STIMULATING ARCHITECTURAL THINKING DURING REQUIREMENTS ELICITATION

An Approach and Tool Support

Preethu Rose Anish
STIMULATING ARCHITECTURAL THINKING DURING REQUIREMENTS ELICITATION
AN APPROACH AND TOOL SUPPORT

Preethu Rose Anish
STIMULATING ARCHITECTURAL THINKING DURING REQUIREMENTS ELICITATION
AN APPROACH AND TOOL SUPPORT

DISsertation

to obtain
the degree of doctor at the University of Twente,
on the authority of the rector magnificus,
prof.dr. T.T.M. Palstra,
on account of the decision of the Doctorate Board,
to be publicly defended
on wednesday, the 11th of March 2020 at 12.45 hours

by

Preethu Rose Anish
born on the 10th of October 1982,
in New Delhi, India
This dissertation has been approved by:

Supervisor
prof. dr. R. J. Wieringa

Co-supervisor
dr. M. Daneva
Graduation Committee:

Chairman
prof. dr. J.N. Kok
(University of Twente, The Netherlands)

Promoter
prof. dr. R.J. Wieringa
(University of Twente, The Netherlands)

Copromoter
dr. M. Daneva
(University of Twente, The Netherlands)

Committee Members:
prof. dr. J. van Hillegersberg
(University of Twente, The Netherlands)
dr. M.J. van Sinderen
(University of Twente, The Netherlands)
prof. dr. P. Lago
(Free University of Amsterdam, The Netherlands)
prof. dr. J. Cleland-Huang
(University of Notre Dame, United States)
dr. S. Ghaisas
(TCS Research, India)
# Table of Contents

Acknowledgement.................................................................................................................. i
Summary................................................................................................................................... v
Samenvatting............................................................................................................................. vii
List of Figures......................................................................................................................... ix
List of Tables.......................................................................................................................... xi
List of Appendixes.................................................................................................................. xiii
List of Abbreviations.............................................................................................................. xv

1 Introduction........................................................................................................................... 1
  1.1 Research Problem and Motivation.................................................................................... 1
  1.2 Definition of Terms......................................................................................................... 2
  1.3 Scope of Research........................................................................................................... 5
  1.4 Research Objective......................................................................................................... 6
  1.5 Research Questions........................................................................................................ 6
  1.6 Research Methodology at a Glance ................................................................................ 8
  1.7 Contributions.................................................................................................................. 11
  1.8 Implications..................................................................................................................... 11
  1.9 Publications..................................................................................................................... 12
  1.10 Outline of the Dissertation............................................................................................ 13

2 Research Methodology........................................................................................................... 15
  2.1 Research Approach......................................................................................................... 15
  2.2 Research Strategy........................................................................................................... 16
    2.2.1 Literature Review..................................................................................................... 16
    2.2.2 Case Study............................................................................................................... 17
    2.2.3 Survey..................................................................................................................... 18
    2.2.4 Experiment.............................................................................................................. 19
  2.3 Research Methods........................................................................................................... 19
    2.3.1 Semi-structured Interviews..................................................................................... 20
2.3.2 Questionnaire ................................................................. 20
2.3.3 Document Analysis ......................................................... 21

2.4 Threat to Validity of Research Results ........................................... 21
2.4.1 Construct Validity ............................................................... 21
2.4.2 Internal Validity ................................................................. 22
2.4.3 External Validity ................................................................. 23

2.5 Summary ............................................................................. 23

3 Related Work ........................................................................ 25
3.1 Introduction ........................................................................ 25
3.2 Architecturally Significant Functional Requirements in RE and SA literature ...................................................... 25
3.3 The Synergistic Relationship between Requirements and Architectural Design ........................................................... 27
3.4 Automated Identification and Classification of Architecturally Significant Requirements ............................................. 29
3.5 Knowledge Based Approaches to Organizing Architecturally Relevant Requirements Knowledge .................................................. 30
3.6 Summary ............................................................................. 32

4 Understanding Architecturally Significant Functional Requirements (ASFRs) ........................................ 33
4.1 Introduction ........................................................................ 33
4.2 The Research Plan and Execution Strategy for the Exploratory Studies ........................................................... 34
4.2.1 Data Collection ................................................................... 34
4.2.2 Data Analysis ................................................................... 35
4.3 The Phase I Study ................................................................. 36
4.4 The Phase II Study ................................................................. 37
4.4.1 Results ............................................................................. 38
4.5 Discussions .......................................................................... 47
4.6 Quantitative Evaluation Study ................................................. 49
4.7 Validity Threats .................................................................... 51
4.8 Conclusion ........................................................................... 51

5 Creating Probing Question Flows ........................................................................ 53
5.1 Introduction ........................................................................ 53
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2</td>
<td>Construction of PQ-flows ..................................................................</td>
<td>54</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Interview with the Software Architects</td>
<td>55</td>
</tr>
<tr>
<td>5.2.2</td>
<td>The Online Survey</td>
<td>55</td>
</tr>
<tr>
<td>5.2.2.1</td>
<td>Online Survey Phase 1 and Analysis</td>
<td>58</td>
</tr>
<tr>
<td>5.2.2.2</td>
<td>Online Survey Phase 2 and Analysis – Enriching the PQ-flows</td>
<td>59</td>
</tr>
<tr>
<td>5.3</td>
<td>Threat to Validity</td>
<td>60</td>
</tr>
<tr>
<td>5.4</td>
<td>Conclusion</td>
<td>60</td>
</tr>
<tr>
<td>6</td>
<td>Contextualizing the PQ-flows</td>
<td>63</td>
</tr>
<tr>
<td>6.1</td>
<td>Introduction</td>
<td>63</td>
</tr>
<tr>
<td>6.2</td>
<td>Contextualizing the PQ-flows</td>
<td>64</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Automated Detection of ASFRs</td>
<td>66</td>
</tr>
<tr>
<td>6.2.1.1</td>
<td>Preliminary Experiment</td>
<td>66</td>
</tr>
<tr>
<td>6.2.1.2</td>
<td>Training on a Larger Set of ASFRs</td>
<td>69</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Recommending PQ-flows Relevant for an SRS Document</td>
<td>73</td>
</tr>
<tr>
<td>6.2.3</td>
<td>Annotating the PQ-flows</td>
<td>74</td>
</tr>
<tr>
<td>6.2.3.1</td>
<td>Analysis of Results for Annotating PQ-flows</td>
<td>76</td>
</tr>
<tr>
<td>6.3</td>
<td>Evaluating the PQ-Flows</td>
<td>77</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Feedback from the BAs</td>
<td>78</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Feedback from the SAs</td>
<td>79</td>
</tr>
<tr>
<td>6.4</td>
<td>Threats to Validity</td>
<td>80</td>
</tr>
<tr>
<td>6.5</td>
<td>Conclusion</td>
<td>80</td>
</tr>
<tr>
<td>7</td>
<td>Empirical Evaluation of the Approach</td>
<td>83</td>
</tr>
<tr>
<td>7.1</td>
<td>Introduction</td>
<td>83</td>
</tr>
<tr>
<td>7.2</td>
<td>Research Questions</td>
<td>84</td>
</tr>
<tr>
<td>7.3</td>
<td>Research Methodology and Research Plan</td>
<td>84</td>
</tr>
<tr>
<td>7.3.1</td>
<td>Study 1</td>
<td>84</td>
</tr>
<tr>
<td>7.3.1.1</td>
<td>Research Methodology</td>
<td>84</td>
</tr>
<tr>
<td>7.3.1.2</td>
<td>Threats to Validity</td>
<td>85</td>
</tr>
<tr>
<td>7.3.1.3</td>
<td>Results</td>
<td>86</td>
</tr>
</tbody>
</table>
8 Knowledge Base for Managing ASFRs and PQ-flows ................................................. 91

8.1 Introduction ............................................................................................................. 91

8.2 Knowledge-assisted Ontology-based Requirements Evolution (K-RE) Framework ............................................................................................................................. 91

8.2.1 Foundations of K-RE ....................................................................................... 92

8.3 Extending K-RE to accommodate the Knowledge of ASFRs and PQ-flows ......................................................................................................................... 94

8.4 Sample Screenshots of the Augmented K-RE Framework .................................. 96

8.5 Validation Study ..................................................................................................... 97

8.6 Conclusion .............................................................................................................. 99

9 Conclusion ................................................................................................................ 101

9.1 Revisiting the Research Questions ....................................................................... 101

9.1.1 How do SAs currently identify architecturally significant information from a SRS? (RQ 1) ................................................................. 101

9.1.2 How to use the mechanism to equip BAs to ask more architecturally relevant information? How could we design an approach that integrates the mechanism in an organization’s software delivery process? (RQ 2) ................................................................. 102

9.1.3 Is the proposed way of working usable, useful and generalizable in practice? (RQ 3) ................................................................. 102

9.2 Contributions ....................................................................................................... 103

9.3 Directions for Future Work .................................................................................... 104

Bibliography .................................................................................................................. 105
Acknowledgement

“The longer you have to wait for something, the more you will appreciate it when it finally arrives. The harder you have to fight for something, the more priceless it will become once you achieve it. And the more pain you have to endure on your journey, the sweeter the arrival at your destination. All good things are worth waiting for and worth fighting for.”

- Susan Gale

Five years back when I enrolled for this PhD program as an external student with a full time job and a very demanding home front, I knew I chose a hard path. However, I also knew that a supporting and loving family, great colleagues at work, understanding supervisors, advisors and collaborators would lead me through this path and provide me with the required motivation and support to arrive at the destination.

Here I would like to express my heartfelt gratitude to everyone who turned my last five years into a great learning experience and also made me realize that nothing is impossible if you have the willpower and the right support system.

Roel, thank you for being such a wonderful mentor. You helped me understand the big picture when I was lost in the details. You taught me how to be clear and precise while proposing ideas. There is so much that I have learnt from you in every interaction that I had with you. I am grateful for your constant supervision, mentoring and encouragement. Thank you for making me realize the joy of research.

Maya, no words can express how much I am indebted to you. This dissertation would not have been possible without your support, encouragement and involvement from the beginning of this research work till date. Despite the fact that the last few years had been very challenging for you, you did not let it impact my research work. You were just an email away whenever I needed you as a mentor, a supervisor or as a friend. I consider it as a blessing that my relationship with you was much more than a formal one. I enjoyed the numerous discussions we had about life, family, kids, festivals, cultures and people.

A part time PhD with a full time job is an insurmountable challenge if you do not have an understanding and accommodating boss. I was fortunate to have such a boss. Smita, I am deeply appreciative of all the support that you provided me during this journey. During our calls with the higher management, you always pitched in to highlight that I am distributing my time between the work assigned at TCS and my PhD work. Because of you as my boss, I am able to manage my professional and personal life so well. I cannot thank you enough for the invaluable role that you have played in this journey.

Jane, thank you for all your valuable inputs, contributions, constructive criticism, and positive appreciation throughout the course of this research work which led to its successful completion. Looking at your working style, I learnt what it means to be a researcher in true
spirit. Your passion, conviction and zeal for perfection has always inspired me to do more and get better at my work. I admire you and I hope I have imbibed some of your qualities as a result of the interactions I had with you.

I am thankful to my colleagues Abhishek and Balaji for participating in the machine learning part of this research work. Your contributions are highly valued. Abhishek, thanks for proofreading certain parts of my dissertation. I appreciate our time spent on the numerous discussions we had about work and everything else under the sky. I am truly blessed to have you as my colleague. Thanks Vivek and Amar for the help in formatting the final thesis draft. I am also thankful for the support of the bachelor’s students Lehanshu and Nikhil who contributed to parts of this research.

The research reported in this dissertation is the result of the numerous case studies that I conducted with practitioners. I would like to thank each and every study participant for the time and effort they have put in shaping this research. Without your inputs, this dissertation was not possible.

I would like to express my sincere gratitude to my employer TCS for providing me with all the required support in terms of the infrastructure to carry out this research work and sponsorship provided for all my conference travels.

I am more than grateful to all the members of my defense committee. I learnt a lot from your comments and I strongly believe that my dissertation has improved significantly as a result of addressing your comments.

Many thanks to Klaas for translating the English summary of the dissertation into the Dutch language. Thank you Lucy for taking out time to verify the Dutch summary for its compliance with the official Dutch spelling and grammar.

Suse and Jaeanette, I truly appreciate all your help and support. Thank you for patiently responding to all my queries.

Thank you Wasim and Prince, the only fellow PhD mates I got to interact with during my very short visits to Twente. Wasim, special thanks for driving me from Twente to Essen and back for REFSQ 2017. Thanks for patiently accompanying me for my shopping trips even though you hate shopping.

I owe a big thanks to my wonderful husband, Anish, for his continued and unfailing love, unwavering support, understanding and advices. You encouraged me when I needed motivation, criticized me when I was wrong, and comforted me when I felt stressed.

Special thanks to my wonderful kids Annjelin and Elvin, for being so adorable. Annjelin, thank you for being so well behaved and cooperative. You do so many things independently and make my life so much easier as a mother. Elvin, thank you for being such a darling. Your innocent smile and mischiefs keeps me going.
The biggest blessing for a working mother is to know that her children are in safe hands while she is at work. I would like to express my in-depth gratitude to my lovely parents, Varghese and Daisy, who enthusiastically volunteered to take care of my kids while I am at work. It is only because of both of you that I was able to work peacefully without worrying about my kids. Thank you for all your love and unconditional support. All my success and achievements in my profession, I owe it to you. I will always be indebted to God for giving me such wonderful parents.

Last but not least, I thank the Almighty God for giving me the strength and patience to work through all these years and making sure that I do justice to all the roles and responsibilities that he has bestowed on me.

Preethu Rose Anish

February 2020
Summary

Requirements engineering (RE) activities involve capturing both functional and non-functional requirements of the software system to be developed. The software requirements specifications (SRS) resulting from these activities often lack the details needed by software architects (SAs) to make informed architecture design decisions. In turn, if wrong architectural decisions are made, the intended but unstated requirements will not be satisfied. To cope with the lack of information, the SAs either make assumptions or go back to the business analysts (BAs) for clarifications or conduct additional stakeholder interviews. However, assumptions, if incorrect, can lead to costly and time-consuming refactoring efforts at later project stages, while additional unplanned interviews by BAs can slow down time-to-market.

Asking BAs to provide architecturally richer specification seems like a plausible solution to this problem, but is going to be ineffective, given that BAs lack the technical architectural knowledge needed to ask the kind of questions that extract architectural details from the client. This is a typical scenario in global software engineering and outsourcing projects where communication between BAs and SAs mostly takes place through a SRS and expertise between these two professional groups is not shared.

Considerable academic research literature exists acknowledging that requirements are never detailed enough to make informed architectural decisions. However, to the best of our knowledge at the time of writing this dissertation, there is no systematic work done to remedy this situation.

Motivated by the problem scenario detailed above, this dissertation introduces an approach that leverages the knowledge of experienced SAs and makes it available to BAs to equip them to elicit architecturally richer specification.

Using an empirical research method, we first discovered an initial set of 15 architecturally significant functional requirements (ASFR) categories. We then identified a set of architecturally relevant questions pertinent to each ASFR category. We called them Probing Questions (PQs). We found that the PQs for the various ASFR categories can be organized in a logical sequence. We referred to these as PQ-flows. Having realized the practical scenario that the ASFRs and PQ-flows can grow based on specific project and business domain needs and technology advancements, we further created a knowledge base (KB) with the initial set of ASFRs and the corresponding PQ-flows to let the SAs evolve the KB with ASFRs and PQ-flows that are relevant to their project settings. We went a step ahead to provide automated support for recommending relevant PQ-flows for an SRS and then generating annotated PQ-flows wherein the annotations on the PQs contain possibly relevant requirements from the SRS that already answers some of the PQs. The BAs can leverage this information to determine what is already known, what knowledge is missing, and what effect this would have on the flow of the PQs.
Samenvatting

Requirements Engineering omvat het vastleggen van functionele en niet-functionele eisen aan een te ontwikkelen softwaresysteem. Het daaruit volgende programma van eisen (Software Requirements Specification, SRS) heeft vaak niet genoeg details voor de softwarearchitect (SA) om goede ontwerpbeslissingen te kunnen nemen. Als verkeerde keuzes worden gemaakt, kan het gebeuren dat aan gewenste, maar niet expliciet gemaakte eisen niet is voldaan. Bij gebrek aan gedetailleerde informatie kan een SA zelf aannames maken of de businessanalist (BA) die de eisen opgesteld heeft, vragen om aanvullende informatie, eventueel te verkrijgen door middel van aanvullende interviews. Worden er foute aannames gemaakt, dan kan leiden tot kostbare en langdurige herstelwerkzaamheden in een later stadium; zijn aanvullende interviews door de BA nodig, dan kan de time-to-market daardoor worden vertraagd.

