *Communication of the ACM* 17 (7).
- Schmidt, Douglas C. 2007. *A Pattern Language for Distributed Computing. Pattern-Oriented Software Architecture*. Vol. 4. Wiley & Sons.
- Schumacher, M, E Fernandez-Buglioni, D Hyberston, F Buschmann, and P Sommerlad. 2006. *Security Patterns: Integrating Security and Systems Engineering*. West Sussex: John Wiley & Sons, Ltd.
- Shadish, W, T Cook, and D Campbell. 2002. *Experimental and Quasi Experimental Designs for Generalized Causal Inference*. Boston: Houghton Mifflin.
- Silva, Fabio Q B da, André L M Santos, Sérgio C B Soares, A César C França, and Cleviton V F Monteiro. 2010. "A Critical Appraisal of Systematic Reviews in Software Engineering from the Perspective of the Research Questions Asked in the Reviews." In *Empirical Software Engineering and Measurement (ESEM)*. Bolzano-Bozen, Italy.
- Sim, S.E., S. Easterbrook, and R.C. Holt. 2003. "Using Benchmarking to Advance Research: A Challenge to Software Engineering." In *25th International Conference on Software Engineering (ICSE)*, 74–83. Portland, OR.
- Sindre, G, and A L Opdahl. 2005. "Eliciting Security Requirements with Misuse Cases." *Requirements Engineering* 10 (1): 34–44. doi:10.1007/s00766-004-0194-4.
- Sjøberg, Dag I. K., Tore Dyba, and Magne Jørgensen. 2007. "The Future of Empirical Methods in Software Engineering Research." In *Future of Software Engineering (FOSE'07)*, 358–78. Minneapolis, MN.
- Slankas, J, X Xiao, L Williams, and T Xie. 2014. "Relation Extraction for Inferring Access Control Rules from Natural Language Artifacts." In *Annual Computer Security Applications Conference (ACSAC 2014)*, 366–75. New Orleans, LA.

- Slankas, J., and L. Williams. 2013. "Automated Extraction of Non-Functional Requirements in Available Documentation." In *International Conference on Software Engineering (ICSE) 1st International Workshop on Natural Language Analysis in Software Engineering (NaturaLiSE)*, 9–16. San Francisco, CA.
- Slavin, Rocky, Jean Michel Lehker, Jianwei Niu, and Travis D. Breaux. 2014. "Managing Security Requirements Patterns Using Feature Diagram Hierarchies." In *Proceedings of the 22nd IEEE International Requirements Engineering Conference*, 193–202. Karlskrona. doi:10.1109/RE.2014.6912261.
- Smith, B, and L Williams. 2012. "On the Effective Use of Security Test Patterns." In *Sixth International Conference on Software Security and Reliability (SERE2012)*, 108–17. Washington, DC.
- Souag, Amina, Raúl Mazo, Camille Salinesi, and Isabelle Comyn-Wattiau. 2015. "Reusable Knowledge in Security Requirements Engineering: A Systematic Mapping Study." *Requirements Engineering* 21 (2): 251–83. doi:10.1007/s00766-015-0220-8.
- "Special Publication 800-53 Revision 4 - Security and Privacy Controls for Federal Information Systems and Organizations." 2013. National Institute of Standards and Technology. <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf>.
- "Standards for Security Categorization of Federal Information and Information Systems." 2004. Federal Information Processing Standards. <http://csrc.nist.gov/publications/fips/fips199/FIPS-PUB-199-final.pdf>.
- Suleiman, Husam, and Davor Svetinovic. 2013. "Evaluating the Effectiveness of the Security Quality Requirements Engineering (SQUARE) Method: A Case Study Using Smart Grid Advanced Metering Infrastructure." *Requirements Engineering* 18 (3): 251–79. doi:10.1007/s00766-012-0153-4.
- Sweller, J. 1988. "Cognitive Load during Problem Solving: Effects on Learning." *Cognitive Science* 12: 257–85.
- Taubenberger, Stefan, Jan Jürjens, Yijun Yu, and Bashar Nuseibeh. 2011. "Problem Analysis of IT-Security Risk Assessment Methods – An Experience Report from the Insurance and Auditing Domain." In *Future Challenges in Security and Privacy for Academia and Industry*, 259–70.
- Taubenberger, Stefan, Jan Jürjens, Yijun Yu, and Bashar Nuseibeh. 2013. "Resolving Vulnerability Identification Errors Using Security Requirements on Business Process Models." *Information Management & Computer Security* 21 (3): 202–23.
- "Underlying Technical Models for Information Technology Security." 2001. National

Institute of Standards and Technology. <http://csrc.nist.gov/publications/nistpubs/800-33/sp800-33.pdf>.

Van Lamsweerde, A. 2001. "Building Formal Requirements Models for Reliable Software." In *Methods*, 1–20. doi:10.1007/3-540-45136-6\_1.

Viera, Anthony J. 2005. "Understanding Interobserver Agreement: The Kappa Statistic." *Family Medicine* 37 (5): 360–63.

Vlissides, John. 1998. *Pattern Hatching: Design Patterns Applied*. 1st ed.

Vocak, M, W F Tichy, D I K Sjøberg, E Arisölm, and M Aldrin. 2004. "A Controlled Experiment Comparing the Maintainability of Programs Designed with and without Design Patterns—a Replication in a Real Programming Environment." *Empirical Software Engineering* 9 (3): 149–95.

Walia, Gursimran Singh, and Jeffrey C. Carver. 2009. "A Systematic Literature Review to Identify and Classify Software Requirement Errors." *Information and Software Technology* 51 (7). Elsevier B.V.: 1087–1109. doi:10.1016/j.infsof.2009.01.004.

Wen, Y, H Zhao, and L Liu. 2011. "Analysing Security Requirements Patterns Based on Problems Decomposition and Composition." In *First International Workshop on Requirements Patterns (RePa)*, 11–20. Trento.

Williams, Laurie, Andrew Meneely, and Grant Shipley. 2010. "Protection Poker : The New Software Security 'Game.'" *IEEE Security and Privacy*. doi:10.1109/MSP.2010.58.

Withall, S. 2007. *Software Requirement Patterns*. O'Reilly.

Wohlin, Claes, Per Runeson, Martin Host, Magnus C Ohlsson, Bjorn Regnell, and Anders Wesslen. 2000. *Experimentation in Software Engineering: An Introduction*. Edited by Victor R Basili. Sweden: Kluwer Academic Publishers.

Yang, J, and L Liu. 2008. "Modelling Requirements Patterns with a Goal and PF Integrated Analysis Approach." In *32nd Annual IEEE International Computer Software and Applications Conference (COMPSAC 2008)*, 239–46. Turku.

Yang, Y., and J. P. Pedersen. 1997. "A Comparative Study on Feature Selection in Text Categorization." In *Fourteenth International Conference on Machine Learning (ICML)*, 412–20. Nashville, TN.

Yin, R K. 2009. *Case Study Research: Design and Methods*. 4th ed. Sage Publications.

Yskout, K, R Scandariato, and W Joosen. 2012. "Does Organizing Security Patterns Focus Architectural Choices?" In *International Conference on Software Engineering (ICSE)*

'12), 617–27. Zurich, Switzerland.

Yue, K. 1987. “What Does It Mean to Say That a Specification Is Complete?” In *Fourth International Workshop on Software Specification and Design (IWSSD-4)*. Monterey, USA.

Zhang, C, and D Budgen. 2012. “What Do We Know about the Effectiveness of Software Design Patterns?” *IEEE Transactions on Software Engineering* 38 (5): 1213–31.

Zhang, He, and Muhammad Ali Babar. 2010. “On Searching Relevant Studies in Software Engineering.” In *14th International Conference on Evaluation and Assessment in Software Engineering*, 111–20. Keele University, UK.