Het lijkt voor de hand te liggen om BA’s te vragen om meer details van de architectuur in de SRS op te nemen. Maar de verwachting is dat dat niet effectief is: BA’s hebben niet de technische kennis van systeemarchitectuur om de klant de juiste vragen te stellen om deze details boven water te halen. Dit scenario doet zich vaak voor in Global Software Engineering en in outsourcingprojecten, waar de communicatie tussen BA’s en SA’s voornamelijk via de SRS loopt en beide professionele groepen elkaars expertise niet delen.

Er is een brede onderzoeksliteratuur waaruit blijkt dat eisen nooit gedetailleerd genoeg zijn om goede architectuurbeslissingen te kunnen nemen. Maar voorzover wij weten is tot op heden geen systematisch onderzoek gedaan naar hoe dit probleem aangepakt kan worden. Deze dissertatie stelt een aanpak voor om de kennis van ervaren SA’s ter beschikking te stellen aan BA’s. Daarmee worden de BA’s in staat gesteld om meer details in een SRS op te nemen die relevant zijn voor de systeemarchitectuur.

Door middel van empirisch onderzoek hebben we een initiële collectie van 15 categorieën van Architecturally Significant Functional Requirements (ASFR’s) opgesteld. Vervolgens hebben we relevante vragen voor iedere categorie geïdentificeerd. We noemen dit Probing Questions (PQ’s). Voor de verschillende ASFR-categorieën bleken de PQ’s in een logische volgorde te kunnen worden geplaatst, aangeduid als PQ-flows. ASFR’s en PQ-flows kunnen verder uitgebouwd worden aan de hand van specifieke projecten, behoeften vanuit specifieke bedrijfstakken en verdere technologische ontwikkeling. Ten behoeve daarvan hebben we een kennisbank opgezet met een initiële collectie ASFR’s en PQ-flows, die door SA’s uitgebreid kan worden met ASFR’s en PQ-flows die relevant zijn voor hun specifieke projectomgeving. Als volgende stap hebben we automatische ondersteuning ontwikkeld voor het aanraden van relevante PQ-flows voor een SRS en voor het genereren van geannoteerde PQ-flows, waarin de annotaties verwijzen naar eisen uit de SRS die sommige PQ’s mogelijk al beantwoorden. De BA kan deze informatie gebruiken om vast te stellen wat al bekend is, welke kennis nog ontbreekt, en welk effect dit zou hebben op de flow van de PQ’s.
# List of Figures

1.1 DESIGN AND ENGINEERING CYCLE USED IN THIS RESEARCH (WIERINGA, 2014) ........... 8  
4.1 CONCEPTS AND THEIR RELATIONSHIP ........................................................................ 39  
5.1 PROCESS FOR CONSTRUCTION OF PQ-FLOWS .......................................................... 55  
5.2 PQ-FLOW FOR AUDIT TRAIL .................................................................................... 57  
5.3 PQ-FLOW FOR BUSINESS PROCESS 'STATE' ALERT .................................................. 57  
6.1 CONTEXTUALIZING THE PQ-FLOW - HIGH LEVEL....................................................... 64  
6.2 ANNOTATED PQ-FLOW FOR BATCH PROCESSING ..................................................... 75  
8.1 EXAMPLE INSTANCES OF THE K-RE ONTOLOGIES ................................................. 93  
8.2 AUGMENTED GENERIC REQUIREMENTS ONTOLOGY ................................................ 95  
8.3 PROCESS MAP TO SHOW HOW THE DIFFERENT ROLES WORK IN K-RE ................. 95  
8.4 SCREEN SHOT 1 – USER INTERFACE FOR MANAGING ASFR CATEGORIES ............ 96  
8.5 SCREEN SHOT 2 – USER INTERFACE FOR MANAGING PQ-FLOWS FOR THE ASFR CATEGORIES ............................................................................................................. 97
List of Tables

1.1 OVERVIEW OF RESEARCH METHODS ................................................................. 11
2.1 OVERVIEW OF RESEARCH METHODS AND MEASUREMENT METHODS USED .......... 16
4.1 PARTICIPANT DETAILS FOR THE PHASE I STUDY ........................................... 37
4.2 PARTICIPANT DETAILS FOR THE PHASE II STUDY ........................................... 38
4.3 QUANTITATIVE STUDY RESULTS ........................................................................ 50
5.1 DEMOGRAPHIC DETAILS OF THE RESPONDENTS .............................................. 56
5.2 PQ COUNTS ......................................................................................................... 59
6.1 CLASSIFICATION RESULTS FOR EACH ASFR CATEGORY ...................................... 68
6.2 CONFUSION MATRIX DEPICTING CLASSIFICATION RESULTS ............................... 69
6.3 REQUIREMENTS COUNT FOR EACH ASFR CATEGORY ........................................ 70
6.4 CLASSIFICATION RESULTS FOR EACH ASFR CATEGORY AT STATEMENT LEVEL .......... 72
6.5 COMPARISON OF CLASSIFICATION RESULTS FOR PRELIMINARY EXPERIMENT AND EXPERIMENT WITH LARGER DATASET ........................................................................ 73
6.6 ACCURACY BY WHICH RELEVANT PQ-FLOWS ARE IDENTIFIED ...................... 74
6.7 RESULTS OF ASSOCIATING REQUIREMENTS WITH PQ-FLOWS ......................... 76
7.1 DETAILS OF PARTICIPANTS FOR STUDY 1 ....................................................... 85
7.2 DETAILS OF PARTICIPANTS FOR STUDY 2 ....................................................... 88
## List of Appendixes

<table>
<thead>
<tr>
<th>Appendix</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>INTERVIEW QUESTIONS FOR THE STUDIES REPORTED IN CHAPTER 4</td>
<td>123</td>
</tr>
<tr>
<td>B</td>
<td>QUESTIONS OF THE SURVEY HOSTED IN SURVEY MONKEY</td>
<td>125</td>
</tr>
<tr>
<td>C</td>
<td>PQ-FLOW CATALOG FOR THE 10 ASFR CATEGORIES</td>
<td>129</td>
</tr>
<tr>
<td>D</td>
<td>QUESTIONNAIRE FOR BUSINESS ANALYST AND SOFTWARE ARCHITECT</td>
<td>137</td>
</tr>
<tr>
<td>E</td>
<td>QUESTIONNAIRE FOR EMPIRICALLY EVALUATING THE APPROACH</td>
<td>139</td>
</tr>
<tr>
<td>F</td>
<td>QUESTIONNAIRE FOR VALIDATING THE KNOWLEDGE BASE</td>
<td>141</td>
</tr>
<tr>
<td>G</td>
<td>LIST OF CATEGORIES OBTAINED FROM INTERVIEW DATA ANALYSIS</td>
<td>143</td>
</tr>
</tbody>
</table>
List of Abbreviations

ACM  Association of Computing Machinery
AKM  Architectural Knowledge Management
ANN  Artificial Neural Network
ASFR  Architecturally Significant Functional Requirement
ASNFR  Architecturally Significant Non-functional Requirement
ASR  Architecturally Significant Requirements
BA  Business Analyst
BERT  Bidirectional Encoder Representations from Transformers
CBAM  Cost Benefit Analysis Method
CBSP  Component-Bus-System-Property
CNN  Convolutional Neural Network
COTS  Commercially Off the Shelf
CS  Computer Science
ERP  Enterprise Resource Planning
FR  Functional Requirement
GT  Grounded Theory
IEEE  The Institute of Electrical and Electronics Engineers
ISO  International Organization for Standardization
IT  Information Technology
KB  Knowledge Base
KM  Knowledge Management
K-RE  Knowledge-assisted Requirements Evolution
LP  Label Powerset
LSTM  Long Short-Term Memory
NB  Naive Bayes
NFR  Non-Functional Requirement
PQ  Probing Question
QR code  Quick Response Code
RAkEL  RAndom k labELsets classifier
RE  Requirements Engineering
RKE  Requirements Knowledge and Experience
RNN  Recurrent Neural Network
RQ  Research Question
SA  Software Architect
SE  Software Engineering
SME  Subject Matter Expert
SRS  Software Requirement Specification
SWEBoK  Software Engineering Body of Knowledge
SWRL  Semantic Web Rule Language
TCS  Tata Consultancy Services
UML  Unified Modeling Language
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>UT</td>
<td>University of Twente</td>
</tr>
<tr>
<td>VSM</td>
<td>Vector Space Model</td>
</tr>
<tr>
<td>WEKA</td>
<td>Waikato Environment for Knowledge Analysis</td>
</tr>
</tbody>
</table>
Introduction

This chapter provides an overview of the research published in this dissertation. It presents the background and motivation for the problem under investigation, presents definitions of the concepts used in this research, the research scope, its goals and the research questions. The chapter further briefly introduces the research methodology of the PhD project and states the contributions to knowledge of the presented work and the implications for practice. The chapter concludes with an outline of the dissertation.

1.1 Research Problem and Motivation

Requirements engineering (RE) activities often involve capturing both functional and non-functional requirements, describing respectively what the system needs to do, and what the system needs to be (e.g. secure, safe, maintainable, available). In a typical project, software architects design an architecture that attempts to meet these requirements. While RE and software architecture textbooks [1, 5] indicate this to be a smooth process, experiences from real-life projects in the IT industry show that in large-scale contexts [2, 3], engineering functional and non-functional requirements is often disjoint, and is also disjoint with the architectural design activities. RE and architectural design activities are executed by different teams at different points of time. This poses various problems to large-scale projects in which the synchronization of requirements documents is error-prone, cannot be accomplished by one person, and where reliance on tacit knowledge is not a viable option [6, 7].

Software architects (SAs) are responsible for designing an architectural solution that satisfies the functional and non-functional requirements of the system to the fullest extent possible. However, software requirement specifications (SRS) often lack the details needed by SAs to make informed architectural decisions. For example, a SRS might include a high level ‘Auditing’ requirement, but fail to specify whether the audit is for internal or regulatory compliance, what data must be produced for compliance, or if notification messages must be sent to stakeholders when certain audit events occur. Such details impact architectural decisions and should be exposed early in the requirements elicitation and analysis process. If wrong architectural decisions are made in the absence of such detail, the intended but unstated functional requirements will not be satisfied. In practice, clients often assume that developers implicitly understand their needs and therefore do not fully express their requirements [8]. Experienced SAs intuitively realize when critical information is missing. To compensate, they may make assumptions based on their own knowledge of the problem or conduct additional formal or informal interviews with the stakeholders to seek clarification.
and to elicit the missing information. However, assumptions, if incorrect, can lead to costly and time-consuming refactoring efforts at later stages of projects [162]; while additional unplanned interviews can slow down time-to-market.

Typically, functional requirements (FRs) are collected and documented by business analysts (BAs) and domain experts [1]. Non-functional requirements (NFRs1), on the other hand, are often dealt with by SAs and technology experts [2, 3]. The knowledge corpora for FRs and architectural solutions are often maintained separately [9]. This represents a problem to both SAs and BAs. To SAs, uncovering requirements that may have a critical impact on architectural decisions becomes hard and as a result they struggle to make informed design decisions. To BAs, in the absence of required technical expertise, they have no way to predict what details would be critical and therefore should be in the SRS for good architectural decision making.

Currently, SAs use ad-hoc and person-specific coping strategies. For example, asking BAs to provide richer, more informative specifications. This may seem like a good idea, but it is unlikely to be effective, given that BAs typically lack technical architectural knowledge. BAs are often seen to be ill-equipped to ask the right kind of questions needed to unearth architecturally significant information. This may result in incorrect requirements and project delays.

Motivated by the problem scenario detailed above, the primary goal of this dissertation is to leverage the knowledge of experienced SAs and to make it available to BAs so that they are equipped to elicit a more complete set of requirements that feed sufficient information into the architecture design process. While Software Engineering (SE) scholars agree that requirements elicitation and architectural design should be performed iteratively [10, 11, 12], little is known about what specifically BAs could or should do, in order to jump-start the process by identifying clients’ requirements in a way that helps SAs more effectively evaluate candidate architectural solutions. This dissertation attempts to provide a solution approach to this problem.

1.2 Definition of Terms

As of today, there is no widely accepted standard terminology among scholars and practitioners’ professional associations in the field of RE and software architecture. Depending on the publication forum and the author preferences, different terms or phrases are used to refer to a particular concept. As can be expected, the same concept can also have different meanings and/or interpretations in works by different authors. In this section, we therefore explicitly state our interpretation of each of the term we use in this dissertation. We consulted definitions from different sources (such as IREB [13] and 20), and for the purpose of this dissertation, we use the following working definitions:

**Functional requirement (FR)** – A functional requirement is a desired behaviour triggered by some event or condition change, and delivering some desired output of the system. For

---

1 As there is no commonly agreed umbrella term for such requirements, we have opted to use the traditional term of NFR, which is broadly understood in both industry and academia

---
example, the requirement *the system must record every modification to client records for audit purposes* is an example of a functional requirement.

**Non-Functional Requirement (NFR)** – Any other desired property of the system that is not functional is called a non-functional requirement. For example, the requirements pertinent to availability, accessibility, performance, reliability, maintainability, serviceability, scalability are examples of non-functional requirements.

**Architecturally Significant Requirements (ASR)** - Architecturally significant requirements are those requirements that have a measurable impact on the architecture of the software system [4]. Both functional as well as non-functional requirements can fall in the category of architecturally significant requirements. For example, the requirements *The downtime for the system shall not exceed 1 hour per 1000 hours of operations* (which is a non-functional requirement) and *An audit trail of all the critical transactions should be maintained* (which is a functional requirement) are both examples of architecturally significant requirements.

Second, we define the roles of Business Analyst (BA) and Software Architect (SA) in the context of large global outsourcing projects. We focus on this context because it is of relevance for this PhD project and the organization where the author of this dissertation is employed:

**Business Analysts (BA)** – A BA is a role involved in understanding the business or functional aspects of requirements for IT projects or a commercially off the Shelf (COTS) implementation. The understanding would result in developing detailed functional specifications or gap analysis or review of requirements artefacts or review of developed solutions from a functional completeness perspective.

**Software Architect (SA)** – A SA is a role involved in converting the business requirements captured by the BA into architecture and design (which then forms the blueprint of the project), organizing the development effort, create a vision and consider the project parameters end-to-end to attain the vision. This role could change in terms of significance based on project context. For example, in Tata Consultancy Services (TCS) - an Indian multinational information technology (IT) service, consulting and business solutions company [105], in case of COTS implementation, the SA has a bigger role. The SA has to constantly evolve the product hand in hand with the marketing and research group. The SA’s role is to help keep the product at the sharper end of the pyramid without having to break support for older versions.

Third, we define the new concepts that we have introduced as a part of our solution approach.

**Architecturally Significant Functional Requirements (ASFR)** – Any functional requirement that has a ‘significant’ impact on the architecture of the software system is called architecturally significant functional requirement (ASFR). While almost all functional requirements may be argued to have some degree of impact on software architecture; our focus is on those functional requirements that have a significant impact. By this, we mean any functional requirement that:

1. has a potentially broad impact across the system,
2. is high-risk, possibly volatile, and
3. if modified could involve expensive refactoring.

For example, the requirement: *an audit trail of all the critical transactions shall be maintained* is an ASFR.

**Architecturally Significant Non-functional Requirements (ASNFR)** – Any non-functional requirement that has a significant architectural impact is an architecturally significant non-functional requirements. The *downtime for the system shall not exceed 1 hour per 1000 hours of operations* is an example of architecturally significant non-functional requirement.

**Architecturally Significant Functional Requirements Categories (ASFR Categories):** The various categories into which ASFRs can be categorized based on the kind of impact they have on the architecture is referred to as ASFR category. *Audit Trail, Batch Processing, Localization, Payment, Print* are few examples of ASFR categories.

**Probing Questions (PQs)** - The questions asked to extract architectural information are called Probing Questions (PQs). For example, for an Audit Trail requirement, questions such as “*Is the audit for internal compliance or for regulatory compliance, Do you want to build reports over the audit*” are examples of PQs.

**Probing Question Flows (PQ-flows)** – The probing questions when logically sequenced in dialogs is referred to as Probing Question Flows (PQ-flows).

**Architectural Details** – The details pertinent to architectural requirements which are needed to make informed architectural decisions are called architectural details. For example, for an architectural requirement “*An audit of all the transactions shall be maintained*, architectural details would include details such as *whether the audit is needed for regulatory compliance or internal compliance, what data must be produced for compliance, if notification messages must be sent to stakeholders when certain audit events occur*” and so on.

**Architectural Knowledge Base** – Architectural knowledge base is a knowledge base (KB) of architectural knowledge organized in the form of ASFR categories and their corresponding PQs and PQ-flows.

**Architectural Knowledge Creator** – Architectural knowledge creators are SAs (essentially junior and mid-level) who contribute architectural knowledge in the form of ASFR categories and their corresponding PQs and PQ-flows to the architectural KB.

**Architectural Knowledge Curator** – Architectural knowledge curator is a senior SA whose role is to maintain the correctness and consistency of the architectural knowledge in the architectural KB. Any contributions to the KB would be reviewed by the architectural knowledge curator before it is made available for use in projects.

**Architectural Knowledge User** – Architectural knowledge users are the BAs who would make use of the architectural knowledge in the architectural KB during requirements gathering. Using such a KB, they can produce architecturally richer specifications that feeds sufficient information into the architectural design process.
1.3 Scope of the Research

Requirements engineering activities involve the capture of both FRs and NFRs, respectively describing what the system needs to do and how it is to be achieved. FRs are collected and documented by BAs and domain experts. Architectural decisions, on the other hand, are made by technology experts. The knowledge corpora for FRs and architectural solutions are therefore maintained separately. Almost all FRs may be argued to have some extent of impact on architecture. However, our focus is only on ASFRs i.e. those FRs that have a significant impact on architecture. For instance, the FR statement Ability to receive notification whenever a transaction is made carry an architectural impact. However, the information that determines the kind of architectural impact (e.g. email notification or real time notification with an ability to subscribe to topics of interest) is typically not explicitly stated in the FR statement. This is an important information, which if not explicitly stated could lead to incorrect architectural decisions. For example, in this case, depending upon the kind of notification, the architecture would change. For simple email, an event-driven architecture would suffice; as opposed to the requirement for real time notification with the ability to subscribe to different topics, for which publish-subscribe – a different architectural style is required [14].

ASFRs can be difficult to identify from a given set of FRs. The BAs often do not have the requisite technical knowledge to infer and articulate the architectural impact from the FRs that they capture from clients. As a consequence, ASFRs carry hidden architectural implications that need to be unearthed. This scenario is quite different from NFRs that explicitly solicit architectural considerations. In fact, NFRs are considered the main drivers for architectural decisions [15, 16, and 17]. For example, a NFR such as “The downtime for the system shall not exceed 1 hour per 1000 hours of operations” is clearly an availability requirement with an explicit architectural impact. In the RE discipline, much of the published literature [17, 18, 19] on architecturally significant requirements has focused on NFRs related to security, reliability, performance, and other such attributes. Criteria to identify ASFRs from a set of FRs, and mechanisms to decipher their exact impact on software architecture have not been studied so far.

In the RE and Software Architecture literature, it is well understood that both FRs and NFRs can have an equally important impact on architectural design. However, with FRs, we observed the following:

1. the architectural impact of FRs is often implicit unlike that of NFRs, which explicitly solicit architectural considerations and

2. Much of the published literature [for example: 19, 20, 17, 21, 22, 23] on architecturally significant requirements has focused on NFRs. Far less emphasis has been placed on addressing the architectural significance of FRs [10].

Nevertheless, we clearly acknowledge that the problem is equally relevant to both FRs and NFRs. However, to scope this dissertation, we have decided to focus on ASFRs.
Next, our focus is on project settings where the BA and SA are distinct roles with distinct tasks and responsibilities. This setting is typically observed in systems level projects and outsourced environments.

1.4 Research Objective

Our objective is to enhance communication between the two roles of BA and SA by introducing a knowledge base (KB) with architectural knowledge to be used by BAs during requirements gathering. The solution idea is to provide BAs with a KB of ASFR categories and PQ-flows per category to elicit architectural details, plus a tool-supported method that allows SAs to extend the KB with relevant ASFRs. It is worth noting that exhaustive list of requirements and architecture decisions exist in Enterprise Resource Planning (ERP) systems but not for the bespoke systems that we focus on. For our kind of systems, we need a dynamic KB which has a mechanism of evolving itself based on specific project needs. As mentioned in section 1.3 (scope of research), we acknowledge that both ASFRs and ASNFRs are equally important in this context. However, to scope this PhD research, we focus on ASFRs.

We aim to provide a solution approach that is validated on the following grounds:

1. the perceived ease of use, effectiveness and relevance;
2. the generalizability;

We define each of these validation criteria as follows:

**Ease of Use:** By referring to the ‘Ease of use’ concept originally published by FD Davis [28], we measure Ease of use by gauging how easy it is for the BAs to use the approach as a part of requirements gathering, do they find it easy to adapt to this new way or do they consider this as a paradigm shift that they are not able to relate to.

**Effectiveness:** By Effectiveness of PQs, we intend to investigate the degree to which the PQs are successful in producing a desired result i.e. assist the BAs in unearthing architectural information from the client during requirements gathering.

**Relevance:** By Relevance we intend to investigate whether the BAs find the approach important to their requirement gathering practices and would add value to it.

Furthermore, regarding examination of generalizability, we include the perspectives of both BAs and SAs. Our objective is to test generalizability to similar contexts as those in which we validate our approach, of two things:

1. of the method to fill the KB, and
2. of the ASFRs and PQ-flows in the KB.

1.5 Research Questions

To discuss our research questions, we use the distinction between design research questions and knowledge research questions. According to Wieringa [29, 30, 31] and Heerkens [31], design research questions ask for a way to achieve a desired output from a given input. They
relate to a situation where some change needs to be enacted according to the way we think the world should be. This type of questions focus mainly on ‘how-to-do’ questions. The second type of research questions is the knowledge question. Knowledge research questions emerge when there is a difference between what we know about the world and what we would like to know about the world. Knowledge questions are either empirical questions or conceptual questions that would be answered by research. Within the context of these questions, we need to study the world to obtain knowledge related to a particular aspect.

The main goal of this research is to define an approach that assists in leveraging the knowledge of experienced SAs and to make it available to BAs so that they are equipped to elicit a more complete set of requirements that feeds sufficient information into the architectural design process. Based on this goal, the central research question (RQ) of this dissertation is:

**How to make the knowledge of experienced SAs available to BAs during requirements elicitation?**

To meet our research objective, the central RQ is elaborated in the following sub-questions:

**RQ 1:** How do SAs currently identify architecturally significant information from a SRS?

- **RQ 1.1:** What are Architecturally Significant Functional Requirements (ASFRs) and what is currently known about them?
- **RQ 1.2:** What categories of ASFRs have implicit architectural impact for SAs?
- **RQ 1.3:** From a SA’s perspective, what mechanism is used to unearth the unspecified architectural details in a SRS?
- **RQ 1.4:** Having found answer to RQ 1.3, what kind of information is discoverable using the mechanism (identified in RQ 1.3)?

**RQ 2:** How to use the mechanism to equip BAs to ask more architecturally relevant information? How could we design an approach that integrates the mechanism in an organization’s software delivery process?

**RQ 3:** To what extent is the proposed way of working usable, useful and generalizable in practice?

- **RQ 3.1:** Can practicing BAs use the approach?
- **RQ 3.2:** Does the proposed approach take less effort overall than the current way of working?
- **RQ 3.3:** Does it lead to architecture designs that are comparable in quality to the current way of working?
- **RQ 3.4:** Is it generalizable to systems other that business information systems?

The validation criteria mentioned in section 1.4 on page 6 are **ease of use, effectiveness, relevance** and **generalizability**. RQ 3.1, RQ 3.2 and RQ 3.3 relate the concepts of usability and
usefulness to the validation criteria of ease of use, effectiveness and relevance. RQ 3.4 relates to generalizability.

1.6 Research Methodology at a Glance

This dissertation adopts a qualitative interview and survey-based research process for the purpose of problem investigation, treatment design and treatment validation. To materialize this, Design Science Methodology for Information Systems and Software Engineering proposed by Wieringa [29] is selected. Figure 1.1 depicts the design and engineering cycles adopted by this research. Real-world implementation in the sense of technology transfer is not part of this research project. The prototype that is tested with real-world BAs and SAs forms part of treatment validation.

As Figure 1.1 indicates, using the design and engineering cycle involves answering a number of questions. Regarding the problem investigation questions (please see the right upper area on Figure 1.1), we note that in our problem definition described in the previous sections, we have already sketched the answers to those questions. We identified stakeholders and their goals, indicated what the problematic phenomenon is, why it exist (mechanisms that produce it) and what effects this has.

Below, we present the steps of our research process that is inspired by the Design Science methodology [29]:

**Step 1:** This step summarizes what is known about the problem, including the conceptual framework (definitions of ASFRs), and also the available treatments, according to previously published empirical research. This step provides answers to RQ 1. We chose Literature Review as it is one of the most important methods used to support evidence-based research [40] for purposes as ours. In our research, the first activity therefore was literature study aimed to capture the state-of-the-art in ASRs, what is already known about the problem, what already exists in terms of treatment proposals. A systematic literature review follows a pre-defined and structured rigorous review protocol and therefore is believed to be the most widely used.
literature review method [41]. Since we relaxed some of the guidelines proposed by Kitchenham et. al. [41], we believe that our literature review qualifies as a scoping review [177, 178]. In this dissertation, we have used this method to study the state-of-art practice in ASRs. Our review process was inspired by the guidelines proposed by Kitchenham et. al. [41] to identify, analyze and synthesize relevant work related to our research. The systematic review process proposed by Kitchenham et. al. [41] comprises of three phases namely (1) planning the review, (2) conducting the review, and (3) reporting the review. We did cover all these three phases but in our research context we streamlined the phases in terms of breadth and depth of search and screening processes, quality assessment and synthesis (as suggested in [177, 178]. For instance, we did not maintain any detailed data extraction forms and we did not tabulate the results of data synthesis in the manner described by Kitchenham et. al. [41].

Step 2: In RQ 1, we investigated the problem and the available treatments. In step 1 we use literature review to find answer to RQ 1. In this step, we further use quantitative semi-structured interviews to supplement the literature review in step 1. In other words, in step 2, we find answer to the same research question i.e. RQ 1 by utilizing a different research method. To answer RQ 1, a qualitative semi-structured interview based technique which is the most widely used data gathering technique in qualitative research [42] was chosen. The technique was designed by following R. Yin’s guideline for exploratory case study design [24]. As part of our research on RQ 1, we executed a qualitative case study with 14 practicing SAs from India, the United States of America and the Netherlands. For the qualitative analysis of the collected data, the guidelines of Charmaz’ Constructive Grounded Theory building method [32] was deployed. This approach is used in social sciences to construct general propositions (called a ‘theory’ in this approach) from textual data. This study is published in [33, 34]. The key findings of this study are: (1) SRSs often lack crucial architecturally relevant information needed to make informed architectural decisions, (2) PQs are an essential mechanism for unearthing underspecified architecturally relevant information, (3) identification of a total of 15 categories of ASFRs, each with its own set of PQs, (4) and there is a logical way to sequence the PQs during the requirements elicitation process. This completes our survey of available treatments. In step 2 we have analyzed these treatments and have started designing our approach consisting of ASFR categories and PQs. In the next step, we move on to answering the design problem RQ 2.

Step 3: Having identified that the PQs have a logical sequence, we headed towards finding answer to our design problem stated in RQ 2. For this step, we followed a design science approach [29] which was used to discover and document the PQ-flow structures for the identified ASFR categories. We started with a ‘basic’ design of the PQ-flows and then integrated practitioners’ feedback into the design. This resulted in a refined version of the PQ-flows for which further feedback was elicited. The process included the following steps: (a) interview SAs, (b) analyze the PQs obtained through the interviews and create the initial basic PQ-flows in consultation with a SA, (c) conduct an online survey to collect feedback on the early PQ-flows that were created in step (b), refine the PQ-flows in the light of the feedback received, and (d) conduct a follow-up online survey to evaluate the refined PQ-flows.
from step (c). Based on the analysis of the survey findings, PQ-flows for the 15 ASFR categories were created. This study is published in [35].

**Step 4:** To find a more complete answer to RQ 2, we attempted further design by including an automated support for recommending relevant PQ-flows for an SRS and then generating annotated PQ-flows wherein the annotations on the PQs contain possibly relevant requirements from the SRS that already answers some of the PQs. The BAs can leverage this information to determine what is already known, what knowledge is missing, and what effect this would have on the flow of the PQs. For automating this recommendation, we evaluated existing machine learning techniques on our dataset to find if it is feasible to automate the recommendation and annotation of PQ-flows. Initial results on this are reported in [35]. We note that design of machine learning techniques is out of scope for this research. This dissertation reports on use of some existing machine learning techniques (e.g. Multinomial Naïve Bayes, Vector Space Model) that fit our research purpose.

**Step 5:** In this step, we further probe on RQ 2, to extend our design to include a KB comprising of the seed knowledge of the 15 ASFR categories identified in step 3, and the corresponding PQ-flows identified in step 4. This KB is amenable to evolution by contributions from SAs. The SA can contribute new ASFRs, PQ and PQ-flows and / or modify the existing ones based on their specific project and/or business domain needs.

**Step 6:** Drawing on methodological sources in empirical software engineering [24, 25, 26, and 27], the empirical validation of our approach is an important step in this PhD research. This is to answer RQ 3. Therefore, this step focuses on treatment validation wherein our goal is to validate and evaluate the utility and usefulness of the approach. For this, we have used opinion-based and experimental validation. In opinion-based research, we have used focus groups and interviews to collect opinions of practicing BAs and SAs on the usability and utility of the approach. This will be used to understand the scope for further improving the approach. In experimental research, we have tested our approach by asking BAs and SAs to perform requirements specification and architecture design tasks using our approach in a context that is as realistic as possible, and compare the results with the current way of working. The most realistic context is where BAs and SAs use our approach in a real project. This would be a Technical Action Research [29]. However, as this is not feasible, we have simulated the real world by asking the subjects to use our approach to perform tasks in a realistic but simulated context. This forms our mechanism experiment.

Table 1.1 presents an overview of the research strategies we adopted in each chapter of this dissertation.
1.7 Contributions

Our study brings out how SAs cope with the process of inferring hidden meaning from ASFRs that are buried in the SRS received from BAs. Our empirical study resulted in the discovery of an initial list of 15 categories of ASFRs. We found that the PQs for the various ASFR categories can be organized in a logical sequence. We referred to it as PQ-flows; that could be used by BAs to elicit architecturally relevant information from the client during requirements gathering and produce architecturally richer specifications. By doing so, we found an inexpensive down-to-earth solution that seems effective for enriching SRs with architecturally relevant information. Using this approach, BAs can produce richer specifications that contain the details that SAs need for making architectural decisions. This could lead to less communication cycles between SAs and clients as there would be little need for follow-up clarifications in later project stages concerning ASFRs.

Having realized the practical scenario that the ASFRs and PQ-flows can grow based on specific project and business domain needs, we further created a KB of the ASFRs and the corresponding PQ-flows to let the SAs evolve the KB with relevant ASFRs and PQ-flows. We went a step ahead to provide automated support for recommending relevant PQ-flows for an SRS and then generating annotated PQ-flows wherein the annotations on the PQs contain possibly relevant requirements from the SRS that already answers some of the PQs. The BAs can leverage this information to determine what is already known, what knowledge is missing, and what effect this would have on the flow of the PQs.

As is evident from the discussion above, our claims are general. However, we have only explored information systems to validate our claims and our focus has mainly been on Insurance domain.

1.8 Implications

Our work has implications for research, practice as well as teaching.

Implications for Research: We noticed that architects’ PQs were grounded on their domain knowledge. An interesting line of research is the automation of the identification and classification of ASFRs and recommending relevant PQs to the BAs and architectural solutions.
to the SAs. Drawing on existing work in the area of machine learning [for example, 36 and 37], we have done some preliminary experiments in this direction (as reported in chapter 6). Our preliminary experiments with existing machine learning techniques establish that such a technique is amenable to automation. The initial results are encouraging and motivates future research in this direction. Moreover, the domain-dependent PQs can be linked to augment existing knowledge repositories in organizations [38]. Further, it would be interesting to find out whether agile approaches would be comparatively more or less effective in revealing hidden ASFRs.

**Implications for Practice:** From a practitioners’ perspective, knowing which ASFR categories have architectural impact can help BAs to elicit a more complete set of requirements. This provides the information that SAs need to make informed architectural decisions and can potentially reduce wasted effort caused by the need to rework the solution later in the project. Further a KB with the seed ASFRs and PQs can be used by any organization for their requirements gathering exercise. They can evolve this KB based on their specific project and/or domain needs.

**Implication for teaching:** It has been the observation of the author of this dissertation that RE textbooks often overlook the important synergies between RE and other SE activities such as software architecture. As more attention is paid to the RE-software architecture relationship, it is worthwhile to revise RE teaching by including concepts that help RE professionals produce requirements documents that actually help SAs come up with suitable architecture that better fits the client’s business. In particular, our study suggests that complementing RE training with education on business rules (Chapter 4, Section 4.4.1, page 45) might be a worthwhile option.

### 1.9 Publications

The majority of the research presented in this dissertation has been published previously. The chapters are based on the following publications [33, 34, 35, 39 and 75]:


- **P.R. Anish, B. Balasubramaniam, A. Sainani, J. Cleland-Huang, R. Wieringa, M. Daneva, S. Ghaisas, Probing for Requirements Knowledge to Stimulate Architectural Thinking.** In proceedings of the 38th International Conference on Software Engineering (ICSE 2016), pp. 843-854. ACM.
P.R. Anish, Towards an Approach to Stimulate Architectural Thinking during Requirements Engineering Phase, Doctoral Symposium, 24th IEEE Requirements Engineering Conference, RE 2016, Beijing, China


1.10 Outline of the Dissertation

This dissertation is largely based on the papers that have been previously published as a part of this research work. It is however, not organized as a collection of papers with a summary. Rather the text from the different papers have been incorporated in different chapters throughout the dissertation. This was done with the goal to logically and systematically sequence and group the various chapters.