## APPENDICES

## APPENDIX A: List of Selected Studies on Software Patterns

Reference Code	Study Bibliographical Reference
<b>S01</b> (A, B, C)	G. Kolfshoten, S. Lukosch, A. Verbraeck, and E. Valentin, "Cognitive learning efficiency through the use of design patterns in teaching," <i>Computers and Education</i> , vol. 54, pp. 652-660, April 2010.
<b>S02</b>	A. Chatzigeorgiou, N. Tsantalis, and I. Deligiannis, "An empirical study on students' ability to comprehend design patterns," <i>Computers and Education</i> , vol. 51, pp. 1007-1016, 2008.
<b>S03</b> (A, B)	L. Prechelt, B. Unger-Lamprecht, M. Philippsen, and W. F. Tichy, "Two controlled experiments assessing the usefulness of design pattern documentation in program maintenance," <i>IEEE Transactions on Software Engineering</i> , vol. 28, 2002.
<b>S03-R1</b>	M. Torchiano, "Documenting pattern use in java programs," in <i>International Conference on Software Maintenance (ICSM'02)</i> , Montreal, Canada, 2002, pp. 230-233.
<b>S04</b>	L. Prechelt, B. Unger, W. F. Tichy, P. Brossler, and L. G. Votta, "A controlled experiment in maintenance comparing design patterns to simpler solutions," <i>IEEE Transactions on Software Engineering</i> , vol. 27, pp. 1134-1144, 2001.
<b>S04-R1</b>	M. Vacak, W. F. Tichy, D. I. K. Sjøberg, E. Arisolm, and M. Aldrin, "A controlled experiment comparing the maintainability of programs designed with and without design patterns—a replication in a real programming environment.," <i>Empirical Software Engineering</i> , vol. 9, pp. 149-195, Sept 2004.
<b>S04-R2</b>	L. Prechelt and M. Liesenberg, "Design patterns in software maintenance: An experiment replication at Freie Universit'at Berlin," in <i>Second International Workshop on Replication in Empirical Software Engineering Research (RESER'11)</i> 2011, pp. 1-6.
<b>S04-R3</b>	N. Juristo and S. Vegas, "Design patterns in software maintenance: An experiment replication at UPM - Experiences with the RESER'11 Joint Replication Project," in <i>Second International Workshop on Replication in Empirical Software Engineering Research (RESER'11)</i> , 2011, pp. 7-14.
<b>S04-R4</b>	A. Nanthaamornphong and J. C. Carver, "Design patterns in software maintenance: An experiment replication at University of Alabama," in <i>Second International Workshop on Replication in Empirical Software Engineering Research (RESER'11)</i> , 2011, pp. 15 - 24.
<b>S04-R5</b>	J. L. Krein, L. J. Pratt, A. B. Swenson, A. C. MacLean, C. D. Knutson, and D. L. Eggett, "Design patterns in software maintenance: An experiment replication at Brigham Young University," in <i>Second International Workshop on Replication in Empirical Software Engineering Research (RESER'11)</i> 2011, pp. 25-34.
<b>S05</b>	M. A. Jalil and S. A. M. Noah, "The difficulties of using design patterns among novices: An Exploratory Study," in <i>5th International Conference on Computational Science &amp; Applications (ICCSA'07)</i> , Kuala Lumpur, Malaysia, 2007, pp. 97-103.
<b>S06</b>	E. Chung, J. Hong, M. Prabaker, J. Landay, and A. Liu, "Development and evaluation of emerging design patterns for ubiquitous computing," in <i>Conference on Designing Interactive Systems (DIS'04)</i> , 2004, pp. 233-242.
<b>S07</b>	B. Ellis, J. Stylos, and B. Myers, "The factory pattern in API design: A usability evaluation," in <i>29th International Conference on Software Engineering (ICSE'07)</i> , Minneapolis, MN 2007, pp. 302-311.

Reference Code	Study Bibliographical Reference
S08	S. Jeanmart, Y. G. H. Sahraoui, and N. Habra, "Impact of the visitor pattern on program comprehension and maintenance," in <i>Third International Symposium on Empirical Software Engineering &amp; Measurement (ESEM'09)</i> , Lake Buena Vista, FL, 2009, pp. 69-78.
S09	T. H. Ng, S. C. Cheung, W. K. Chan, and Y. T. Yu, "Toward effective deployment of design patterns for software extension: a case study," in <i>International Workshop on Software Quality (WoSQ'06)</i> , 2006, pp. 51-56.
S10	T. H. Ng, S. C. Cheung, W. K. Chan, and Y. T. Yu, "Do maintainers utilize deployed design patterns effectively?," in <i>29th International Conference on Software Engineering (ICSE'07)</i> , Minneapolis, MN, USA, 2007.
S11	B. Unger and W. Tichy, "Do design patterns improve communication? an experiment with pair design," in <i>International Workshop on Empirical Studies of Software Maintenance</i> , 2000, pp. 1-5.
S12	E. Golden, B. E. John, and L. Bass, "The value of a usability-supporting architectural pattern in software architecture design: A controlled experiment," in <i>27th International Conference on Software Engineering (ICSE'05)</i> , St. Louis, MI, 2005.
S13	R. Porter and P. Calder, "Patterns in learning to program - An experiment?," presented at the <i>Sixth Australasian Conference on Computing Education (ACE'04)</i> , 2004.
S14	T. H. Ng, S. C. Cheung, W. K. Chan, and Y. T. Yu, "Work experience versus refactoring to design patterns: A controlled experiment," in <i>14th ACM SIGSOFT International Symposium on Foundations of Software Engineering. (SIGSOFT'06/FSE-14)</i> Portland, OR, 2006, pp. 12-22.
S15	T. Ihme and P. Abrahamsson, "The use of architectural patterns in the agile software development of mobile applications," in <i>International Conference on Agility (ICAM'05)</i> , Helsinki, Finland, 2005.
S16	T. H. Ng, Y. T. Yu, S. C. Cheung, and W. K. Chan, "Human and program factors affecting the maintenance of programs with deployed design patterns," <i>Information and Software Technology</i> , vol. 54, pp. 99-118, 2012.
S17	U. v. Heesch, P. Avgeriou, U. Zdun, and N. Harrison, "The supportive effect of patterns in architecture decision recovery— A controlled experiment," <i>Science of Computer Programming</i> , vol. 77, pp. 551-576, 2012.
S18	K. Yskout, R. Scandariato, and W. Joosen, "Does organizing security patterns focus architectural choices?," in <i>International Conference on Software Engineering (ICSE '12)</i> , 2012.
S19	J. Abramova, A. Sturma, and P. Shovala, "Evaluation of the Pattern-based method for Secure Development (PbSD): A controlled experiment," <i>Information and Software Technology</i> , vol. 54, pp. 1029-1043, 2012.
S20	A. W. Kamal, P. Avgeriou, and U. Zdun, "The use of pattern participants relationships for integrating patterns: a controlled experiment," <i>Software - Practice and Experience</i> , vol. 43, pp. 807-833, 2013.
S21	G. Mangalaraj, S. Nerur, R. Mahapatra, and K. H. Price, "Distributed cognition in software design: An experimental investigation of the role of design patterns and collaboration," <i>Management Information Systems Quarterly</i> , vol. 38, pp. 249-274, 2014.

## APPENDIX B: Quality Appraisal of Selected Studies

**Categories:** R=Research questions / Hypothesis specified; S=Sample described; P=Patterns details provided; F=Findings reported; L=Limitations / Threats to validity discussed;

**Quality Assessment Score:** 0=Not at all; 1=Somewhat; 2=Mostly; 3= Fully;

Ref. Code	Publication Venue	Year	# of pages	Quality Assessment Score					Average (0-3)
				R	S	P	F	L	
S01 (A, B, C)	Computers and Education, Journal	2010	9	3	3	3	3	1	2.6
S02	Computers and Education, Journal	2008	10	2	2	3	3	3	2.6
S03 (A, B)	IEEE Transactions on Software Engineering, Journal	2002	10	3	3	3	3	3	3
S03-R1	International Conference on Software Maintenance	2002	4	3	2	3	1	0	1.8
S04	IEEE Transactions on Software Engineering, Journal	2001	11	3	3	3	3	3	3
S04-R1	Empirical Software Engineering, Journal	2004	48	3	3	3	3	3	3
S04-R2	Int'l Workshop on Replication in Empirical Software Engineering Research (RESER)	2011	6	3	3	3	2	3	2.8
S04-R3	RESER	2011	8	3	1	3	3	3	2.6
S04-R4	RESER	2011	10	3	3	3	3	3	3
S04-R5	RESER	2011	10	3	3	3	3	3	3
S05	International Conference on Computational Science & Applications	2007	7	2	2	3	3	0	2
S06	Conference on Designing Interactive Systems	2004	10	3	3	3	3	1	2.6
S07	International Conference on Software Engineering (ICSE)	2007	10	2	3	3	3	3	2.8
S08	Empirical Software Engineering and Measurement	2009	10	3	3	3	3	3	3
S09	International Workshop on Software Quality	2006	6	3	3	3	3	3	3
S10	ICSE	2007	10	3	2	3	3	3	2.8
S11	International Workshop on Empirical Studies of Software Maintenance	2000	5	3	3	3	2	3	2.8
S12	ICSE	2005	10	2	3	3	3	1	2.4
S13	Australasian Conference on Computing Education	2004	6	2	1	2	2	1	1.6
S14	International Symposium on Foundations of Software Engineering	2006	11	3	2	3	3	3	2.8
S15	International Conference on Agility	2005	9	3	2	3	3	1	2.4
S16	Information and Software Technology, Journal	2012	29	3	2	3	3	3	2.8
S17	Science of Computer Programming, Journal	2012	26	3	3	3	3	3	3
S18	ICSE	2012	11	3	3	3	3	3	3

Ref. Code	Publication Venue	Year	# of pages	Quality Assessment Score					Average (0-3)
				R	S	P	F	L	
S19	Information and Software Technology, Journal	2012	15	2	2	3	3	3	2.6
S20	Software - Practice and Experience	2013	27	3	2	3	3	3	2.8
S21	Management Information Systems Quarterly	2014	32	3	3	3	3	3	3

## APPENDIX C: Research Questions and Hypotheses of Selected Studies

Reference Code	Research Questions and Hypotheses of Selected Studies (RQs that are inferred are in <i>italics</i> )
S01 (A, B, C)	<ul style="list-style-type: none"> <li>• H<sub>a1</sub>: Novices will faster gain understanding in problem solving and design skills, when they learn to design with the design patterns approach first, before they learn to understand entire systems.</li> <li>• H<sub>a2</sub>: Experienced (domain experts or more experienced designers) will not experience a learning effect from the use of design patterns, but might find them useful in other ways.</li> <li>• R<sub>1</sub>: Training novices with the use of design patterns will increase the quality of the schemas they build to represent a system.</li> </ul>
S02	<ul style="list-style-type: none"> <li>• R<sub>1</sub>: <i>During pattern selection, to what extent are students able to identify applicable design patterns?</i></li> <li>• R<sub>2</sub>: <i>During pattern application, to what extent are students able to comprehend and instantiate design patterns?</i></li> </ul>
S03 (A, B); S03-R1	<ul style="list-style-type: none"> <li>• R<sub>1</sub>: Does it help the maintainer if the design patterns in the program code are documented explicitly (using source code comments) compared to a well-commented program without explicit reference to design patterns?</li> <li>• H<sub>11</sub>: By adding pattern comment lines (PCL), pattern-relevant maintenance tasks are completed faster.</li> <li>• H<sub>12</sub>: By adding PCL, fewer errors are committed in pattern-relevant maintenance tasks.</li> </ul>
S04; S04-R1 through S04-R5	<ul style="list-style-type: none"> <li>• H<sub>01</sub>: Design pattern P does not improve the performance of subjects doing a maintenance exercise X on program A (containing P) when compared to subjects doing the same exercise X on an alternative program A' (not containing P).</li> <li>• R<sub>1</sub>: Do design patterns improve the understandability of software designs? [<i>S04-R4</i>]</li> </ul>
S05	<ul style="list-style-type: none"> <li>• R<sub>1</sub>: <i>What are the problems associated with pattern usage among novices?</i></li> </ul>
S06	<ul style="list-style-type: none"> <li>• R<sub>2</sub>: Are patterns useful for introducing designers to ubicomp?</li> <li>• R<sub>3</sub>: Are patterns useful for communicating between designers? For example, do designers adopt the pattern language vocabulary as they talk about a design?</li> <li>• R<sub>4</sub>: Are patterns useful for creating designs?</li> <li>• R<sub>5</sub>: Are patterns useful for creating higher-quality designs?</li> </ul>
S07	<ul style="list-style-type: none"> <li>• R<sub>1</sub>: <i>How does the Factory pattern compare to using constructors in API design?</i></li> </ul>
S08	<ul style="list-style-type: none"> <li>• H<sub>11</sub>: A class diagram with the Visitor reduces the subjects' efforts during program comprehension when compared to one without it.</li> <li>• H<sub>12</sub>: A class diagram using the canonical representation of the Visitor reduces the subjects' efforts during program comprehension when compared to one using the Visitor and another layout.</li> <li>• H<sub>21</sub>: A class diagram with the Visitor reduces the subjects' efforts during program modification when compared to a class diagram without it.</li> <li>• H<sub>22</sub>: A class diagram using the canonical representation of the Visitor reduces the subjects' efforts during program modification when compared to one using the Visitor and another layout.</li> </ul>
S09	<ul style="list-style-type: none"> <li>• H<sub>01</sub>: When the theme of patterns is not satisfied, there is at least 20% of chance where the Open-Closed principle is followed.</li> <li>• H<sub>02</sub>: When the theme of patterns is satisfied, there is at least 20% of chance where the Open-Closed principle is violated.</li> </ul>
S10	<ul style="list-style-type: none"> <li>• R<sub>1</sub>: What proportion of maintainers utilize the relevant design patterns in completing the required anticipated change?</li> <li>• R<sub>2</sub>: Do maintainers have a higher tendency to perform one task over another?</li> <li>• R<sub>3</sub>: Comparing the codes delivered by maintainers who utilize the relevant design patterns in completing the required anticipated change to those delivered by maintainers who do not utilize patterns, are they equally faulty?</li> </ul>
S11	<ul style="list-style-type: none"> <li>• H<sub>11</sub>: If team members have common design pattern knowledge and vocabulary, they can communicate more effectively than without.</li> </ul>
S12	<ul style="list-style-type: none"> <li>• R<sub>1</sub>: <i>Do the Usability-Supporting Architecture Patterns (USAP) help in considering relevant responsibilities and usability concerns during architectural design?</i></li> </ul>
S13	<ul style="list-style-type: none"> <li>• R<sub>1</sub>: <i>How do patterns help in learning to program?</i></li> </ul>

Reference Code	Research Questions and Hypotheses of Selected Studies (RQs that are inferred are in <i>italics</i> )
S14	<ul style="list-style-type: none"> <li>• R1: Would refactoring a program using design patterns (in)conclusively supersede the effect of work experience to guide maintainers to complete a maintenance task, or vice versa?</li> </ul>
S15	<ul style="list-style-type: none"> <li>• R1: How to select and present patterns in the training material so as to motivate and facilitate their use by the project team?</li> <li>• R2: How can patterns help the team to create new software effectively, while simultaneously maintaining the quality of the developed software at a high level?</li> <li>• R3: How can patterns help the team to develop useful and effective software architectures?</li> <li>• R4: Are the Mobile-D approach with its agile values and the Agile Architecture Line Model suitable to be used with patterns?</li> <li>• R5: How can patterns be used as a documentation aid to preserve vital design information that helps to maintain and evolve the developed product after the project?</li> </ul>
S16	<ul style="list-style-type: none"> <li>• R1: What human and program factors may contribute to the productivity (in terms of time spent) of maintainers in making correct software changes when they work on programs with deployed design patterns (Factors: Deployment of design patterns; Presence of pattern-unaware solutions; Prior exposure to design patterns; Prior exposure to the program; Prior exposure to the programming language; Prior work experience)?</li> </ul>
S17	<ul style="list-style-type: none"> <li>• H11: The quality of recovered decisions is higher when the recovery focuses on identifying patterns in the architecture, compared to ad-hoc, intuitive recovery.</li> <li>• H12: The quantity of recovered decisions is higher when the recovery focuses on identifying patterns in the architecture, compared to ad-hoc, intuitive recovery.</li> </ul>
S18	<ul style="list-style-type: none"> <li>• R1: Does Organizing Security Patterns Focus Architectural Choices?</li> </ul>
S19	<ul style="list-style-type: none"> <li>• R1: <i>How does the use of security patterns affect the quality of resulting database access control specifications when compared to coding the access control specifications directly?</i></li> </ul>
S20	<ul style="list-style-type: none"> <li>• H00: The use of pattern participants relationships does not help software architects to more accurately combine architectural patterns within software architectures as compared with integrating architectural patterns without using such relationships.</li> <li>• H01: The integration of architectural patterns using pattern participants relationships does not result in a more comprehensible software architecture as compared with integrating patterns without the use of such relationships.</li> <li>• H02: The use of pattern participants relationships does not help software architects to better document architectural design decisions as compared with documenting design decisions without using such relationships.</li> <li>• H03: The use of pattern participants relationships does not help software architects to more effectively partition a software architecture into components and sub-components, and assign responsibilities as compared with partitioning a software architecture without the use of such relationships.</li> </ul>