In this dissertation, each chapter begins with a short introduction and a reference (if any) to the paper (s) on which the chapter is based. The chapters are:

Chapter 1 – Introduction: This chapter provides an introduction to this dissertation by presenting the motivation, the definition of terms, the scope of research, the research objective, questions and the methodology used, the contributions to research, practice and teaching and finally the list of publications based on which this dissertation is written.

Chapter 2 – Research Methodology: This chapter details the research methodology we adopted to develop and validate our approach. It would further detail the data gathering and the data analysis technique adopted by us to carry out this research.

Chapter 3 – Related Work: This chapter details the literature study we did to compare and contrast our work with respect to existing work done in this area.

Chapter 4 – Understanding ASFRs: This chapter details the interview based study we conducted with 14 SAs in three countries to identify various categories of ASFRs. This chapter is based on paper [34].

Chapter 5 – Creating Probing Question Flows: This chapter details our study with over 40 experienced SAs to identify reusable PQs for various ASFR categories and to organize them into structured flows. This chapter is based on part of the paper [35].

Chapter 6 – Contextualizing the PQ-flows: In this chapter, we detail our preliminary experiments with existing machine learning techniques to determine whether it is feasible to automate the contextualization of PQ-flows i.e. to determine when a PQ-flow is appropriate for use in a project, and to annotate individual PQs with relevant information extracted from the existing SRS. This chapter is based on part of the paper [35].
Chapter 7: Empirical Evaluation of the Approach: This chapter presents the experimental studies we carried out to validate our approach. This chapter is partially based on the paper [39].

Chapter 8 - Knowledge Base for Managing ASFRs and PQ-flows: This chapter presents the prototype for a KB that we have created to manage the ASFRs and PQ-flows. The KB would facilitate the updating of ASFRs and PQ-flows based on specific project context. The chapter also includes validation of the KB.

Chapter 9: Conclusion: This chapter reflects on the work presented in this dissertation. It discusses the lessons learnt and comments on the reusability of the proposed approach in other application domains and project settings. It further outlines potential future research directions for this work.
Research Methodology

This chapter presents details on the research methods and techniques adopted at various stages of this research. We present our rationale on the choice for a particular research method over another. We further detail the data collection and the data analysis techniques that we used. The application of each research method is described and can be found in the respective chapters where the respective method is applied.

2.1 Research Approach

As already indicated in Chapter 1 (Section 1.6), this research adopts a literature review, qualitative interview and a survey-based process for the purpose of problem investigation, treatment design and treatment validation. To materialize this project, Design Science Methodology for Information Systems and Software Engineering proposed by Wieringa [29] is selected. During each phase of the design cycle, multiple research methods were used as they produce results that are more robust and compelling than single method studies [43].

We used semi-structured interviews, questionnaires and documents to gather the data and grounded theory approach to analyse the collected data. Our data analysis was guided by the reasoning that underlies the sense-making techniques associated with the less procedural versions of the Grounded Theory (GT) [44]. Specifically, we deployed the guidelines of Charmaz’ Constructive Grounded Theory building method [32], which is an approach used in social sciences to construct general propositions (called a ‘theory’ in this approach) from textual data. These general propositions summarize beliefs embedded in the textual data. The method is exploratory and serves in situations where the researcher does not have pre-conceived ideas. It is driven by the desire to capture as much general information as possible from collected qualitative data and to allow the theory to emerge from the data. Our choice of using GT for data analysis agrees with Matavire and Brown [45] who profiled the use of GT in information system research. A notable example of a qualitative study of this nature is the one by Ramesh et al. [58].

In order to validate our approach, we used a combination of experts’ opinion and mechanism experiment. To validate a treatment is to justify that the proposed treatment would contribute to stakeholder’s goals when implemented in the problem context. The goal of validation research is to develop a design theory of an artefact in context that allows us to
predict what would happen if the artefact were transferred to its intended problem context [29].

2.2 Research Strategy

A variety of research strategies are available that could be applied to any research problem, and often a combination of several viable research strategies is needed at each step of the research to fully understand the problem, to design a suitable treatment and to validate it [46]. Each research strategy has its own set of advantages as well as disadvantages. No strategy can be claimed to be more appropriate than all the others for all research problems. It is the nature of the research topic and the research goals that actually influence the choice of the research strategies [50].

Table 2.1 shows the research strategies and the corresponding research methods that we have used for finding answer to each of our research question mentioned in Chapter 1 (Section 1.5 on page 6-7). Research strategy used in each chapter is already depicted in Chapter 1 (Figure 1.1 on page 8). In Table 2.1, we elaborate it further with corresponding research questions, research method and measurement method used.

**TABLE 2.1 OVERVIEW OF RESEARCH METHODS AND MEASUREMENT METHODS USED**

<table>
<thead>
<tr>
<th>Research Question</th>
<th>Research Methods</th>
<th>Measurement Method</th>
</tr>
</thead>
</table>
| **Problem investigation RQ1:** How do SAs currently identify architecturally significant information from a SRS? | o Literature review  
  o Case study                         | o Interviews  
  o Document Analysis                 |
| **Treatment Design RQ2:** How to use the mechanism to equip BAs to ask more architecturally relevant information? How could we design an approach that integrates the mechanism in an organization’s software delivery process? | o Literature review  
  o Case study  
  o Survey  
  o Experiment                       | o Interviews  
  o Questionnaires  
  o Document Analysis                 |
| **Treatment Validation RQ3:** Is the proposed way of working usable and useful in practice? | o Expert opinion  
  o Experiment                        | o Interviews  
  o Prototyping                       |

In the remainder of this chapter, we discuss each research strategy and method that we used in further detail.

**2.2.1 Literature Review**

A literature review is an essential part of any research project. It synthesizes current knowledge pertaining to the research question(s). Literature reviews have different purposes depending upon the nature of the enquiry [47]. In this dissertation, we utilized the literature
review to argue a position about the current state of knowledge on our research topic. Hence, in this work, as per [47], the literature review that we performed qualifies as ‘a simple literature review’. As is the case with simple literature reviews, our goal is to document, analyze and draw conclusions about what is currently known about ASRs. Fink [48, pp.3] succinctly defines literature review as “a systematic, explicit, and reproducible method for identifying, evaluating, and synthesizing the existing body of completed and recorded work produced by researchers, scholars and practitioners”.

A systematic literature review is a rigorous review of previous research on a given area and therefore is considered as the most followed practice [41]. As a part of this dissertation work, we however have done a semi-systematic literature review. Our literature review qualifies as semi-systematic as we only partially followed the systematic review process propose by Kitchenham [41]. Through this literature review, our goal was to understand (1) what others in the same area of interest have published, and (2) how our work links to or compliments previously published work. To achieve this goal, we searched for relevant literature using several electronic indexing services such as Google scholar, ACM digital library, Citeseer library, Science Direct and IEEE Explore. When we found any relevant literature, we also traced all the references in that literature to find other relevant sources.

We formulated the following inclusion criteria [106]: (1) the paper is peer-reviewed (2) the paper is written in English language (3) the title, keyword list and abstract of the paper is within the scope of our research topic. Next, we used the following exclusion criteria: (1) the paper is outside the scope of our research topic, (2) the paper is not available for download, (3) the paper is not written in English language. Given the above defined inclusion and exclusion criteria, we cannot claim our literature review to be completely free from omission and inclusion errors. For example, there may be relevant literature published in other languages as well.

Our literature review was conducted both during the problem investigation phase and the solution design phase.

**2.2.2 Case Study**

Case studies are well known to provide a deeper understanding of the phenomena under study. Runeson et.al [49, pp. 13] define Case study in software engineering as “an empirical enquiry that draws on multiple sources of evidence to investigate one instance (or a small number of instances) of a contemporary software engineering phenomenon within its real-life context, especially when the boundary between phenomenon and context cannot be clearly specified”. Case studies typically combine data collection techniques such as interviews, questionnaires and observations. The data collected in empirical studies can be qualitative (involving words, pictures etc.) or quantitative (involving numbers and classes) or both [49, 51]. A case study can cover either single or multiple cases [52]. There are 4 different types of purposes of case study namely **Exploratory** (to explain a specific situation), **Descriptive** (to provide description), **Explanatory** (to illustrate certain topic) and **Improving** (to improve certain aspect of the studied phenomenon) [49]. We have used case study for exploratory
purposes i.e. to understand from practicing BAs their notion about ASFRs. In the early stages of our research (during problem investigation and treatment design), we used exploratory case study to discover deeper and practical insights into the studied phenomena. In the later stages of the research (during treatment validation), we used case studies to validate our proposed approach from different perspectives.

In addition to the above, our motivation for using case study method rests on the following:

1. Our research is on a practice-based problem where experience of the actors is important and case study is suitable in such situations [49].
2. Our intent is to delve into things in more detail and discover things that might not have become apparent through more superficial research. Case study facilitates in-depth study of any phenomena [53].
3. Case study emphasizes on natural settings rather than artificial situations. As Yin [54] stresses, the case is a ‘naturally occurring’ phenomenon. It exists prior to the research project and, it is hoped, continues to exist once the research has finished.
4. Case study approach allows the researcher to use a variety of sources, a variety of types of data and a variety of research methods as part of the investigation [53]. This was suitable for an enquiry like ours.

2.2.3 Survey

According to Robson [56], a Survey is “a collection of standardized information from a specific population, or some sample from one, usually, but not necessarily by means of a questionnaire or interview”. Performing surveys is a well-known strategy for doing empirical studies. It is especially known from its use in disciplines like management science and social sciences [55].

Surveys can be taken by paper questionnaires, web forms, oral interviews or other data collection means [30]. A survey is useful when the research is expected to have a wide and inclusive coverage. The survey approach works best with clear and narrow targets in terms of information it is trying to gather. The approach lends itself to dealing with specific issues and is at its best when the researcher knows in advance precisely which factors are important and what kind of information is needed.

Surveys have relatively lower internal validity but better external validity in comparison to case studies and experiments. Better external validity can be attributed to the fact that surveys typically involve a large number of people from different real-world contexts. Therefore, the results are better generalizable when compared to controlled experiment or case studies [55]. We would discuss more about threats to validity in section 2.4.

We choose survey during our treatment design phase to validate with practicing SAs our initial PQ-flows that we had created and based on the response from the survey, we improved the initial PQ-flows.
2.2.4 Experiment

Denscombe [53] defines experiment as “an empirical investigation under controlled conditions designed to examine the properties of, and relationship between, specific factors”. A treatment is systematically applied to subjects to observe how they respond. A treatment is applied for knowledge gathering only. Some of the known advantages of experiments are their repeatability, precision, convenience and credibility [53]. The most talked about disadvantage with of experiments as a strategy is that they are carried out in artificial settings. Therefore, there are question marks about whether the experimental situations create conditions comparable with the ‘real-world’ situations.

A case-based experiment is a test of a mechanism in one or more object(s) of study with a known architecture. When the test is in a single object of study, it is called single-case mechanism experiment and when there is more than one object of study, it is referred to as multiple-case experiment. In such experiments one is interested in the mechanisms that produce the responses to the treatment. This contrasts with sample-based experiments, where the interest is in average and variation in outcomes, and hope to show a difference in outcomes between a treatment A and a treatment B.

In this dissertation, we have used the so-called ‘single case-based experiment’ for treatment validation. In such an experiment, we are interested in the mechanisms that produce the responses to the treatment. We make the note that the case-based experiment and the researcher’s interest in the mechanisms contrasts with sample-based experiments, where one is interested in average and variation in outcomes, and hopes to show a difference in outcomes between two treatments (e.g. assuming A and B are treatments, a sample-based experiment is interested in revealing differences in outcomes of treatment A and treatment B). We would use case-based experiment to test our approach by asking BAs and SAs to perform requirements specification and architecture design tasks using our approach.

2.3 Research Methods

In the previous section we explained the various research strategies that we have adopted in this dissertation. We further included discussion on the advantages and disadvantages of each of the chosen strategy and explained our rationale for choosing what we have chosen. In this section, we will discuss the corresponding research methods used while conducting the research. There are four main research methods used to collect data. These are questionnaire, interviews, observation and documents. As with research strategy, each research method also has its own advantages and disadvantages. Selecting suitable method(s) requires careful consideration of the research design as well as the pragmatics of the research setting. In this dissertation we have used semi-structured interviews, questionnaires and documents to gather data. We explain each of these in the sub sections below.
2.3.1 Semi-structured Interviews

Semi-structured interviews are the most widely used data-gathering technique in qualitative research [42]. In semi-structured interviews, the answers are open-ended and there is more room for the interviewees to elaborate on their points of interest. Further, it provides the flexibility to the researcher to ask follow up questions, depending on the course of conversation. As compared to other qualitative research strategies such as structured questionnaires, it often provides a more flexible approach by letting us better investigate interesting issues that appear during conversations [32]. Allowing interviewees to ‘speak their minds’ is a better way of discovering things about complex issues and, generally, semi-structured and unstructured interviews have as their aim ‘discovery’ rather than ‘checking’ [53, pp. 176].

We used in-depth one-to-one semi-structured interviews for data collection during the problem investigation and treatment design phase. We chose one-to-one interviews as it is the best suitable technique for our research context and also because it makes it easier for the researcher to control the data collection process.

2.3.2 Questionnaire

The questionnaire is a data collection technique wherein information is supplied directly by people (respondents) in response to pre-defined questions asked by the researcher. They are designed for both statistical quantitative analysis of responses as well as to collect qualitative information. The questionnaire may be self-administered, posted or presented in an interview format. A questionnaire may include check lists, attitude scales, projective techniques, rating scales and a variety of other methods. Questionnaires are an apt choice when the intent of the researcher is to gather data from large number of geographically dispersed respondents, when what is required tends to be fairly straightforward and the need is for standardized data [53]. The main advantage of questionnaire over other survey techniques, such as interviews, is that it allows better access to larger number of respondents from geographically dispersed locations, supply standardized answers, incurs relatively lower cost, and eliminates interviewer bias. There are some disadvantages as well such as lack of follow-up mechanism and some questions being left unanswered.

We used an online questionnaire for the survey we conducted during the treatment design phase. The survey was used to validate the PQ-flows with practicing SAs from different organizations and geographically dispersed locations. Our survey was qualitative in nature. Online questionnaire was an apt choice for us as: (1) our intent was to reduce the potential bias of the researchers and the senior SA who were originally involved in creation of the initial PQ-flows, and (2) it enabled the inclusion of large number of participants which in turn enabled us to recognize trends in responses as the design of the PQ-flows was incrementally improved. We further kept the number of questions as small as possible in an effort to encourage participants to complete the entire survey.
2.3.3 Document Analysis

Documents are often used as a supplementary strategy to the other data collection methods [57]. Documents can be analyzed both in qualitative and quantitative way. Documents are most often available as texts, but they can also be available in the form of databases, pictures or even sound.

We used documents during problem investigation and treatment design phase. Our documentary data archive comprised of real-life SRSs from insurance domain. These documents were of restricted access and they were “sanitized” by the software organization who owned it before granting us access to analyze them. These documents were analyzed to (1) establish if the SRS documents in fact lack the architectural details needed by SAs to make informed architectural decisions, and (2) for our experiments intended to provide an automated support for annotating the PQ-flows.

2.4 Threats to Validity of Research Results

This section presents discussion on the validity threats [68, 69] to this work and how we addressed them. We make a note that the threats pertinent to each study would be discussed separately in each of the chapters in which the study is detailed along with details of the mitigation strategies adopted and its effects on the results that we obtained.

Research methodologists, for example Wieringa [68] and Yin [69], advice to consider and mitigate the following possible threats to validity:

- Construct validity
- Internal validity
- External validity

In the following sub-sections, we detail each of these threats to validity.

2.4.1 Construct Validity

Construct validity involves making inferences from the sampling particulars of a study to the higher-order construct they represent [67, pp. 65]. It is essentially a check on the degree to which evidence found in a study matches its theoretical definition. In essence, it emphasizes to link events, data and theory.

We accounted for construct validity in our research by conducting an extensive semi-structured review during the problem investigation phase and solution design phase, which resulted in, among others, the definitional foundation for our research.

Furthermore, in case study research, construct validity is especially problematic because of potential subjectivity on the side of the researcher. Yin [69], proposes three alternatives to mitigate this:
Using multiple sources of evidence,
Establishing a chain of evidence, and
Having a draft case study report reviewed by key informants.

To counter this, we ensured that we do not make any subjective judgments from interview data. We stayed close to the interview transcripts by way of providing citations and even using “in-vivo” codes [32] and got the data analysis report reviewed by some interviewees. This way we ensured that our data represents reality, rather than our own subjective judgement. We constructed concepts bottom-up, from our data, using grounded theory approach.