Reference Code	Research Questions and Hypotheses of Selected Studies (RQs that are inferred are in <i>italics</i> )
S21	<ul style="list-style-type: none"> <li>• H1: Quality of the design solution will be higher when design patterns are used during software design.</li> <li>• H2: Time taken to complete a design task will be shorter when design patterns are used.</li> <li>• H3: Solution quality of a collaborating pair will be higher than that of the second-best member of a nominal pair in a software design task.</li> <li>• H4a: Time taken by collaborating pairs to complete a software design task will be longer than that of the best member of a nominal pair.</li> <li>• H4b: Time taken by collaborating pairs to complete a software design task will be longer than that of the second-best member of a nominal pair.</li> <li>• H5a: When design patterns are available, the performance gap in solution quality between the second best member of a nominal pair and the best member of the nominal pair and the pair will be reduced in contrast to when design patterns are not used.</li> <li>• H5b: Solution quality of the second-best member of a nominal pair with design patterns will be higher than that of the collaborating pair without design patterns in a software design task.</li> <li>• H5c: Task completion time of the second-best member of a nominal pair with design patterns will be lower than that of the collaborating pairs without design patterns in a software design task.</li> <li>• H6: Task satisfaction will be higher when design patterns are used in a software design task.</li> <li>• H7: The average level of task satisfaction among collaborating pairs will be higher when compared with the average level of task satisfaction of nominal pairs.</li> </ul>

## APPENDIX D: Prevention Patterns for Security Requirements Engineering

### P-C\_I\_PR-1

- *Name\**: P-C\_I\_PR-1: Prevent unauthorized access to information.
- *Problem\**: <prevent> a breach of <Confidentiality | Integrity | Privacy> of information.
- *Context\**: All access to information should be in accordance with an access control / privacy policy and applicable regulations.
  - Type of the pattern\*: Technical
  - Scope of the pattern\*: Information
- *Solution\**: The system shall:
  - a) enforce that all access to <information | system> are in accordance with <applicable access control policies | access control decisions>.
  - b) enforce that flow of information <within the system | between systems> is in accordance with applicable <information flow policies | access restrictions>.
  - c) <handle | retain> information <within | output from> the system in accordance with <applicable laws | policies | standards | regulations | operational requirements>.
  - d) protect the <confidentiality | integrity | privacy> of information <at rest | in use | in transmission>.
  - e) have provision to <establish | assign | retain> security attributes for information <at rest | in use | in transmission>.
- *Source\**: AC-3, AC-4, AC-16, AC-21, AC-24, SC-16, SC-8, SC-28, SI-12.
- *See Also\**: P-C\_I\_PR-2: Limit authorized access to information, P-C\_I\_PR-3: Restrict access to media, P-C\_PR-1: Prevent information leakage, P-ID-1: Identify and authenticate system users, P-AY-1: Enable auditing of events, D-C\_PR-1: Detect unauthorized disclosure of information, R-C\_PR-1: Respond to unauthorized disclosure of information.

### P-C\_I\_PR-2

- *Name\**: P-C\_I\_PR-2: Limit authorized access to information.
- *Problem\**: <prevent> a breach of <Confidentiality | Integrity | Privacy> of information.
- *Context\**: All access to information should be limited to only necessary accesses required to perform the authorized tasks.
  - Type of the pattern\*: Technical
  - Scope of the pattern\*: Information
- *Solution\**: The organization shall:
  - a) define access authorizations to support separation of duties of <users | processes acting on behalf of users> to reduce the risk of abusing authorized privileges.
  - b) employ least privilege, allowing only authorized accesses for <users | processes acting on behalf of users> which are necessary to accomplish assigned tasks.
- *Source\**: AC-5, AC-6.
- *See Also\**: P-C\_I\_PR-1: Prevent unauthorized access to information, P-C\_PR-1: Prevent information leakage, P-ID-1: Identify and authenticate system users, P-AY-1: Enable auditing of events, D-C\_PR-1: Detect unauthorized disclosure of information, R-C\_PR-1: Respond to unauthorized disclosure of information, D-ALL-1: Monitor for security incidents, D-ALL-2: Analyze audit records to detect malicious activity.

### P-C\_I\_PR-3

- *Name\**: P-C\_I\_PR-3: Restrict access to media.
- *Problem\**: <prevent> a breach of <Confidentiality | Integrity | Privacy> of media containing information.
- *Context\**: The types of media used in the system and types of access to the media containing information should be in accordance with the necessary authorizations.

- Type of the pattern\*: Technical
- Scope of the pattern\*: Information
- *Solution*\*: The organization shall:
  - a) restrict access to <digital | non-digital> media to <authorized users | authorized locations | authorized transfers outside the organization>.
  - b) protect the <digital | non-digital> medial until the media are <destroyed | adequately sanitized | appropriately downgraded>.
  - c) maintain <confidentiality | integrity | privacy | accountability> of <digital | non-digital> media during <transfer> outside of controlled areas.
  - d) <restrict | prohibit> the usage of <certain type of media> on <specific system components>.
- *Source*\*: MP-2, MP-4, MP-5, MP-6, MP-7, MP-8.
- *See Also*\*: P-C\_I\_PR-1: Prevent unauthorized access to information, P-C\_I\_PR-2: Limit authorized access to information, P-C\_PR-1: Prevent information leakage, P-AY-1: Enable auditing of events, D-C\_PR-1: Detect unauthorized disclosure of information, R-C\_PR-1: Respond to unauthorized disclosure of information.

#### P-C\_PR-1

- *Name*\*: P-C\_PR-1: Prevent information leakage.
- *Problem*\*: <prevent> a breach of <Confidentiality | Privacy> of information.
- *Context*\*: The system shall prevent unintended disclosure of information that is not in accordance with an access control / privacy policy and applicable regulations.
  - Type of the pattern\*: Technical
  - Scope of the pattern\*: Information
- *Solution*\*: The system shall:
  - a) prevent <unauthorized | unintended> information transfer via shared <internal | external> resources (e.g., registers, memory, disks).
  - b) validate information output from <defined programs | applications> to ensure consistency with expected content so that information is only disclosed as <authorized | intended>.
  - c) protect against information disclosure during <error handling | authentication> by revealing feedback <without revealing information that could be exploited | only to authorized personnel>.
- *Source*\*: SC-4, SI-11, SI-15, IA-6.
- *See Also*\*: P-C\_I\_PR-1: Prevent unauthorized access to information, P-C\_I\_PR-3: Restrict access to media, P-ALL-1: Enable continuous monitoring, D-C\_PR-1: Detect unauthorized disclosure of information, R-C\_PR-1: Respond to unauthorized disclosure of information.

#### P-C\_PR-2

- *Name*\*: P-C\_PR-2: Prevent unauthorized remote sensing and activation.
- *Problem*\*: <prevent> a breach of <Confidentiality | Privacy> of information accessed through remote sensors.
- *Context*\*: The system allows remote activation of sensors and devices only in exceptional, pre-defined situations.
  - Type of the pattern\*: Technical
  - Scope of the pattern\*: Communication
  - Type of communication channel: Remote
- *Solution*\*: The system shall:
  - a) prohibit remote activation of <environmental sensing capabilities | collaborative computing devices> other than <defined exceptions>.
  - b) provide explicit indication when <sensor | device> is in use to <authorized user>.
- *Source*\*: SC-15, SC-42.

- *See Also*\*: P-C\_I\_PR-1: Prevent unauthorized access to information, P-C\_PR-1: Prevent information leakage, P-CIA\_PR-1: Manage secure internal and external connections, P-ID-1: Identify and authenticate system users, P-AY-1: Enable auditing of events, D-ID-1: Detect malicious activity during authentication.

#### P-AY-1

- *Name*\*: P-AY-1: Enable auditing of events.
- *Problem*\*: <prevent> a breach of <Accountability>.
- *Context*\*: The organization wants to have capability to perform audits in accordance with audit and accountability policy and procedures.
  - *Type of the pattern*\*: Technical
  - *Scope of the pattern*\*: System
- *Solution*\*: The organization shall:
  - a) identify auditable events that can support after-the-fact investigation of security incidents.
  - b) define <frequency | conditions> under which the <auditable events> should be <recorded | reviewed>.
  - c) allocate sufficient audit storage capacity in accordance with <defined audit storage requirements> to avoid <loss of audit records | limitation of audit capability>.
  - d) define methods for coordinating <audit information> when <audit information> is transmitted across organization boundaries.
- *Source*\*: AU-2, AU-4, AU-16.
- *See Also*\*: P-AY-2: Generate audit records and reports, P-AY-3: Protect confidentiality and integrity of audit records, P-ID-1: Identify and authenticate system users, D-ALL-1: Monitor for security incidents, D-ALL-2: Analyze audit records to detect malicious activity.

#### P-AY-2

- *Name*\*: P-AY-2: Generate audit records and reports.
- *Problem*\*: <prevent> a breach of <Accountability>.
- *Context*\*: The system shall generate audit records and reports to support accountability and non-repudiation.
  - *Type of the pattern*\*: Technical
  - *Scope of the pattern*\*: System
- *Solution*\*: The system shall:
  - a) generate audit records containing necessary information including type, time, location, source and outcome of the <auditable event> and identity of <individuals | subjects> associated with the <auditable event>.
  - b) use internal system clocks to generate and record timestamps for <auditable events> at <specified granularity>.
  - c) provide <on-demand | after-the-fact> audit report generation capability.
  - d) protect against <user> denying having performed <defined actions>.
- *Source*\*: AU-3, AU-7, AU-8, AU-10.
- *See Also*\*: P-AY-1: Enable auditing of events, P-AY-3: Protect confidentiality and integrity of audit records, P-ID-1: Identify and authenticate system users, D-ALL-2: Analyze audit records to detect malicious activity.

#### P-AY-3

- *Name*\*: P-AY-3: Protect confidentiality and integrity of audit records.
- *Problem*\*: <prevent> a breach of <Accountability>.
- *Context*\*: The system shall protect the confidentiality and integrity of audit records and any reports generated from the records.

- Type of the pattern\*: Technical
- Scope of the pattern\*: Security / Information
- *Solution\**: The system shall:
  - a) prevent alteration of the original content and ordering of <audit records>.
  - b) protect audit <information | tools> from unauthorized <access | modification | deletion>.
- *Source\**: AU-7, AU-9.
- *See Also\**: P-AY-1: Enable auditing of events, P-AY-2: Generate audit records and reports, D-ALL-1: Monitor for security incidents, D-ALL-2: Analyze audit records to detect malicious activity, R-AY-1: Respond to failures in accountability.

#### P-I-1

- *Name\**: P-I-1: Protect system integrity at runtime.
- *Problem\**: <prevent> a breach of <Integrity> of the system functions.
- *Context\**: The integrity of system functionality should be maintained during execution.
  - Type of the pattern\*: Technical
  - Scope of the pattern\*: System
- *Solution\**:
  - a) The system shall implement a reference monitor to enforce <defined access control policies> that is tamperproof, always invoked, and small enough to be subject to analysis and testing, the completeness of which can be assured.
  - b) The system shall have provision to load and execute <define components | defined applications> from hardware-enforced, read-only media.
  - c) The system shall maintain a separate execution domain for each executing process.
  - d) The system shall check the validity of the <syntax | semantics> of the information input to the system.
  - e) The organization shall employ <isolated execution environment | virtualized sandbox> for <defined systems | defined components | defined locations> to <quickly identify malicious code | reduce likelihood of malicious code propagation>.
- *Source\**: AC-25, SC-34, SC-39, SC-44, SI-10, SI-16.
- *See Also\**: P-C\_PR-1: Prevent information leakage, P-I\_A-1: Limit unnecessary exposure of system functionality, P-ALL-1: Enable continuous monitoring, P-ID-2: Establish authenticity of system components, P-ALL-2: Limit system exposure to persistent attackers, P-ALL-3: Deceive persistent attackers, D-I-1: Detect malicious system modification attempts, D-I-2: Detect vulnerabilities in the system, R-I-1: Respond to vulnerabilities in the system.

#### P-I A-1

- *Name\**: P-I\_A-1: Limit unnecessary exposure of system functionality.
- *Problem\**: <prevent | respond to> a breach of <Integrity | Availability> of the system functions.
- *Context\**: The integrity of system functionality should be maintained by limiting unnecessary exposure.
  - Type of the pattern\*: Technical
  - Scope of the pattern\*: System
- *Solution\**: The organization shall:
  - a) <physically | logically> separate user functionality from information system management functionality.
  - b) employ <diverse set of technologies | physical distribution of processing and storage | logical separation of processing and storage> so that an attack on one part of the system does not affect other parts of the system.
  - c) isolate security functions from non-security functions.
  - d) physically partition <defined components> based on <defined circumstances>.
  - e) physically <disable | remove> <defined connection ports | input devices> on <defined components>.

- *Source\**: SC-2, SC-3, SC-29, SC-32, SC-36, SC-41.
- *See Also\**: P-C\_PR-1: Prevent information leakage, P-I-1: Protect system integrity at runtime, P-ALL-2: Limit system exposure to persistent attackers, P-ALL-3: Deceive persistent attackers, D-I-3: Detect problems with security functions.

#### P-ID-1

- *Name\**: P-ID-1: Identify and authenticate system users.
- *Problem\**: <prevent> a breach of <Identification and authentication> of the system users.
- *Context\**: The identity and authenticity of system users should be established and maintained through each user session.
  - *Type of the pattern\**: Technical
  - *Scope of the pattern\**: User
- *Solution\**: The system shall:
  - a) uniquely identify and authenticate <organizational | non-organizational> <users | processes acting on behalf of users | services | devices> before allowing access to the system <locally | via a network>.
  - b) allow access to the system only after <establishing | re-establishing> identification and authentication procedures.
  - c) explicitly specify and document rationale for <defined actions> that are allowed without identification or authentication.
  - d) limit the number of concurrent sessions for <user account type> to <defined limit>.
  - e) prevent access to the system by <locking | terminating> the session <at user's request | after defined inactive period>.
- *Source\**: IA-2, IA-3, IA-8, IA-9, AC-10, AC-11, AC-14.
- *See Also\**: P-AY-1: Enable auditing of events, P-C\_PR-1: Prevent information leakage, P-C\_I\_ID-1: Manage security of identifiers and authenticators, P-C\_I\_ID-2: Manage security of cryptographic functions, P-C\_I\_ID-3: Manage security of communication session, D-ID-1: Detect malicious activity during authentication, R-ID-1: Respond to malicious activity during authentication.

#### P-ID-2

- *Name\**: P-ID-2: Establish authenticity of system components.
- *Problem\**: <prevent> a breach of <Identification and authentication> of the system components.
- *Context\**: The authenticity of system components should be established.
  - *Type of the pattern\**: Technical
  - *Scope of the pattern\**: System
- *Solution\**: The organization shall:
  - a) detect and prevent counterfeit components from entering the system based on <defined anti-counterfeit policy and procedures>.
  - b) report counterfeit components to <source of the counterfeit component | authorized organizations | authorized personnel>.
- *Source\**: SA-19.
- *See Also\**: P-AY-1: Enable auditing of events, P-I-1: Protect system integrity at runtime, D-I-1: Detect malicious system modification attempts.

#### P-C\_I\_ID-1

- *Name\**: P-C\_I\_ID-1: Manage security of identifiers and authenticators.
- *Problem\**: <prevent> a breach of <Confidentiality | Integrity | Identification & Authentication> during authentication.
- *Context\**: The organization uses identifiers and authenticators to establish identity and authorization for the security of the system. The security of the identifiers and authenticators themselves must be protected.

- Type of the pattern\*: Technical
- Scope of the pattern\*: Security
- *Solution\**: The system shall:
  - a) assign unique identifiers for <individuals | groups | roles | services | devices> only after receiving necessary authorizations.
  - b) prevent the reuse of identifiers for <defined time period>.
  - c) disable identifiers <after defined inactive period>.
  - d) verify identity of <individual | group | role | service | device> before initially distributing the authenticator.
  - e) <establish | implement> procedures for <initial authenticator distribution | updating the authenticator | a lost or compromised authenticator | a damaged authenticator | revoking authenticators>.
  - f) establish <initial content | lifetime restrictions | reuse conditions> for authenticators.
  - g) protect authenticator content from unauthorized <disclosure | modification>.
  - h) change authenticators in the event of <system installation | change in group membership | change of role | expiration based on defined time>.
- *Source\**: IA-4, IA-5.
- *See Also\**: P-ID-1: Identify and authenticate system users, P-C\_PR-1: Prevent information leakage, D-ID-1: Detect malicious activity during authentication, R-ID-1: Respond to malicious activity during authentication, P-C\_I\_ID-2: Manage security of cryptographic functions, R-SEC-1: Respond to threats to security functions.