2.4.2 Internal Validity

Internal validity is one of the most essential manifestations of validity in qualitative research. It aims at ensuring that the collected data enables the researchers to draw valid conclusions [59]. It assures that the results of the research are caused by the phenomena under investigation and are not random fluctuations. It validates if an explanation that a difference in an outcome is caused by the difference in treatments, is valid i.e. well supported by evidence and by arguments based on the research setup. In other words, if we find a significant correlation between treatment and outcome, then internal validity is the level of support for the explanation that a difference in treatment causes a different in outcome. Such a level of support must be found in the research setup, in particular in the plausibility that the “other” causes are unlikely. Based on the above argument, a generalized definition of internal validity would be as follows: internal validity is the level of support, in terms of evidence and arguments and in terms of the research setup, for our causal, architectural or rational explanation of the outcome.

In the qualitative interview based study that we conducted, our first concern was the extent to which the participants answered our questions truthfully. We minimized this threat by involving volunteers under the assumption that if a practitioner cannot be honest, he/she could decline his/her participation at any stage of the research. Secondly, we did peer briefing [61, 62], that is, we got our data and research methodology reviewed by senior researchers experienced in software engineering and empirical studies. Thirdly, it is possible that the researcher might instill her bias in the data collection process. To counter this threat, we applied Yin’s recommendations [24] by (i) including participants working in diverse application domains, which allowed the same phenomenon to be evaluated from diverse perspectives (data triangulation [24]); (ii) sending the interview answers to each participant prior to data analysis to confirm the information he/she provided; and (iii) getting the data analysis report reviewed by some interviewees (some of whom provided feedback on clarifying the concepts, but this did not change the analysis). Further, we used description that are very close (verbatim [63]) to interviewee’s answers to ensure richness and correctness of the gathered data. The fourth weakness is the possibility of asking leading questions (suggestive questions) and implying that one answer might be better over others. We, however, made every attempt to leave our interviewees to respond on their own terms,
expressing their own perspectives and values, and giving examples of what they do and how they cope in particular projects.

2.4.3 External Validity

External validity is concerned with whether the research results are transferable to a wider context [64]. “It concerns with the ability of research findings to explain, or occur in, similar phenomena at a general or universal level rather than being something that is unique to the particular case(s) used for the research” [53, pp.313]. It is also referred to as generalizability.

We used multiple research techniques (i.e. interviews and online survey) for carrying out the ground work that formed the basis for contextualizing PQ-flows. We evaluated the external validity threat in this multi-method research by using the checklist in [29]. As our design of PQ-flows was exploratory in nature and relied on refining the PQ-flows at each phase by collecting practitioners’ feedback, the generalizability of our PQ-flow structures was our most important concern. To what extent can the observations of the interview study participants and the participants of the two surveys be considered representative for other project organizations? We cannot claim universal generalizability of the PQ-flows that we derived by using the responses from our participants. While not all settings in which the SAs infer hidden meanings from ASFRs are similar to the companies included in this case study, some of them are [65, 66]. It is reasonable to expect similar experiences in projects and organizations working in the application domains in which our practitioners work. For example, other organizations in sectors that must ensure compliance of their IT-solutions to business sector-specific regulations might have observations similar to those in this case study. Further, to address generalizability, we introduced the concept of an evolvable KB of PQs and PQ-flows that would allow SAs to add or update the ASFR categories and PQ-flows based on their specific project needs.

2.5 Summary

This chapter presents details on the research strategies and methods adopted at various stages of our research. Our research follows the design science methodology [68] and is comprised of three research phases – problem investigation, treatment design and treatment validation. We further presented our rationale on why we choose a particular research strategy or method over the other. Finally, we discussed the possible threats to validity of our research and also discussed the ways we adopted to mitigate them.
This chapter positions our work with respect to relevant existing work in the area of RE and software architecture. To the best of our knowledge, at the time of writing this dissertation, there is no systematic empirical research that directly addresses our RQs. In this chapter, we address our first research question “How do SAs currently identify architecturally significant information in a SRS?”, and the sub-questions therein (see Section 1.5 in Chapter 1).

3.1 Introduction

As a part of preparing this dissertation, we searched for related work on four topics which are relevant to our work. These are: (1) ASFRs both in RE and software architecture literature (2) the synergistic relationship between requirements and architectural design (TwinPeaks of Requirements and Architectural Design) (3) automated identification and classification of requirements (4) KB based approaches to organizing architecturally relevant requirements knowledge. As stated in Chapter 2 (Section 2.2.1, page 17), we performed a semi-systematic literature review with the goal to document, analyze and draw conclusions about what is currently known about ASRs. To achieve this goal, we searched for relevant literature using several electronic indexing services such as Google scholar, ACM digital library, Citeseer library, Science Direct and IEEE Explore. When we found any relevant literature, we also traced all the references in that literature to find other relevant sources.

In the subsequent sub-sections in this chapter, we explain the related work on each topic in detail.

3.2 Architecturally Significant Functional Requirements (ASFRs) in RE and SA literature

We found many publications from the field of RE that are relevant to our work. It is worthwhile noting that although in the RE literature, there is much research on ASNFRs, we do not include it here as our focus is on ASFRs exclusively. We would therefore focus only on work in the field of RE in relation to ASFRs and to ASRs in general. We found 7 relevant publications. We summarize each of these below.
Chen et al. [4] presented an evidence-based framework to systematically characterize ASRs. Their framework consisted of four sets of characteristics: Definition, Descriptive, Indicators, and Heuristics. In particular, their finding that ASRs tend to be described vaguely, is a motivation for this dissertation work. Khan et al. [71] presented an analysis model that categorized requirements dependencies that are architecturally significant in terms of change impact. They reported on an exploratory study based on the change history analysis of a real-life web-based information system in order to gather the most ASRs’ dependencies from their model. Niu et al. [19] introduced a practitioner-oriented approach to analyze and evaluate ASRs for enterprise systems. However, their focus is on NFRs exclusively. Eeles [126] focused on an approach to gather requirements of particular significance to the architecture of a system. The article provided a list of ASFRs which is a small sub set of the ASFR categories that we identified through the interviews. Further the article acknowledged that it is a challenge to gather architectural requirements from stakeholders and also proposed using a questionnaire to gather architectural requirements. However, unlike our approach, their approach is not grounded in sound empirical evidence. Further, they do not organize the questions in the form of logical flows and also do not provide a KB for evolving the PQs or the ASFR categories. M. Jackson [127] proposed problem frames by means of which he identified frequently occurring problem structures in requirements. The first problem frames identified by Jackson included: Required behaviour problem - some part of the physical world needs to be controlled so that it satisfies certain conditions. Commanded behaviour problem - some part of the physical world needs to be controlled according to the commands of an operator. Information display problem - Information about some part of the world needs to be displayed. Work pieces problem - Some data, text, graphics, etc. needs to be manipulated. Transformation problem - data needs to be transformed into other data. These essentially are generic problems. These problems are so generic that we see them everywhere and they do not seem to help in specifying the requirements correctly and completely. Jackson introduced variants of the problems but the resulting larger set of problems is still very generic. He defined and discussed a list of concerns, such as overrun, initialization and identity that, if ignored, could lead to incorrect or incomplete requirements. However, we identify frequently occurring requirements, rather than frequently occurring problems. The difference is that requirements are desired behaviour or desired properties of a software system, whereas Jackson’s problems are goals or needs that need to be fulfilled in the real world. Banniassad et al. [128] described how to identify and capture early aspects in requirements and architecture activities and how they are carried over from one phase to another. Our approach also focuses on requirements-level aspects. However, unlike their approach, our approach is grounded in sound empirical evidence. In our approach, we organize the PQs in the form of logical flows and provide a KB for evolving the PQs and/or the ASFR categories. Chauhan and Probst [163] proposed a framework for ASR Identification, Classification and Change Management exclusively for multi-tenant cloud-based systems. They have identified a set of critical ASRs which are essentially NFRs for the cloud-based systems. They further classified the ASNFRs into three classes including system management requirements, communication and collaboration requirements, and monitoring requirements. However, unlike our approach, their work is relevant exclusively to cloud-based systems and they focus only on NFRs.
As is evident from the discussion presented above, there is no work in the RE literature that
directly addresse (Chapter 1, page 6 - 7). Our study brings out how SAs cope with
the process of inferring hidden meaning from ASFRs that are buried in the SRS received from
BAs. Through a sound empirical study, we identified a set of ASFR categories and PQs
pertinent to each of these categories. To cater to the needs of varied projects and business
domains, we created a KB of ASFRs and PQ-flows that facilitate SAs to evolve the KB with
ASFRs and PQs that relevant to their project settings. We further provided an automated
support for recommending relevant PQ-flows for an SRS and then generating annotated PQ-
flows wherein the annotations on the PQs contain possibly relevant requirements from the
SRS that already answers some of the PQs. The BAs can leverage this information to
determine what is already known, what knowledge is missing, and what effect this would
have on the flow of the PQs.

As in the RE literature, in the software architecture literature as well there is much research
on ASNFRs but we found very few in relation to ASFRs. As stated in section 3.1, we do not
include the NFR related work here as our focus is on ASFRs exclusively.

In the discipline of software architecture, a 2013 systematic review on empirical studies by
Quershi et al. [10] indicated the prominent role of NFRs with respect to software architecture,
but reported no study dealing explicitly with the topic of ASFRs. Moreover, the 2013 mapping
study of Li et al. [73] on the use of KB based approaches in software architecture indicated
that the phenomenon of architectural knowledge recovery has been under-researched and
“needs to be explored seriously” (pp. 790). This contributed to the motivation for this
dissertation work.

3.3 The Synergistic Relationship between Requirements and Architectural Design

Much work can also be found on the synergistic relationship between requirements and
architecture design [for example: Bass et al. [76], Dikel [77], Hofmeister et al. [78], Ran et al.
[79], Babar [80], Bosch [81], Clements et al. [82, 83], Garland [84], Ferrari and Madhavji [70],
Salehin [74] and Gross, Dörr [27]]. In 2001, Nuseibeh proposed the Twin Peaks model to
depict the interdependencies that exist between the problem and solution space – often
referred to as requirements and architectural design [85]. Since requirements specifications
and architecture design affect and constrain each other, this model emphasized the need to
progressively discover and specify requirements while concurrently exploring alternative
architectural decisions. Since then, many researchers have focused their efforts on
understanding the impact of requirements on architecture [Galster et al. [11], Cleland-Huang
et al. [12] and Gross and Dörr [27]].

There is also active work that investigates traceability from requirements to architectural
designs [Cleland-Huang [86], Goknil et al. [87], Omoronyia et al. [95], Grunbacher et al. [129]],
including links between key stakeholder concerns, ASRs, important architectural decisions
and sections of code where architectural decisions are implemented [86]. Goknil et al. [87]
used the semantics of traces, requirements relations and architecture verification techniques
to automatically generate and validate trace links between requirements and architecture leveraging model transformation and term-rewriting logic.

Grunbacher et al. [129] presented Component-Bus-System-Property (CBSP), a lightweight approach intended to provide a systematic way of reconciling requirements and architectures. CBSP leverages a simple set of architectural concepts (components, connectors, overall systems, and their properties) to recast the requirements in a way that facilitates their straightforward mapping to architectures. However, their approach works on stated requirements and they do not provide any option to explicate the unstated requirements. Ferrari and Madhavji [70] conducted an empirical study to understand the impact of requirements knowledge and experience (RKE) on SA activities. Their findings suggest that SAs with RKE perform better than those without. The authors further discussed the possible implications of their findings on the areas of training, education and technology. Hesse and Paech [164] provided insights through a study on discussion transcripts of two design sessions with professional software designers on how requirements could be enhanced with feedback from the design process, for instance by clarifying potential uncertainties with the stakeholders. This work compliments our work. In addition to the PQs we received from SAs, the decision-related knowledge created in the work by Hesse and Paech [164] could also contribute to creation of PQs.

Salehin [74] investigated in a case study the extent to which requirements or requirements attributes’ information is found to be missing during the software architecture process and the impact of that missing information on system architecting in terms of effort. The study suggested that SAs actively search for strategies to reduce missing information in the requirements, so that the software architecture is designed with less effort. This, in turn, highlights the importance and the need for PQs as a vehicle for deriving ASRs from dispersed FR and NFR specifications. Through our approach, we attempt to provide a solution to the problem identified by Salehin [74] in his case study. Yu [130] proposed the combined use of a goal-oriented language and a scenarios-oriented architectural notation. In their approach, goals are used in the refinement of FRs and NFRs, the exploration of alternatives, and their operationalization into architectural constructs. The architectural notation is used to depict the incremental elaboration and realization of requirements into architectural design. However, they discussed only the stated requirements and do not focus on explicating the unstated requirements that may have an architectural impact. In et al. [131] introduced an integrated framework for coordinating architectural decisions with requirements negotiation framework. Their proposed framework is based on WinWin [132, 133 and 134] and Cost Benefit Analysis Method (CBAM) [135], which aids in systematically determining architecture alternatives from negotiated requirements among stakeholders. Similar to Yu [130], the work of In et al. [131] is also limited to stated requirements. Brandozzi et al. [136, 137] proposed a technique for formal transition from requirements to architecture phase. They suggested the use of intermediate descriptions between requirements and architecture that they call ‘architectural prescriptions’, which describe the mapping relationship between requirements and architectures. Their method takes as input goal oriented requirement specifications (using a goal oriented requirements specification language) and returns as output an architecture prescription. The authors introduced an Architecture Prescription Language,
which specifies the structure of the software system and its components in the language of the application domain. This higher-level architecture specification can be then easily translated, if necessary, in an architecture description, in the solution domain. If we augment their approach with ours, the resultant ‘architectural prescriptions’ would be more elaborate, as using our approach, we enhance ASFRs with more architecturally relevant information. Gross and Dörr [27] reported first results of explorative studies aimed at revealing SA’s information needs that should be fulfilled in SRS. They indicated the relevance of certain requirements artifact types to SA, such as system responsibilities, data requirements, system functionalities, interactions, technical constraints, stakeholder goals and their suitable representation in terms of notations. Furthermore, these authors studied the variances among SAs in a group, regarding each one’s information needs. In their observations, factors such as expertise and individual motivation might influence the information needs of SAs. The authors call for further research into the information needs of architects and the factors that precondition them. The dissertation responds to this call.

Researchers have also addressed the problem of missing requirements [Poort et al. [22], Li [73]]. Poort et al. [22], proposed a facilitated method which elicits business goals and establishes the link between those goals and the quality attribute requirements for a system under development. It helps to discover missing quality attributes and empowers the SA to question the necessity of overly stringent requirements by appealing to stakeholder-expressed business goals.

3.4 Automated Identification and Classification of Architecturally Significant Requirements

Many techniques for identification, classification and mining various types of ASRs can be found in the literature. Most of work on such techniques can be found in the area of automating the identification and classification of NFRs [For instance, Cleland-Huang et al. [36, 92], Casamayor et al. [88], Jalaji et al. [93], Slankas et al. [94], Hussain et al. [140]]. Other than NFRs, work in the area of automating the detection of other requirements artifacts such as business rules and use cases are also present in the literature. Sharma et al. [89] presented work on automating the identification of business rules requirements documents. Ghaisas et al. [37], proposed an automated approach to detect system use cases and validations from documents. Ko et al. [90, 91] proposed an automatic requirement classification method to be applied in web-based analysis supporting systems. Boutkova and Houdek [138] proposed a technique for semi-automatic identification of features in natural language specifications based on lexical analysis. Sampaio et al. [139] proposed an Early Aspect Miner tool to provide automated support for identifying and separating aspectual and non-aspectual concerns as well as their cross-cutting relationships at the requirements level. Ferrari et al. [141], presented an approach for commonality and variability mining from domain-specific natural language documents. To the best of our knowledge, this is the first time the concept of PQ-flows and their contextualization is introduced.
3.5 Knowledge Based Approaches to Organizing Architecturally Relevant Requirements Knowledge

For any software organization, the main asset is its intellectual capital. To ensure that the intellectual capital remains intact within the organization, even when the key individuals leave the organization, it is important to invest in Knowledge Bases (KBs) [143]. The concepts of knowledge and organizational knowledge management are far from new. The Knowledge Management (KM) concept emerged first in the mid-1980s. Since then a lot of work in the area of KM has been done. KM technologies have been employed across SE activities for more than two decades [165]. KM facilitates the retention and distribution of intellectual capital within an organization and helps the organization to gain competitive advantage and create business value [143].

Hansen et al. [142] defined two main strategies for KM: (1) Codification strategy that was aimed at systematically storing knowledge in a repository, structured or unstructured, so that it can be made available to people in the organization and (2) Personalization that supported the flow of information in a company by having a centralised store of information about knowledge sources, like a “yellow pages” of who knows what in a company.

Ontologies which are an example of the codification strategy proposed by Hasen et al. [142], had been increasingly used for KM since long [143]. At the software industry level, groups of experts have identified patterns such as software design patterns by Gamma et al. [102] and generated handbooks and standards generally applicable to software development (e.g., IEEE and ISO standards). Another example of the codification strategy is the Software Engineering Body of Knowledge (SWEBoK), which was initiated in 1998 by the Software Engineering Coordinating Committee, a joint effort between IEEE Computer Society and Association of Computing Machinery (ACM).

As a part of the related work study, we looked closely at knowledge-based approaches for managing architectural knowledge. The importance of Architectural Knowledge Management (AKM) for software development has been highlighted over the past ten years. As a result, a variety of models, approaches, and research tools have leveraged the interests of researchers and practitioners in AKM [for example, Clerc et al. [150, 153], Farenhorst et al. [151], Jansen and Bosch [152], Clerc [166], van Vliet et al. [167]]. Capilla et al. [144] presented an informal retrospective analysis of what has been done and the challenges and trends for a future research agenda to promote use of architectural knowledge in modern software development practices. Galster and Babar [147] reported on an empirical study aimed at understanding how AKM is currently practised in industry. Li et al. [148] collected studies on the application of knowledge-based approaches in software architecture and make a classification and thematic analysis on these studies, in order to identify the gaps in the existing application of knowledge-based approaches to various architecting activities, and promising research directions. The study results showed an increased interest in the application of knowledge-based approaches in software architecture in recent years. We would like to highlight that we do not claim any novelty in creation of architectural KB. The novelty is in the knowledge that we store in our KB. To the best of our knowledge, there does
not exist any KB that organizes and stores architectural knowledge in the form of ASFR categories and PQ-flows. However, in the paras that follow, we present the work done in the past to create KBs for managing architectural knowledge.

A number of knowledge-based approaches, including knowledge capture and representation, reuse, sharing, recovery, and reasoning, have been employed in a spectrum of architecting activities. Tang et al. [149] provided a classification for architectural knowledge. According to them, architectural knowledge can be classified into 4 general categories:

1. **Context knowledge** - a collection of information about the problem space, for instance, ASRs and the context of a project.
2. **General knowledge** - a collection of knowledge that helps architects to design software and systems, for example, architectural styles and patterns, and tactics.
3. **Reasoning knowledge** - a collection of reasoning information about a design, for example, design decisions, design rationale, design alternatives, and trade-offs performed and
4. **Design knowledge** - a collection of designs of a system such as components and architectural models.

A variant of this model has been proposed by Bhat et al. [161], wherein the authors do not distinguish between context and design knowledge as they consider the design knowledge to be part of the context knowledge which is dynamic and evolves as the project progresses. They refer to the context and design knowledge as **dynamic** knowledge and the general knowledge as **static** knowledge. The general knowledge captures architectural methods, styles, patterns, and organization-specific corporate information (e.g., processes and standards) that helps SAs while designing software systems. Reasoning knowledge contains information that guides SAs to apply static knowledge in the project context. It also maintains design decisions, rationales, and alternatives that were considered during decision making in specific project instances, which can be reused in similar projects. In our work, we focus on the context knowledge, as defined by Tang et al. [149]. KM becomes more and more important in global settings [145, 146], the kind of settings in which we envisage our approach to be used. However, as stated earlier, to the best of our knowledge, this is the first time a KB to store architecturally relevant requirements knowledge is being attempted.

As a part of preparing this dissertation, we also reviewed knowledge-based approaches used in RE. We present some of the recent relevant publications here. Specifically, three recent systematic literature reviews [Sillaber et al. [168], Schneider et al. [169], Souag et al. [171]] supply information on the topic of our interest. In the first review [168], the authors identified 17 relevant studies and based on these studies reported challenges and benefits of the introduction of knowledge sharing platforms to RE processes. The main finding of this systematic review is that currently little empirical data on knowledge sharing platforms and their use in RE processes is available and further empirical research is necessary.

In the second review [169], the authors identified eight methods that can be applied to RE knowledge creation. They mapped the eight methods on six common RE problems and analysed to which extent the methods overcome the associated problems. The identified
methods provide an adequate approach to reduce the risk of potential RE problems, thus making project failure less likely. The authors concluded that there are various approaches or methods of KM that are of value for the RE and should be applied in this context to encounter the existing problems.

In the third review (which is a mapping study) [171], the authors identified, analysed, and categorized the state-of-the-art research on security knowledge reuse in security RE. Based on the mapping study, they defined a framework for analysing and comparing the different proposals as well as categorizing future contributions related to knowledge reuse and security RE. They further identified the different forms of knowledge representation and reuse. They concluded that most methods should introduce more reusable knowledge to manage requirements. However, the authors’ focus was exclusively on security requirements.

Most recently, in the 2017 study by Knauss et al. [170], the authors proposed that a KM framework can facilitate Just-in-time RE by structuring, representing, and allowing updates of long-term knowledge about quality requirements. Such a KM framework would allow to map user value to system requirements and have important properties to allow Just-in-time RE and sustainable evolution.

3.6 Summary

In this chapter we positioned our work with respect to relevant existing work in the area of RE and software architecture. We focused on four topics, namely: (1) ASFRs both in RE literature and software architecture literature (2) the synergistic relationship between requirements and architectural design (3) automated identification and classification of requirements (4) KB approaches to organizing knowledge pertinent to architecturally relevant requirements. From the related work presented in this chapter, it is evident that even though there is an acknowledgement in the academic community that requirements are never detailed enough to make informed architectural decisions [for instance, Eeles [126], Chen et al. [4], Gross and Dörr [27]], there is no systematic work done to remedy this situation. Next, automating the identification of various software artifacts have been attempted in the past. However, no work has been done so far in the area of identification and classification of ASFRs. Finally, using KBs to organize various kinds of software engineering knowledge is not at all a new idea. However, using the concept of KBs to organize the knowledge pertinent to ASFRs, as we do, is a new contribution.
4

Understanding Architecturally Significant Functional Requirements (ASFRs)\(^2\)

This chapter details the qualitative studies that we carried out to understand how architects perceive ASFRs, which categories of ASFRs exist in the covered domains, what kind of information is discoverable using Probing Questions (PQs) and how do SAs use these PQs to infer and explicate hidden architectural impact. This chapter is largely based on two publications [33] and [34].

4.1 Introduction

SAs are responsible for designing an architectural solution that satisfies the FRs and NFRs of the system to the fullest extent possible. However, the details they need to make informed architectural decisions are often missing from the SRS. In this chapter we present results from two studies (we refer to them as phase I and phase II study) we conducted with the aim of identifying various categories of ASFRs from various business domains, exploring relevant PQs for each category, and then grouping PQs by type. Both the studies were qualitative interview-based studies. Using the interview data, we first established that SAs often use PQs as a mechanism to extract architecturally relevant information form the stakeholders, identified 15 categories of ASFRs and 6 types of PQs. Second, we found that the domain knowledge of the SA and her experience influence the choice of PQs significantly. We further performed a preliminary quantitative evaluation of the results against real-life SRS.

---

\(^2\) This chapter is based on the following papers:

documents. This evaluation indicated that software specifications in our sample largely do
not contain the crucial architectural details that may impact architectural choices and that
PQs are indeed a necessary mechanism to unearth them.

4.2 The Research Plan and Execution Strategy for the
Exploratory Studies

We adopted the same research plan and execution strategy for the two studies. Whenever a
step in the research process is performed differently in any of the studies, we highlight it
while we explain it.

In what follows, we refer to the first exploratory study as to ‘Phase I study’. In the same vein,
we refer to the second study as to ‘Phase II study’.

4.2.1 Data Collection

We designed a qualitative interview-based research process by implementing R. Yin’s
guidelines for exploratory case study design [24]. We used qualitative interviews as data
collection technique because it is deemed useful when a researcher needs insightful and
context-specific information so that he/she explores an issue in depth. Combining both the
studies, a total of 24 practicing SAs from different geographies took part in our studies.

We used semi-structured open-ended in-depth interviews [96]. The steps we followed in
cconducting the interview based study is as follows:

1. Composing a questionnaire,
2. Confirming the interview questionnaire with 3 experienced researchers,
3. Implementing changes in the questionnaire based on the received feedback,
4. Conducting a pilot interview to check the applicability and feasibility of the
questionnaire to real-life context,
5. Conducting the interviews with the practitioners by using the finalized
questionnaire, and
6. Following-up with those participants that demonstrated deeper knowledge or
shared a more specific example of how the study’s phenomenon happens in real-
life. The questionnaire used during the interview can be found in Appendix A on
page 123.

The duration of each interview in the Phase I study was between thirty and forty minutes
while in the Phase II study, it was between sixty and ninety minutes. The interviews were tape
recorded. This was done with the consent of each interviewee. All our participants were
informed in advance of our research goals and the interview process. The interviews were on
a one-to-one basis. At each meeting, each researcher and the respective interviewee used
the questionnaire\textsuperscript{3} to guide the conversation. The questionnaire for the follow-up detailed study was an extended version of the questionnaire we used for the preliminary study.

As after the pilot interview, no substantial changes were made to the questionnaire and the case study protocol, we included the pilot interview also as an integral part of the case studies. Furthermore, during the interviews there were cases when other questions arose in addition to the ones included in the questionnaire. These questions were not previously anticipated, however the researcher conducting the studies considered them interesting and pursued the interview in that direction. As confidentiality agreements were a premise for the study, we cannot provide any information on the organizations employing the practitioners. However, as practicing SAs, their professional profiles had the following common characteristics:

1. All of them were currently working on large business information system projects,
2. All of them had worked as a developer in their initial professional years before taking roles of SAs,
3. Most of them were experienced in more than one domain,
4. Except 2 participants (P4, P14), all others work on fixed price projects, and
5. Except one (P14), all participants were SAs from the vendor’s side.

4.2.2 Data Analysis

All the tape-recorded interviews were transcribed by the author of this dissertation. Since the analysis was planned to be done in collaboration, while transcribing the interviews, the author of the dissertation also added her notes about the non-verbal communication of her interviewees, and clearly indicated where in the interview an interviewee paused and expressed doubts regarding the completeness or correctness of the information provided. This turned out to be critical information for the other researchers who were engaged in the coding.

Once the transcripts were ready, for the qualitative data analysis [104], we deployed the guidelines of Charmaz’ Constructive Grounded Theory (GT) building method [32]. Grounded theory is a systematic methodology used in the social sciences to construct general propositions (called ‘theory’ in this approach) through the analysis of textual data. These general propositions summarize beliefs embedded in the textual data. Grounded theory is a research methodology which operates inductively, in that the researcher has no preconceived notions or ideas to prove or disprove. The method is exploratory and allows emergence of theory from data. The researcher analyses the data by constant comparison initially of data with data, progressing to comparisons between their interpretations translated into codes and categories and more data [107]. Essentially, GT analysis includes ‘coding’ and ‘constant comparison’ of the interview data. The resultant codes and categories guide the writing-up of the results and aid in improving the accuracy of the claims [32].

\textsuperscript{3} The questionnaire can be found in Appendix A on page 123
The data analysis was done manually, without the help of specialized software programmes. Three researchers (the author of this dissertation, one supervisor from TATA Consultancy Services, India and one supervisor from University of Twente, Netherlands) first individually read the interview transcripts and attached a coding word to a portion of the text – a phrase or a paragraph or a sentence. The coding words were selected to reflect the relevance of the respective portion of the interview text to a specific part of the studied phenomenon. This could be a concept (e.g. ‘business rules’), or an activity (e.g. ‘operationalization of a FR’, ‘integration’) or abstract or social phenomena (e.g. feelings, culture, politics). At this first stage of data analysis, coding resulted in dozens of codes. We however observed that a big set of codes reappeared in every interview. After removing the duplicate codes, nearly 200 distinct codes were generated. The researchers then suspended their independent judgments regarding the concepts that were discovered individually and worked together to systematically reorganize their combined concepts into higher-level categories. This helped us understand the data better. We also shared and clarified assumptions to arrive at a consensus regarding how and why particular concepts were important. This resulted in a total of 35 categories. Next, the results of the coding and interpretation of the data were discussed and peer-reviewed iteratively with two other researchers (the collaborator from De Paul University The United States and the promoter from University of Twente, The Netherlands), to establish consistency and categorization of the emerging clusters. We discarded a few of the categories to scope it further within the purview of the phenomena under study. We finally ended up with 28 categories. These categories are listed in Appendix G on page 143.

4.3 The Phase I Study

We first conducted Phase I of the study with 8 SAs to develop an understanding of how they cope with ASFRs in real projects. The main objective of the Phase I study was to qualitatively explore if the SAs have encountered ASFRs in their projects; and if so, how they perceive and interpret them. Table 4.1 summarizes the details of the participants in Phase I study. The author of this dissertation conducted all the interview in the Phase I study. Even though all the SAs who participated in our Phase I study were from India, we would like to highlight that they worked in multinational companies; their clients were geographically dispersed, projects were global and were executed in outsourcing contexts. Therefore, we could consider the experiences of these SAs similar to experiences possibly observable in global projects of other companies across the globe.

From the analysis of the interview transcripts, we found that all the 8 SAs emphasized that almost all the times the architectural impact in FRs is implicit and the impact is unearthed only after a series of follow up questions are asked to the client. Further, the study revealed that the SAs intuitively recognize ASRs in a project, and often seek out relevant stakeholders in order to ask PQs that help them acquire the information they need.
### TABLE 4.1 PARTICIPANT DETAILS FOR THE PHASE I STUDY

<table>
<thead>
<tr>
<th>Participant ID</th>
<th>Application domain</th>
<th>Country</th>
<th>Total experience as a SA (Years)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Medical</td>
<td>India</td>
<td>9</td>
</tr>
<tr>
<td>P2</td>
<td>Banking</td>
<td>India</td>
<td>15</td>
</tr>
<tr>
<td>P3</td>
<td>Insurance</td>
<td>India</td>
<td>14</td>
</tr>
<tr>
<td>P4</td>
<td>Embedded Systems</td>
<td>India</td>
<td>22</td>
</tr>
<tr>
<td>P5</td>
<td>Insurance</td>
<td>India</td>
<td>19</td>
</tr>
<tr>
<td>P6</td>
<td>Insurance</td>
<td>India</td>
<td>20</td>
</tr>
<tr>
<td>P7</td>
<td>Banking</td>
<td>India</td>
<td>19</td>
</tr>
<tr>
<td>P8</td>
<td>Manufacturing</td>
<td>India</td>
<td>11</td>
</tr>
</tbody>
</table>

### 4.4 The Phase II Study

The main objective of our Phase II study was to qualitatively explore whether the SAs have encountered hidden FRs of architectural significance in their projects; and if so, how do they identify and interpret those FRs. Having learned from our Phase I study [Section 4.3] that PQs are used by SAs to seek additional architecturally relevant information, we next set out to answer the following research questions (RQs):

**RQ 1:** What categories of ASFRs in the covered domains contain hidden / implicit architectural impact?

**RQ 2:** From a SAs’ perspective, what kind of information is discoverable using PQs?

**RQ 3:** How do architects use PQs to infer and explicate hidden architectural impact?

To find answers to our RQs, our questionnaire (Appendix A, page 123) included three sections designed to:

1) Collect information about each participant’s experience and application domain.
2) Collect perceptions and observations with respect to each practitioner’s understanding of ASFRs, and
3) Collect elaborate details on practitioner’s understanding of PQs.

Table 4.2 shows the details of the participants of the Phase II study. Three researchers conducted the interviews in the Phase II study (the author of this dissertation conducted 14 out of the 16 interviews, while one of the supervisor from UT conducted the interview with the practitioner (P22) from the Netherlands and the collaborator on this project from DePaul University, United States of America conducted the interview with the SA (P 21) from the United States of America.
### TABLE 4.2 PARTICIPANT DETAILS FOR THE PHASE II STUDY

<table>
<thead>
<tr>
<th>Participant ID</th>
<th>Application domain</th>
<th>Country</th>
<th>Total experience as an SA (Years)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P9</td>
<td>BFSI, Healthcare</td>
<td>India</td>
<td>13</td>
</tr>
<tr>
<td>P10</td>
<td>Insurance</td>
<td>India</td>
<td>9</td>
</tr>
<tr>
<td>P11</td>
<td>BFSI, Healthcare</td>
<td>India</td>
<td>10</td>
</tr>
<tr>
<td>P12</td>
<td>Retail</td>
<td>India</td>
<td>5</td>
</tr>
<tr>
<td>P13</td>
<td>Banking</td>
<td>India</td>
<td>3</td>
</tr>
<tr>
<td>P14</td>
<td>BFSI, Retail, Government, Healthcare</td>
<td>India</td>
<td>7</td>
</tr>
<tr>
<td>P15</td>
<td>Telecommunication</td>
<td>India</td>
<td>6</td>
</tr>
<tr>
<td>P16</td>
<td>Banking, Medical</td>
<td>India</td>
<td>9</td>
</tr>
<tr>
<td>P17</td>
<td>Retail</td>
<td>India</td>
<td>12</td>
</tr>
<tr>
<td>P18</td>
<td>Insurance</td>
<td>India</td>
<td>14</td>
</tr>
<tr>
<td>P19</td>
<td>BFSI</td>
<td>India</td>
<td>15</td>
</tr>
<tr>
<td>P20</td>
<td>Insurance, Banking</td>
<td>India</td>
<td>11</td>
</tr>
<tr>
<td>P21</td>
<td>Banking, Airlines, Telecommunication</td>
<td>United States of America</td>
<td>15</td>
</tr>
<tr>
<td>P22</td>
<td>Telecommunication</td>
<td>Netherlands</td>
<td>4</td>
</tr>
<tr>
<td>P23</td>
<td>Telecom</td>
<td>Israel</td>
<td>24</td>
</tr>
<tr>
<td>P24</td>
<td>Retail</td>
<td>Ireland</td>
<td>20</td>
</tr>
</tbody>
</table>

### 4.4.1 Results

In this section we present our findings regarding our RQs. Figure 4.1 depicts some of the concepts and their relationships based on our findings. Therein, the dashed line is used to indicate that we enumerate a larger set of concepts than those depicted.