#### P-C I ID-2

- *Name\**: P-C\_I\_ID-2: Manage security of cryptographic functions.
- *Problem\**: <prevent> a breach of <Confidentiality | Integrity | Identification & Authentication> of cryptographic functions.
- *Context\**: The organization uses cryptographic functions for confidentiality and integrity of information. The security of the cryptographic functions themselves must be protected.
  - Type of the pattern\*: Technical
  - Scope of the pattern\*: Security
- *Solution\**: The organization:
  - a) <establishes | manages | implements> cryptographic functions in accordance with <applicable laws | regulations | policies | standards>.
  - b) implements mechanism for authentication to cryptographic modules that meet the requirements of <applicable laws | regulations | policies | standards>.
  - c) issues public key certificates in accordance with <a defined certificate policy>
  - d) obtains certificates from <an approved service provider>.
- *Source\**: SC-12, SC-13, SC-17, IA-7.
- *See Also\**: P-C\_I\_ID-1: Manage security of identifiers and authenticators, P-C\_PR-1: Prevent information leakage, R-SEC-1: Respond to threats to security functions.

#### P-C I ID-3

- *Name\**: P-C\_I\_ID-3: Manage security of communication session.
- *Problem\**: <prevent> a breach of <Confidentiality | Integrity | Identification & Authentication> of communication sessions.
- *Context\**: The organization allows communication sessions over a network. The security of the established communication sessions themselves must be protected.
  - Type of the pattern\*: Technical
  - Scope of the pattern\*: Security / Communication
  - Type of communication channel: Local, Network, Wireless.

- *Solution\**: The system shall:
  - a) establish a trusted communications path with the user for <authentication | re-authentication | other security functions>.
  - b) terminate network connections associated with a communications session <at session end | after defined inactive period>.
  - c) protect the authenticity of communications sessions.
- *Source\**: SC-10, SC-11, SC-23.
- *See Also\**: P-ID-1: Identify and authenticate system users, P-C\_PR-1: Prevent information leakage, P-I\_A\_ID-1: Setup secure name and address resolution functions, P-CIA\_PR-1: Manage secure internal and external connections.

#### P-CIA PR-1

- *Name\**: P-CIA\_PR-1: Manage secure internal and external connections.
- *Problem\**: <prevent> a breach of <Confidentiality | Integrity | Availability | Privacy> when establishing connections.
- *Context\**: The organization allows connections with components, devices, networks and systems internal or external to the organization.
  - *Type of the pattern\**: Technical
  - *Scope of the pattern\**: Communication
  - *Type of communication channel*: Local, Network, Wireless
- *Solution\**: The organization shall:
  - a) authorize internal connections with <defined component classes (e.g., printers, scanners, mobile devices)>.
  - b) document <interface characteristics | security requirements | nature of information communicated> for each internal connection with the <defined component class>.
  - c) allow connection to external networks only through managed interfaces (e.g., gateways, routers, firewalls, encrypted tunnels) in accordance with <organizational security architecture>.
  - d) monitor and control communication at the <external boundary | key internal boundaries> of the system.
  - e) implement subnetworks for publicly accessible system components that are <logically | physically> separated from internal organizational networks.
  - f) protect <external | internal> <defined wireless links> from <information disclosure | denial of service | spoofing | defined types of attacks>.
- *Source\**: CA-9, SC-7, SC-40.
- *See Also\**: P-C\_I\_PR-1: Prevent unauthorized access to information, P-C\_PR-1: Prevent information leakage, P-C\_PR-2: Prevent unauthorized remote sensing and activation, P-C\_I\_ID-2: Manage security of cryptographic functions, P-C\_I\_ID-3: Manage security of communication session.

#### P-I A ID-1

- *Name\**: P-I\_A\_ID-1: Setup secure name and address resolution functions.
- *Problem\**: <prevent> a breach of < Integrity | Availability | Identification & Authentication> when establishing communication.
- *Context\**: The system shall be able to verify integrity and authenticity of the source of information across a network.
  - *Type of the pattern\**: Technical
  - *Scope of the pattern\**: Communication
  - *Type of communication channel*: Local network, remote network, wireless network.
- *Solution\**: The system shall:
  - a) have provision that the <name | address> resolution services employ separate servers for internal and external roles.

- b) implement any <name | address> resolution services in a fault-tolerant manner by employing <redundancy | geographical distribution>.
- c) perform <data origin authentication | data integrity verification> for <name | address> resolution services provided by <external authoritative source | recursive resolvers | caching resolvers>.
- d) enable verification of <child zones | chain of trust among parent and child domains> for <distributed | hierarchical> namespaces.
- *Source\**: SC-20, SC-21, SC-22.
- *See Also\**: P-C\_I\_ID-3: Manage security of communication session, P-CIA\_PR-1: Manage secure internal and external connections, D-ID-1: Detect malicious activity during authentication, R-ID-1: Respond to malicious activity during authentication.

#### P-ALL-1

- *Name\**: P-ALL-1: Enable continuous monitoring.
- *Problem\**: <prevent | detect | respond to> a breach of < Confidentiality | Integrity | Availability | Identification & Authentication | Accountability | Privacy>.
- *Context\**: The organization shall enable continuous monitoring of events.
  - *Type of the pattern\**: Technical
  - *Scope of the pattern\**: System
- *Solution\**: The organization shall:
  - a) <develop | implement> a continuous monitoring strategy.
  - b) define <metrics> to be monitored and <frequency> for <monitoring | security control assessments>.
  - c) perform <security control assessments | security status monitoring of defined metrics> in accordance with the continuous monitoring strategy.
  - d) correlate information from <monitoring | assessment> activities and take appropriate response actions.
- *Source\**: CA-7.
- *See Also\**: P-I-1: Protect system integrity at runtime, P-AY-1: Enable auditing of events, D-I-1: Detect malicious system modification attempts, D-ALL-1: Monitor for security incidents, D-ALL-2: Analyze audit records to detect malicious activity, R-CIA-1: Respond to system failure conditions, R-AY-1: Respond to failures in accountability, R-SEC-1: Respond to threats to security functions.

#### P-ALL-2

- *Name\**: P-ALL-2: Limit system exposure to persistent attackers.
- *Problem\**: <prevent> a breach of <Confidentiality | Integrity | Availability | Identification & Authentication | Accountability | Privacy> of the system.
- *Context\**: Advanced persistent threats (APTs), e.g., attackers who persists in their attempts to gain unauthorized access to the system, exist. Limiting the system functionality available to the attackers may safeguard the system.
  - *Type of the pattern\**: Technical
  - *Scope of the pattern\**: System
  - *Type of threat*: Advanced persistent threat
- *Solution\**: The organization may:
  - a) employ <defined components> with minimal <functionality | information storage>.
  - b) may implement non-persistent <defined components | defined services> that are initiated in a known state and terminated <upon end of session | periodically at defined frequency> to limit opportunity for attack.
- *Source\**: SC-25, SI-14.
- *See Also\**: P-C\_PR-1: Prevent information leakage, P-I\_A-1: Limit unnecessary exposure of system functionality, P-ALL-1: Enable continuous monitoring, P-ALL-3: Deceive persistent attackers, D-ALL-

1: Monitor for security incidents, D-ALL-2: Analyze audit records to detect malicious activity, R-CIA-1: Respond to system failure conditions, R-SEC-1: Respond to threats to security functions.

### P-ALL-3

- *Name\**: P-ALL-3: Deceive persistent attackers.
- *Problem\**: <prevent> a breach of <Confidentiality | Integrity | Availability | Identification & Authentication | Accountability | Privacy> of the system.
- *Context\**: Advanced persistent threats (APTs), e.g., attackers who persists in their attempts to gain unauthorized access to the system, exist. Deceiving the attackers may provide valuable time to safeguard the system.
  - *Type of the pattern\**: Technical
  - *Scope of the pattern\**: System
  - *Type of threat*: Advanced persistent threat
- *Solution\**: The organization may:
  - a) setup easier targets for the attackers such as <honeypots | others>.
  - b) employ tactics such as <randomness | uncertainty | virtualization > to confuse and mislead the attackers.
- *Source\**: SC-26, SC-30.
- *See Also\**: P-C\_PR-1: Prevent information leakage, P-ALL-1: Enable continuous monitoring, P-ALL-2: Limit system exposure to persistent attackers, D-ALL-1: Monitor for security incidents, D-ALL-2: Analyze audit records to detect malicious activity, R-CIA-1: Respond to system failure conditions, R-SEC-1: Respond to threats to security functions.