**RQ1 - What categories of ASFRs in the covered domains contain hidden / implicit architectural impact?**

In the experiences of our participants, ASFRs were never sufficiently detailed in the SRS to a point that the SAs could start designing the architecture of the software without eliciting further information from BAs or domain experts.
In the words of Participant P16: “The requirement is never elaborate enough to choose the correct architecture. There will always be a gap between our understanding and what the client is conveying. Communication plays a key role”.

Our data analysis suggests that several hidden architectural implications are found in ASFRs. Based on the kind of impact they have on architecture, we found that the following categories of ASFRs exist. Some of the examples are obfuscated for confidentiality reasons.

FIGURE 4.1: CONCEPTS AND THEIR RELATIONSHIP

1. **Audit Trail**: Audit Trail facilitates auditing of system execution. Twelve participants mentioned Audit Trail as an ASFR category. An example Audit Trail FR is: “System must record every modification to client records for audit purposes.”

   Participant P9 suggested: “An important question is to understand the purpose of audit. Whether it is for regulatory compliance or for internal compliance. Depending on this, your audit architecture would change”.

   Participant P22 suggested: “There are many regulatory compliances like SOX compliance. Not only the person who is doing the activity but also the originating transaction needs to be audited. So, based on that, many organizations had to undergo a drastic change in terms of the architecture they were using. Many who were using legacy system had to put a different system to replace the old system to support auditing.”
2. **Batch Processing**: This includes requirements facilitating batch processing. Eight participants mentioned Batch Processing as an ASFR category. An example requirement is: "The disbursement process should be a daily batch process for regular claim pay-outs".

Participant P13 mentioned: "A common thing to know is the frequency on which the batch process should run. One thing that we normally miss out is the privilege that it needs. What level of access specification you need for that batch to run?"

3. **Localization/Multilingual**: Localization/Multilingual involves providing support for multiple languages. Nine participants mentioned Localization/Multilingual as an ASFR category. An example requirement is: "During the term, the authority may require the contractor to deliver specific communications in other languages”.

Participant P18 suggested: "Localization has major impact on architecture. People usually go for upper end ERPs like Oracle or SAP as they have multi-language support”.

4. **Business Process “State” Alerts**: This class of ASFR is concerned with notifications, generated often as a part of executing a business process by the respective system. All 16 participants mentioned Business Process ‘State’ Alerts as an ASFR category. An example requirement is: "Workflow is required to send notification to Underwriters”.

Participant P13 mentioned: “There are two types of notification. One is where we do not expect a response. Second one is where the transmitter wants the receiver to send back something. We had a scenario where we actually missed out that. Because we thought that it was just a notification we receive from the client; but the client wanted the notification to be acknowledged. That was one of the architectural changes we encountered”.

5. **Data-related Dialog**: This category includes requirements specifying input data mechanisms.

Participant P22 described the various ways to enter data into a billing system that processes discount vouchers: "A voucher may be entered by means of QR code, by means of user’s click on a link sent by email, or by user’s manual data entry of voucher number”.

For example, using a QR code assumes an external devise (a smartphone) to be used as a QR code scanner, displaying the code and converting it to some useful form (such as a standard URL for a website). The smartphone would allow QR codes to send metadata to existing applications on the device and then hard-link it to an external URL.

6. **Payment**: This includes requirements facilitating financial transactions. Eight participants mentioned Payment as an ASFR category.

Participant P20 suggested: “Payment is always a problem area. You need to know whether the payment is to be done through card or through net banking. Whether you need to link it with one bank or with multiple banks. All this will impact architecture”.

---

40 | Page
7. **Print:** This includes requirements facilitating support for printing documents. Seven participants mentioned Print as an ASFR category. An example requirement is: “The commission statement should be printed using the package-supplied format”.

Participant P13 suggested: “The problem is that they (clients) would always want a web application for printing. Now, if you use web application for printing, you need ActiveX—a component that deals with the printer from the website. It is an important component without which printer would not work. ActiveX has a security compromise. This is always a matter of negotiation between client and technical team because client would initially say to go ahead with printing from website. When the technical team is ready with printer and during deployment they have to reduce the security of the browser. Now client would say that it is not a good design to reduce security of the browser. So printing from browser is always a confusion. So this is one architectural decision”.

Participant P21 said: “If you are having a simple desktop application, then directly give print to printer connected to your desktop. Printing is available in most of the technologies, components are available you can directly use them. What if your application is running in virtual environment, may be a virtual machine? No physical attachment to a physical machine. So how to handle such scenarios? You may need to route through other machines so that it gets printed in original single machine. Scope of requirement again affects architectural Design. Client might initially say that print it needed. When we drill down the requirement further, we see these types of complexities.”

8. **Report:** This category includes FRs that facilitate report generation. Twelve participants mentioned Report as an ASFR category. An example requirement is: “System should generate reports on complaint register on monthly basis”.

Participant P11 suggested: “Whether reporting requirement is for operational reports or for analytical reports? Whether it includes structured report or includes unstructured data as well? If somebody is giving some document and things like that and if you need to extract that information and try to produce some report, then it is more complex from architectural perspective than structured data”.

9. **Search:** This category includes FRs that provide support for search functionality. Eleven participants mentioned Search as an ASFR category. An example requirement is: “Claims Assessor should be able to search for claim records to be processed”.

Participants P13 suggested: “Search functionality is very big. This will need detailed level of analysis, whether we are looking for completely web-based search where we search for all text within web or we are looking for specific things, maybe I can have a drop down of all credit, I need to check all transactions of a particular client. You can have an advanced level search or a very generic search that will search through web. So for search, you would need a detailed level of requirements gathering to come up with right architecture”.


10. **Third Party Interaction**: This category includes requirements that facilitate interaction with third party components. Six participants indicated Third Party Interaction as an ASFR category. An example requirement is: “Once an application has been applied for, it will be exported to ABC back office where it may be processed both automatically by ABC back office system and manually by an underwriter.”

Participant P12 mentioned: “Yes, third party interaction is critical. We have to understand where the third party tool is going to be installed. Is it inside client architecture or outside the architecture? What sort of patterns are acceptable by third party? All this would decide your architecture.”

11. **Workflow**: This includes requirements that provide support to move work items, facilitate reviews and approvals. Twelve participants mentioned Workflow as an ASFR category.

Participant P10 and P20 suggested: “Workflow kind of requirements have a very big impact on architecture. After deployment, client would say he wants a workflow included. The entire architecture has to be reworked in this case”.

Participant P18 suggested: “Workflow requirements are the key things to decide on your technical architecture, because workflows in organizations are not straightforward.”

Participant P12 suggested: “This is the key thing. People go for separate workflow engine even though when package products are available. This is because, many of the standard workflows available in standard ERP etc., they have some limitations. They have limitation in terms of number of hierarchy, type of hierarchy they can use etc. and these are not customizable. The kind of hierarchy certain organization needs are complex. E.g. in [my organization], if I apply for leave, it follows a different hierarchy, my appraisal follows different hierarchy, if I request for laptop, it follows different hierarchy. So, when you are addressing this kind of versatile workflow needs, one way is to have in-system workflows for all systems, but maintenance of these workflows become heavy. Another way is to have a loop between all these systems in the standalone workflow system which can be tweaked as per the requirement for different business needs. So, workflow requirements are the key things to decide on your architecture.”

12. **Online Help**: This category includes requirements pertinent to providing online help facility. Two participants mentioned Online Help as an ASFR category. An example Online Help FR is: “An online help facility should be available for claim intimation process”.

Participant P11 mentioned: “Someone trying to ‘save some form’ type of online help is different from someone trying to just navigate for some information. Is it only information consumption type of online help or is it about how you navigate to get to the right information? Second one could be that you are the contributor of information, you need help on how to populate the information. So consumer verses provider could be one sub-category, and internal verses external could probably be another”. 
13. **Licencing**: This includes facilitating services for acquiring, installing and monitoring license usage. Three participants mentioned Licensing as an ASFR category. An example requirement is: “There should be facility for installing and monitoring license usage”.

Participant P11 suggested: “Yes this is also very important from architectural perspective. For licensing, there are different models available in market for different systems. Some could be user based, some could be CPU or system based, and some could be perpetual licenses. So, many variants are available. Then, based on the questions asked, you could choose the right model”.

14. **User Behaviour Analysis**: This includes FRs implied by the business goals of the client organization to collect, analyse and aggregate data on users’ behaviour. Examples of such ASFRs came from architects’ experiences in web-based systems with consumer-oriented front end, for example, e-commerce applications and retail systems. In these system’s contexts, architecting for User Behaviour Analysis means implementing web monitoring features.

As Participant P9 says, features for tracking the “Number of hits, number of page views, number of conversion, and number of orders processed”.

15. **Storage Mechanism**: This type of ASFR includes features needed for automatic handling of documents, converting paper to electronic files and integrating the files with client organization’s legacy business and possibly client relationship management systems.

Participant P11 indicated: “The client said that he wants to store all documents for x number of years. This looks very simple but from an architectural perspective, how would you store it, how would you maintain it is important”.

**RQ2 - From a software architects’ perspective, what kind of information is discoverable using PQs?**

We found that all our participants in fact were using PQs as a routine approach to ensure that the ASFRs included sufficient information to support architectural decisions in their projects. The PQs essentially unearthed the unspecified architectural details hidden in the ASFR statements.

Participant P16 suggested: “The scope of requirement again affects architectural design. Client might initially say that print is needed. When we drill down the requirement further, we see these types of complexities”.

Based on the concerns in the PQs specified by the SAs, we found that the following types of PQs exist to bring out the unspecified architectural details in ASFR:

1. **Business rules in client’s organization**: Our analysis suggests that most of the PQs are about the possible constraints on software system’s behaviour and/or the constraints that provide support to system’s behaviour in client’s organization. For example, in the
case of Audit Trail ASFR category, participant P9 suggested the following PQs which in fact point towards one or more business rules in client’s organization:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>How do you enable audit trail?</td>
</tr>
<tr>
<td>2.</td>
<td>How long does the audit trail need to be kept?</td>
</tr>
<tr>
<td>3.</td>
<td>What audit trails need to be maintained?</td>
</tr>
<tr>
<td>4.</td>
<td>What information does business want to see in the audit trail?</td>
</tr>
<tr>
<td>5.</td>
<td>How often will the audit trail be retrieved?</td>
</tr>
</tbody>
</table>

2. **Strategic technology choices**: PQs of this type are concerned with the architectural impact of the investment decisions of client organization in specific hardware and software technologies. For example, users’ devices that should be accounted for as part of technology infrastructure on which the system (for which a SA is responsible) would run or specific statistics that need to be collected, tracked and aggregated at system’s run time, for user’s behaviour analysis. Such PQs were put forward by participants working in e-commerce systems or retail web-based systems that assume the implementation of specific tools for web monitoring.

3. **Non-functional aspects related to ASFRs**: Many PQs referred to relevant non-functional aspects that should be considered while making an architectural decision. PQs of this nature unearthed NFRs, which if not probed, would be left unattended or assumed. For example, participant P10 suggested the following PQs related to scalability and security:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>What is the volume of data?</td>
</tr>
<tr>
<td>2.</td>
<td>How many master tables are needed for audit purposes?</td>
</tr>
<tr>
<td>3.</td>
<td>What level of security is needed in case of sensitive data?</td>
</tr>
</tbody>
</table>

4. **Regulatory compliance**: PQs pertinent to regulatory compliance have major influence on architectural decisions. This was explicitly suggested by five of the interview participants.

Participant P9 said in case of the Localization category of ASFR: “Are there regulatory or legal requirements in that region that necessitate a particular kind of content delivery? Regulatory requirement is a big thing”.

Regarding the Audit Trail category of ASFR, Participant P11 mentioned: “An important question is to know the purpose of audit. Is it for regulatory compliance or for internal compliance? Depending on this your audit architecture would change”.

5. **Project Context**: These are PQs concerned with contextual project parameters that impact software architecture. This includes PQs pertaining to functional fit, business case, cost considerations, time frame, technology stack available with the client, skill set availability on both vendor’s and client’s side, organizational culture, mentality of the client’s organization, and even strategic requirements imposed on the project.
On cost considerations, participant P16 suggested: “With SMS, you need to consider whether a service provider is needed. You need to see whether you need a free service or you have budget to pay for the service. Client says you just send SMS, I don’t want to pay for it. In this scenario you need to go for free providers. Thus, client’s budget and timeframe will affect architectural decision”.

Furthermore participant P11 said: “I think you need to understand why typically you should not get into the batch operation unless there is a valid reason for it. So, why you need a batch process would be a question to ask. Is it because of the technology or is it because of some real functionality?”

On strategic considerations, participant P22 indicated: “Sometimes it is hard to arrive at a consensus. So the requirements you are asking about are strategic requirements. Whose systems to keep? Whose systems to phase out?”

On business case, participant P22 suggested: “The impact depends on whether we need a temporary solution till the period we are waiting for the adoption of a better platform. In this case, you can tolerate a less good architecture. We will drop the system as soon as we update the infrastructure or we make a long term investment. We have a portfolio of infrastructure projects that phase out old systems and get new ones. So, the impact depends on where in this spectrum your project is. In other words, the business case matters”.

6. **Compound architectural effect on two or more ASFRs**: A PQ of this type provides an answer to more than one category of ASFRs.

Participant P14 used this PQ when asking for payment transaction processing details: “What confirmation would you be receiving from third party gateway to indicate success of payment?”

This PQ jointly addresses the ASFRs of Business Process ‘State’ Alerts and the Third Party Integration categories.

**RQ 3 - How do architects use PQs to infer and explicate hidden architectural impact?**

Our participants indicated that asking PQs seemed an intuitive and common sense approach to unearthing the unspecified architectural details from the ASFRs. We found that their reasoning about what to include as PQs was traceable to one or more of the following:

1. Domain knowledge,
2. The SA’s knowledge about the nature of the client’s business and in the vendor’s processes of delivering projects, and
3. Maturity of the client organization.

More specifically, in the perceptions of our practitioners, these concepts had the following roles in the SA’s judgments of what PQs to ask:
Domain knowledge was deemed critical in operationalizing ASFRs in terms of detailed steps (for example, as illustrated in the Audit Trail category of ASFR). Furthermore, domain knowledge is instrumental in bringing out the constraints on system behaviour.

Participant P21 stated: “Yes, you can recognize ASFRs immediately if you have experience of a domain. In new domains you may be suspicious about them, but with some digging [PQs] you could classify them. Experience plays a significant role”.

The SAs’ knowledge about the nature of client’s business and about the vendor’s processes of delivering projects was important for formulating the PQs concerned with the client’s infrastructure and strategic technology choices.

Participant P10 mentioned: “The application is needed on desktop, laptop and tablet. Client will not explicitly say that he wants it on the tablet. After implementation, they would say that they want it on tablet. Generally if you are not a seasoned architect, you would not take care of it”.

Also in the experience of P10: “You do not want to recommend an open source solution to a health care client where data availability is a very critical requirement and you do not want to depend on community supported systems for such critical requirements. You may want a solution suitable for mission critical systems”.

Example of knowledge about vendor’s processes are the PQs concerned with parts of the business case. For example, money considerations, resource availability, business criticality and affordable or desired time frame for the project.

Maturity of the client organization in terms of process-oriented thinking was deemed important for judging the relevance of PQs to be asked, because it is reflected in the way the FRs are documented. In the experience of our participants, more mature client organizations usually write their FRs with more discipline, which ensures breadth and depth of the documents provided to vendors.

As Participant P14 suggested: “The maturity with which FRs are documented varies considerably from one client to another. Sometimes we get FRs which are as good as technical specifications. But sometimes we get a FR which is at a very abstract level. Thus PQs will depend on the level of maturity and detail with which the FRs are mentioned”.