## APPENDIX E: Detection Patterns for Security Requirements Engineering

### D-C PR-1

- *Name\**: D-C\_PR-1: Detect unauthorized disclosure of information.
- *Problem\**: <detect> a breach of <Confidentiality | Privacy> of information.
- *Context\**: All disclosures of information should be in accordance with an access control / privacy policy and applicable regulations. Any disclosure not in accordance with the policy should be detected.
  - Type of the pattern\*: Technical
  - Scope of the pattern\*: Information
- *Solution\**: The system shall detect unauthorized disclosures of <resource> through:
  - a) monitoring the status and location of system components that <store> unencrypted <resource>.
  - b) monitoring when <resource> leaves the secure system boundary during <storage | transfer>.
  - c) detecting malicious attempts at gaining access to information during <transfer> by exploiting the <transmission protocols>.
  - d) protecting against and detecting unauthorized mining of <resource> during <read | store | transfer>
  - e) performing analysis of potential covert channels that can be used to <read | transfer> the <resource>.
- *Source\**: AU-13, SC-31, AC-23, SC-19.
- *See Also\**: P-C\_I\_PR-1: Prevent unauthorized access to information, P-C\_I\_PR-2: Limit authorized access to information, P-C\_I\_PR-3: Restrict access to media, P-C\_PR-1: Prevent information leakage, R-C\_PR-1: Respond to unauthorized disclosure of information, P-ALL-1: Enable continuous monitoring, D-ALL-1: Monitor for security incidents, D-ALL-2: Analyze audit records to detect malicious activity.

### D-ID-1

- *Name\**: D-ID-1: Detect malicious activity during authentication.
- *Problem\**: <detect> a breach of <Identification & Authentication> of the user.
- *Context\**: An attacker may try to authenticate as a regular user and gain unauthorized access to the system.
  - Type of the pattern\*: Technical
  - Scope of the pattern\*: User
- *Solution\**: The system shall maintain a record of:
  - a) unsuccessful logon attempts by <user>.
  - b) previous logon <time | location> that the <user> accessed the <resource> to <detect | alert authorized user about> suspicious activity.
  - c) frequent <location | device> from where the <user> accesses the system to <detect | alert authorized user about> deviations from the norm that may indicate potentially malicious activity.
- *Source\**: AC-7, AC-9.
- *See Also\**: P-ID-1: Identify and authenticate system users, P-C\_I\_ID-1: Manage security of identifiers and authenticators, R-ID-1: Respond to malicious activity during authentication, P-ALL-1: Enable continuous monitoring, D-ALL-1: Monitor for security incidents, D-ALL-2: Analyze audit records to detect malicious activity.

### D-I-1

- *Name\**: D-I-1: Detect malicious system modification attempts.
- *Problem\**: <detect> a breach of <Integrity> of the system.
- *Context\**: An attacker may try to modify parts of the system functionality to gain unauthorized access to the system.
  - Type of the pattern\*: Technical
  - Scope of the pattern\*: System

- *Solution\**: The organization shall:
  - a) detect if a code is untrusted or malicious through <vulnerability scanning | tamper resistance and detection | spam protection>.
  - b) monitor and maintain record of any <local | remote> maintenance activities performed on the system.
  - c) monitor and maintain record of any <local | remote> tools used for maintenance.
  - d) check for <integrity | authenticity> of <external components> that are included as part of the system.
- *Source\**: SA-18, SA-19, SC-18, SI-3, SI-7, SI-8, MA-2, MA-3, MA-4.
- *See Also\**: P-I-1: Protect system integrity at runtime, P-ID-2: Establish authenticity of system components, D-I-2: Detect vulnerabilities in the system, D-I-3: Detect problems with security functions, D-ALL-1: Monitor for security incidents, D-ALL-2: Analyze audit records to detect malicious activity.

#### D-I-2

- *Name\**: D-I-2: Detect vulnerabilities in the system.
- *Problem\**: <prevent | detect | respond to> a breach of <Integrity> of the system.
- *Context\**: The system may contain vulnerabilities that an attacker can exploit to gain unauthorized access to the system.
  - *Type of the pattern\**: Technical
  - *Scope of the pattern\**: System
- *Solution\**: The system shall:
  - a) scan for vulnerabilities in <platforms | software | configurations> using <defined vulnerability scanning tools | defined techniques>.
  - b) conduct the vulnerability scans <periodically | as new vulnerabilities become known>.
- *Source\**: RA-5.
- *See Also\**: P-I-1: Protect system integrity at runtime, P-I\_A-1: Limit unnecessary exposure of system functionality, P-C\_I\_ID-1: Manage security of identifiers and authenticators, R-I-1: Respond to vulnerabilities in the system, P-ALL-1: Enable continuous monitoring, D-ALL-1: Monitor for security incidents, D-ALL-2: Analyze audit records to detect malicious activity.

#### D-I-3

- *Name\**: D-I-3: Detect problems with security functions.
- *Problem\**: <detect> a breach of <Integrity> of the security functions.
- *Context\**: An attacker may try to compromise or bypass the security mechanisms themselves to gain unauthorized access to the system.
  - *Type of the pattern\**: Technical
  - *Scope of the pattern\**: Security
- *Solution\**: The system shall
  - a) verify integrity of security functions and notify <authorized individual> if any problems are identified.
  - b) record any <local | remote> maintenance activities performed on the system or any tools used for maintenance.
- *Source\**: SI-6.
- *See Also\**: P-I\_A-1: Limit unnecessary exposure of system functionality, D-I-1: Detect malicious system modification attempts, P-C\_I\_ID-1: Manage security of identifiers and authenticators, P-ALL-1: Enable continuous monitoring, D-ALL-1: Monitor for security incidents, D-ALL-2: Analyze audit records to detect malicious activity.

#### D-ALL-1

- *Name\**: D-ALL-1: Monitor for security incidents.
- *Problem\**: <detect > a breach of <Confidentiality | Integrity | Availability | Identification & Authentication | Accountability | Privacy> of the system.
- *Context\**: Security incidents, such as intrusion or unauthorized access to the system, should not go unnoticed.
  - *Type of the pattern\**: Technical
  - *Scope of the pattern\**: System
- *Solution\**: The system shall:
  - a) setup mechanisms for monitoring security incidents related to <potential attacks | unauthorized connections> for <resource>.
  - b) have provisions to heighten the level of <monitoring activities> if an attack is detected or perceived.
  - c) setup notifications about the usage of system to detect <unauthorized access> of <resource>.
  - d) protect the <confidentiality | integrity> of information obtained from the intrusion monitoring tools.
- *Source\**: IR-4, IR-5, IR-6, SI-4, AC-8.
- *See Also\**: P-ALL-1: Enable continuous monitoring, D-I-1: Detect malicious system modification attempts, D-ALL-2: Analyze audit records to detect malicious activity, R-CIA-1: Respond to system failure conditions, R-AY-1: Respond to failures in accountability.

#### D-ALL-2

- *Name\**: D-ALL-2: Analyze audit records to detect malicious activity.
- *Problem\**: <detect > a breach of <Confidentiality | Integrity | Availability | Identification & Authentication | Accountability | Privacy> of the system.
- *Context\**: Regular audits can help uncover malicious activity in the system.
  - *Type of the pattern\**: Technical
  - *Scope of the pattern\**: System
- *Solution\**: The system shall:
  - a) setup mechanisms for <review | analysis | reporting> of audit records related to <system access | maintenance>.
  - b) retain audit records for <resource> for a <designated period> in accordance with applicable <policy | law>.
  - c) have mechanisms to generate activity for <resource> based on audit records.
  - d) provide <authorized users> with ability to <capture | record | view | hear> <user session>.
- *Source\**: AU-6, AU-11, AU-12, AU-14.
- *See Also\**: P-AY-1: Enable auditing of events, P-AY-2: Generate audit records and reports, P-AY-3: Protect confidentiality and integrity of audit records, D-C\_PR-1: Detect unauthorized disclosure of information, D-ID-1: Detect malicious activity during authentication, D-ALL-1: Monitor for security incidents, R-AY-1: Respond to failures in accountability.

## APPENDIX F: Response Patterns for Security Requirements Engineering

### R-C PR-1

- *Name\**: R-C\_PR-1: Respond to unauthorized disclosure of information.
- *Problem\**: <respond to> a breach of <Confidentiality | Privacy> of information.
- *Context\**: In case of unauthorized disclosure of information, the organization shall take appropriate remedial actions in accordance with the appropriate policies and regulations.
  - Type of the pattern\*: Technical
  - Scope of the pattern\*: Information
- *Solution\**: The system shall:
  - a) respond to information spillage related to <resource> by <notifying authorize user | isolating compromised components | removing information from the compromised components>.
  - b) proactively assess if any additional components may have been compromised by the information spillage related to <resource>.
- *Source\**: IR-9.
- *See Also\**: P-ALL-2: Limit system exposure to persistent attackers, P-C\_I\_PR-1: Prevent unauthorized access to information, D-C\_PR-1: Detect unauthorized disclosure of information, D-ALL-1: Monitor for security incidents, D-ALL-2: Analyze audit records to detect malicious activity, R-CIA-1: Respond to system failure conditions.

### R-ID-1

- *Name\**: R-ID-1: Respond to malicious activity during authentication.
- *Problem\**: <prevent | respond to> a breach of <Identification & Authentication>.
- *Context\**: An attacker may try to pose as a regular user and gain unauthorized access to the system.
  - Type of the pattern\*: Technical
  - Scope of the pattern\*: User
- *Solution\**: The system shall:
  - a) have provision to lock the <local | remote> session of <user> if a breach of security is detected.
  - b) have provision to terminate the <local | remote> session of <user> if a breach of security is detected.
  - c) have provision to request additional authentication from <user> if a potential threat is detected.
  - d) require <user> to re-authenticate if a potential threat is detected.
- *Source\**: AC-11, AC-12, IA-10, IA-11.
- *See Also\**: P-ID-1: Identify and authenticate system users, P-C\_I\_ID-1: Manage security of identifiers and authenticators, D-ID-1: Detect malicious activity during authentication, D-ALL-1: Monitor for security incidents, D-ALL-2: Analyze audit records to detect malicious activity, R-CIA-1: Respond to system failure conditions, P-ALL-2: Limit system exposure to persistent attackers.

### R-I-1

- *Name\**: R-I-1: Respond to vulnerabilities in the system.
- *Problem\**: <respond to> a breach of <Integrity> of the system.
- *Context\**: The system may contain vulnerabilities that can be exploited by an attacker.
  - Type of the pattern\*: Technical
  - Scope of the pattern\*: System
- *Solution\**: The system shall:
  - a) analyze vulnerabilities detected in <platforms | software | configurations> for potential harmful impact.
  - b) take steps to remediate legitimate vulnerabilities in <the given system | other information systems with potentially similar vulnerabilities>.

- *Source\**: RA-5.
- *See Also\**: P-I-1: Protect system integrity at runtime, P-I\_A-1: Limit unnecessary exposure of system functionality, D-I-2: Detect vulnerabilities in the system, D-ALL-1: Monitor for security incidents, D-ALL-2: Analyze audit records to detect malicious activity, P-ALL-2: Limit system exposure to persistent attackers.

#### R-A-1

- *Name\**: R-A-1: Setup backup and recovery procedures.
- *Problem\**: <prevent | respond to> a breach of <Availability> of information assets.
- *Context\**: An attacker may compromise the system and render the information assets unavailable.
  - *Type of the pattern\**: Technical
  - *Scope of the pattern\**: Information
- *Solution\**: The system shall:
  - a) conduct regular backup of <user | system | security> information.
  - b) have provision to protect the <confidentiality | integrity> of backup information similar to the original information.
  - c) be able to recover from backups of <user | system | security> information.
- *Source\**: CP-9, CP-10.
- *See Also\**: P-C\_I\_PR-1: Prevent unauthorized access to information, D-C\_PR-1: Detect unauthorized disclosure of information, R-A-2: Prevent and respond to denial of service, R-AY-1: Respond to failures in accountability, R-CIA-1: Respond to system failure conditions.

#### R-A-2

- *Name\**: R-A-2: Prevent and respond to denial of service.
- *Problem\**: <prevent | respond to> a breach of <Availability> of the system.
- *Context\**: An attacker may compromise the system and render parts of the information system unavailable.
  - *Type of the pattern\**: Technical
  - *Scope of the pattern\**: System
- *Solution\**: The system shall:
  - a) <protect against | limit> the effects of the denial of service attacks on <system components>.
  - b) <protect | limit> the availability of shared resources by allocating <defined resources> based on <priority | quota> of requesting <processes | users>.
  - c) include platform-independent applications for <organization-defined functions> to <promote portability | increase availability> in case a specific platform is under attack.
- *Source\**: SC-5, SC-6, SC-27.
- *See Also\**: P-ALL-1: Enable continuous monitoring, D-ALL-1: Monitor for security incidents, D-ALL-2: Analyze audit records to detect malicious activity, R-ALL-1: Respond to security incidents in the system, R-CIA-1: Respond to system failure conditions.

#### R-AY-1

- *Name\**: R-AY-1: Respond to failures in accountability.
- *Problem\**: <respond to> a breach of <Accountability> of user actions.
- *Context\**: An attacker may compromise the mechanisms for accountability to covertly perform malicious activity in the system. Failures in accountability may also occur due to non-malicious activity.
  - *Type of the pattern\**: Technical
  - *Scope of the pattern\**: Security
- *Solution\**: The system shall:
  - a) respond to audit processing failures by alerting <authorized user>.

- b) provision for alternate audit capability to record <designated actions> if the primary audit capability fails.
- *Source\**: AU-5, AU-15.
- *See Also\**: P-ALL-1: Enable continuous monitoring, P-ALL-2: Limit system exposure to persistent attackers, D-ALL-1: Monitor for security incidents, D-ALL-2: Analyze audit records to detect malicious activity, R-ALL-1: Respond to security incidents in the system, R-A-1: Setup backup and recovery procedures.

#### R-CIA-1

- *Name\**: R-CIA-1: Respond to system failure conditions.
- *Problem\**: <prevent | respond to> a breach of <Confidentiality | Integrity | Availability> of the system.
- *Context\**: The system should respond to failure conditions such that security of the system is not further compromised beyond the current breach.
  - *Type of the pattern\**: Technical
  - *Scope of the pattern\**: System
- *Solution\**: The system shall:
  - a) have provision to switch operations to safe mode when <organization defined conditions> are detected.
  - b) fail in a known state in response to <security attacks> to help prevent further loss to security of the system.
  - c) proactively provide alternate <system components> in case of predictable failures (e.g., based on mean time to failure).
  - d) implement fail-safe procedures such as <alerting authorized personnel | reestablish system settings | shut down | restart > when <predefined failure conditions> occur.
- *Source\**: CP-12, SC-24, SI-13, SI-17.
- *See Also\**: P-ALL-1: Enable continuous monitoring, P-ALL-2: Limit system exposure to persistent attackers, P-ALL-3: Deceive persistent attackers, D-ALL-1: Monitor for security incidents, R-A-1: Setup backup and recovery procedures, R-A-2: Prevent and respond to denial of service, R-AY-1: Respond to failures in accountability.

#### R-SEC-1

- *Name\**: R-SEC-1: Respond to threats to security functions.
- *Problem\**: <respond to> a breach of <Confidentiality | Integrity | Availability | Identification & Authentication | Accountability | Privacy> of the security functions.
- *Context\**: The system should handle situations in which the security functions and mechanisms may themselves be compromised.
  - *Type of the pattern\**: Technical
  - *Scope of the pattern\**: Security
- *Solution\**: The system:
  - a) may provide alternate <security mechanisms> in case a threat to <security mechanism> is detected (e.g., biometrics, temporary passwords, alternate communication protocols).
  - b) may employ separate communication channels to share security sensitive information including <identifiers | configuration information | cryptographic keys | security updates | system backups>.
  - c) shall have provision to <shut down | restart | other actions> the system when anomalies are discovered during security functions verification.
- *Source\**: CP-11, CP-13, SC-37, SI-6.
- *See Also\**: P-C\_I\_ID-1: Manage security of identifiers and authenticators, P-C\_I\_ID-2: Manage security of cryptographic functions, P-C\_I\_ID-3: Manage security of communication session, P-ALL-1: Enable continuous monitoring, P-ALL-2: Limit system exposure to persistent attackers, P-ALL-3: Deceive

persistent attackers, D-ALL-1: Monitor for security incidents, R-CIA-1: Respond to system failure conditions, R-AY-1: Respond to failures in accountability.