We were also interested in discovering how the SAs acquired answers to the PQs. We found that this depended on:

1. The project organization’s setup, and
2. Organizational hierarchy of the SA

We found that most of the time, the SAs escalate the PQs to the respective RE-specialist (the BA in their projects), who then forward the PQs to the business managers at the client’s organization. However, sometimes, the SA is a person with skills from both business and architecture side, in which case the SA poses the PQs directly to the clients.
Further, we wanted to know what the SAs do with the answers to the PQs. Fourteen out of our sixteen participants, amended the FR document by adding the newly discovered ASFR details. In one case, the SA used the repository of his company’s Architecture Office to document the ASFRs pertaining to his particular project.

4.5 Discussion

This section compares our finding to those in previously published studies relevant to our work. It also provides possible explanations about why we observed what we observed. Our discussion follows our RQs.

RQ 1: What categories of ASFRs in the covered domains contain hidden / implicit architectural impact?

We found 15 categories of ASFRs. One of these categories, the Third Party Interaction, overlaps with the infrastructure-related ASFRs studied in COTS context in [72]. Our findings agree with the results of Ernst et al. [72] that components of the infrastructure – tools, platforms, converging technologies, may have significant architectural impact that are not obvious when the FRs are elicited and tools selected. However, we found 14 additional categories of ASFRs that have not been addressed in prior empirical research. This suggests that the world of ASFRs is richer and more diverse than what was previously documented.

The presence of these categories in our results can be explained as follows: All our participants design enterprise information systems, which in fact are reactive systems [97] with some ‘information provisioning’ function – a function to facilitate business processes to provide communication among actors involved in those processes. Knowing that the SAs design enterprise systems, the categories of ASFRs identified so far, in fact pertain to user functionality of the following types:

1. Providing memory (Audit Trail, Data-related Dialog, User Behaviour Analysis, Storage Mechanisms),
2. Providing information about memory for users (Print, Report, Search),
3. Facilitating processes in the user environment of the system (Batch Processing, Business Process “State” Alerts, Payment, Workflow, Licensing),
4. Facilitating or prompting communication among actors and/or systems (Third Party Interaction), and
5. Supporting functionality (that is needed to provide services to the user, such as Localization and Online Help).

Most ASFR categories were indicated by more than one participant from different domains. Therefore the ASFR categories we found seem to be applicable across the domains we studied. However, we expect that as we include more diverse domains such as information systems in other domains, safety critical systems or systems with real-time command and control requirements into our study, we would be in a much better position to classify ASFR categories as domain-agnostic versus domain-dependent. Therefore, more research into
these aspects is necessary if we are to understand the range of possible mechanisms for narrowing the gap between RE and software architecture.

We found that architects are skillful in spotting missing ASFRs or ill-specified ASFRs. As Participant P22 explained, in his practice he had the habit of red-flagging suspicious FRs that could potentially derail his software architecture design (for example, features that have very few users were routinely inspected for potential negative effects they might have on the software architecture):

“I also have other ‘red-alert requirements’. That is how we call requirements that are problematic for architecture. For example features that do not have many users. These get ignored until later when someone figures out that an exotic feature of a VP has not been considered for implementation”.

We therefore think it is important to explicitly augment the ASFRs with the architectural insights derived from PQ based conversations.

RQ 2: From a SA’s perspective, what kind of information is discoverable using PQs?

We found six types of PQs used by SAs to discern ASFRs. We found that PQs embed architectural knowledge and help greatly in deriving non-functional details relevant to specific FRs. This is close to what other researchers refer to as ‘scenarios’ in [5] or what are defined as ‘fit criteria’ in [98]. Reflecting on the six types, one might think these types are structured as the layers of an onion: system - project - business - society. Considering the system, the SA wants to know if ASFRs interact with each other (such as in PQ type 6) or with NFRs (such as in PQ type 3). Concerning the project, there may be financial or regulatory standards that impact an ASFR (such as in PQ type 5). The business may impose, or experience, constraints as a result of an architectural choice (such as in PQ type 1) and it may have strategic considerations that impact an ASFR (as in PQ type 2). It is important to question our interpretation; we may have misunderstood the PQs. However the fact that the interpretation assumes a generic onion-like layered structure makes us think that it is possible to observe our findings in other contexts where such a structure is present. That is, our evidence suggests that in other projects, architects would ask similar PQs to unearth the unspecified architectural details from the ASFR statements. Additional case studies will be needed to validate this.

RQ 3: How do architects use PQs to infer and explicate hidden architectural impact?

We found that domain knowledge is central to an SA’s judgment on what PQs to ask. This agrees with studies on the SAs’ professional skillset and qualifications [99], which indicate the criticality of the SA’s strong knowledge of the targeted problem domain. We also found that in some cases what was obvious to the client was not self-explanatory for the SAs. This confirms the previous findings that stakeholders’ tacit knowledge is often not well articulated
For example, when a client requested an SMS notification, the SA working on this project assumed that no email notifications would be included. The client found it hard to relate to the SA’s assumption and thought that sending email notifications was an “obvious requirement”. There are several plausible explanations for this. For example, in a large project where the client’s business representatives and the respective SA may be in two different countries, with different email usage levels and practices, a disconnect might arise regarding what is obvious and what is not. Domain knowledge alone seems insufficient in such cases, as it is also important to understand how the domain knowledge is situated in the practical context of the client and its organizational culture. This example also suggests that if BAs watch for “obvious requirements”, they could be of great service to the SAs and help prevent architectural rework.

We found that the SA’s knowledge about the nature of client’s business and the vendor’s processes for project delivery were instrumental for the generation of PQs. In fact, PQs were the SA’s way to approach the elicitation of ASFRs as a reflective conversation between clients/BAs and SAs. This finding is new in the sense that it has not been discussed so far in empirical studies on ASRs. However, we do not find it particularly surprising. It could well be explained by using Schön’s model [99] of consulting practice according to which consulting is about assisting clients to solve their organizations’ IT problems by the complementary use of consultant’s expertise and clients’ knowledge of their business in a process of framing and negotiating various perspectives. The act of asking PQs suggest that SAs act as experts who assist their clients in improving the precision of their FRs definitions. By asking PQs, SAs demonstrate their commitment to having their positions confronted and tested so that the risk of making sub-optimal architectural design choices is reduced. Moreover, each of our 6 types of PQs frames the ASFRs in a particular way. Following Schön, a consulting process that uses alternative ways to frame a problem (in this case the ASFR) is grounded on two types of consulting expertise: knowing-in-action and reflection-in-action. Knowing-in-action is defined as the consultants’ professional knowledge, their employer’s appreciative system, and frames of a particular community of practice (in this case, the SA’s community of practitioners), all of which serve as the source of coping strategies when solving problems in a consulting intervention. In addition, reflection-in-action [99] (p.125), means focusing on certain details of the problem while leaving others in the background, thus framing the problem in a particular way. If we view our SAs from the perspective of consultants embedded in communities of practitioners (i.e. SAs employed at a vendor’s organization and engaged in client’s projects), the PQs clearly are a coping strategy for solving SA-related problems (in this case, discerning architectural details from ASFRs).

4.6 Quantitative Evaluation Study

In order to validate our theory against actual SRS, we conducted a preliminary evaluation using real-life SRS documents. The goal of this first evaluation was to validate if the SRS documents in fact lack the needed architectural details to make informed architectural decisions. We took 450 FR statements from 30 real-life SRS documents (from the Insurance domain), using Castillo’s stratified sampling technique [101]. This is a probability sampling
technique wherein the researcher divides the entire population into different subgroups (or ‘strata’), then randomly selects the final subjects proportionally from the different strata. In our case, the strata were the projects and the lines of businesses (such as life, property and causality, health, marine etc.) within the insurance projects. Two of the researchers (the author of this dissertation and one of the supervisors - Ghaisas) studied the 450 FR statements and identified 199 ASFR statements from them. These 199 ASFR statements were then manually classified into the 15 ASFR categories presented in Section 4.4.1 above. However, in the studied SRS documents, we found FRs belonging to 10 out of these 15 categories. For each of those 10 categories, we compared the number of architectural details addressed in FR statements ($AD_{PRESENT}$) and the number of architectural details identified in the interview study ($AD_{IDENT}$). This was needed in order to see how many details occur in the sample.

Table 4.3 presents this quantitative comparison. We found that very few (0% to 33 %) architectural details were actually present in the ASFR statements in the SRS. This strengthens our working hypothesis that ASFRs do not explicitly state the architectural details. PQs are a necessary mechanism to unearth them.

<table>
<thead>
<tr>
<th>ASFR Category</th>
<th>FRs</th>
<th>$AD_{IDENT}$</th>
<th>$AD_{PRESENT}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business process ‘state’ alert</td>
<td>60</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Search</td>
<td>21</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>Print</td>
<td>24</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Audit Trail</td>
<td>21</td>
<td>11</td>
<td>2</td>
</tr>
<tr>
<td>Batch Processing</td>
<td>10</td>
<td>14</td>
<td>3</td>
</tr>
<tr>
<td>Localization</td>
<td>5</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>Report</td>
<td>7</td>
<td>19</td>
<td>5</td>
</tr>
<tr>
<td>Workflow</td>
<td>23</td>
<td>14</td>
<td>2</td>
</tr>
<tr>
<td>Payment</td>
<td>21</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Third party interaction</td>
<td>7</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

FRs: Number of functional requirements in the SRS

$AD_{IDENT}$:Number of architectural details identified from interviews

$AD_{PRESENT}$ :Number of architectural details addressed in FR statements
4.7 Validity Threats

There are three inherent weaknesses of interview techniques [96] that we acknowledge in this study. First is the extent to which the participant’s answered our questions truthfully. We minimized this threat by involving volunteers, under the assumption that if a practitioner cannot be honest, he/she could decline his/her participation at any stage of the research. Secondly, it is possible that the researcher might instil her bias in the data collection process. To counter this threat, we applied Yin’s recommendations [24] in this respect by (i) including participants working in diverse application domains, which allowed the same phenomenon to be evaluated from diverse perspectives (data triangulation [24]); (ii) sending the interview answers to each participant prior to data analysis to confirm the information he/she provided; and (iii) getting the data analysis report reviewed by some interviewees (some of whom provided feedback on clarifying the concepts, but this did not change the analysis). The third weakness is the possibility of asking leading questions - suggestive and implying that one answer might be better over others. We, however, made every attempt to leave our interviewees to respond on their own terms, expressing their own perspectives and values, and giving examples of what they do and how they cope in particular projects.

We used the checklist in [18, 97] to assess the possible threats to validity in this study. As it is exploratory, an important question regarding the validity of our results is: to what extent can our practitioners’ observations be considered representative for other project organizations? While not all settings in which the SAs infer hidden meanings from ASFRs are similar to the case study companies, some of them are [65, 26]. For instance, other vendors of large scale solutions that deliver enterprise information systems that serve clients in the business domains in which our SAs worked, and that must ensure compliance to business sector specific regulations, might experience phenomena similar to those in this case study. Moreover, our study is qualitative in nature and therefore we do not seek generalizability based on statistical representativeness, but based on similarity of contexts. Next, our quantitative evaluation (Section 4.6) found that 10 out of the 15 ASFR categories were present in the studied FR documents. We think the other 5 categories were not present because of less relevance to the domain of the projects included in the quantitative study (we included FR documents only from the insurance domain as these were easily accessible to us). However, further research is needed to uncover the contexts in which the remaining 5 categories would be more relevant.

4.8 Conclusion

The two exploratory studies presented in this chapter brings out how SAs cope with the process of inferring hidden meaning from ASFRs that are buried in large scale project documentation. We found 15 categories of ASFRs. We also found 6 types of PQs that are indispensable for revealing client-organization-specific and project-specific architectural details that have impact on the solutions’ architecture.
Our study has implication for RE research, practice, and teaching. We noticed that SAs’ PQs were grounded on their domain knowledge. An interesting line of research would be to automate the identification and classification of ASFRs and recommending relevant PQs to the BAs and architectural solutions to the SAs. For example, we can use machine learning techniques such as association rule mining to link the detected ASFR patterns with the right set of PQs that are necessary to be asked. We plan to draw on earlier experiences on automating the detection of artefacts from text [36, 37]. Moreover, the domain-dependent PQs can be linked to augment existing knowledge repositories in organizations [38]. In order to have an effective mechanism that is able to accurately identify ASFRs from FR documents, indicate impact possibilities comprehensively, and offer design solutions more pointedly, we need a collaborative mechanism wherein experts can contribute and curate requirements and architectural knowledge. We plan to harness our previous work for this purpose [38]. Further, it would be interesting to find out whether agile approaches would be comparatively more or less effective in revealing hidden ASFRs.

From RE practitioners’ perspective, knowing which ASFR categories have architectural impact can help BAs to elicit a more complete set of requirements. This provides the information that SAs need to make informed decisions and can potentially reduce wasted effort caused by the need to rework the solution later in the project.

Finally, it has been the observation of the author that RE textbooks often overlook the important synergies between RE and other SE activities such as software architecture. As more attention is paid to the RE-SA relationship, it is worthwhile to revise RE teaching by including concepts that help RE produce SRS that actually help SAs come up with suitable architecture that better fits the client’s business. In particular, our study suggests that complementing RE training with education on business rules (PQ type 1) might be a worthwhile option.
Creating Probing Question Flows

In this chapter we present the results of a study with over 40 experienced SAs to identify reusable PQs for ten areas of functionality and organize them into structured flows. These PQ-flows can be used by BAs to elicit and specify architecturally relevant information. This chapter is largely based on the publication [35].

5.1 Introduction

From the exploratory studies reported in chapter 4, we learnt that SAs ask PQs to gather information crucial to architectural decision-making. Our goal is to equip BAs with appropriate PQs so that they can ask these questions themselves. In this chapter, we explain how we systematically elicit, structure, and document these PQs in order to make them available for use by BAs. The goal is to leverage the knowledge of experienced SAs and to make it available to BAs so that they are equipped to elicit a more complete set of requirements that feed sufficient information into the architecture design process. While SE scholars agree that requirements elicitation and architectural design should be performed iteratively [10, 11, 12], little is known about what specifically BAs could or should do, in order to jump-start the process by identifying clients’ requirements in a way that helps SAs more effectively evaluate candidate architectural solutions.

To this end, we collected and analyzed the perceptions of more than 40 experienced SAs on the process of identifying, analyzing, organizing, and evaluating PQs for ten commonly occurring functional areas namely: Audit Trail, Batch Processing, Business Process ‘State’ Alerts, Report and Workflow, Localization/Multilingual, Search, Print, Online Help and Third Party Interaction. These areas were selected because (a) they occur commonly across many different systems, and (b) they emerged as important topics of architectural significance in the earlier study we had conducted [detailed in chapter 4]. The resulting PQ-flows can be used as catalogs by BAs in requirements elicitation. We note that the survey participants were from various multi-national companies in which BA and SA are separate disciplines and where the two sets of skills – requirements knowledge and architectural knowledge, typically reside with different people.

The study presented in this chapter is motivated by the findings from our previous qualitative interview-based study [33, 34] that we described in chapter 4. For the benefit of the reader,
we briefly summarize the studies from our previous chapter here. Our previous study [34] explored current practices for working with ASFRs as perceived by 24 SAs from large organizations in India, the United States and the Netherlands. The participating SAs were from the domains of Banking, Financial Services and Insurance (BFSI), Healthcare, Retail, Government, Telecommunications, and Airlines Management. Our key findings were that (1) SRSs often lack crucial architecturally relevant information needed to make informed architectural decisions, (2) PQs are an essential mechanism for unearthing underspecified architecturally relevant information. Our study identified a total of 15 categories of ASFRs, each with its own set of PQs.

Our study also suggested that there was a logical way to sequence the PQs during the elicitation process. We observed that experienced SAs understand what PQs need to be asked, have a tacit conceptual mental model of the interdependencies between the PQs, and therefore sequence their questions in a meaningful way. We note that similar observations were made by Li et al., who reported that much architectural knowledge is tacit in nature and should be recovered and documented [73]. If SAs’ tacit knowledge on the order of use of the PQs becomes explicit and shared with BAs, creating a repository of PQ-flows for each ASFR category could generate a body of knowledge similar to the Design Patterns of Gamma et al. [102] and be used by BAs to enhance the quality and completeness of SRSs. While the focus of our work is on ASFRs from the application domains mentioned earlier, we expect that PQ-flows could similarly be documented and made available for ASFRs from other areas.

5.2 Construction of PQ-flows

To discover and document PQ-flow structures for the ten selected ASFR categories, we followed a research process that was inspired by the design science approach [29]. The process for construction of PQ-flows in depicted in figure 5.1. We started with a ‘basic’ design of the PQ-flow and then integrated practitioners’ feedback into the design. This resulted in a refined version of the PQ-flows for which further feedback was elicited. The process included the following steps:

1. Interview SAs,
2. Analyze the PQs obtained through the interviews and create the initial basic PQ-flows in consultation with a SA,
3. Conduct an online survey (henceforth referred to as Phase 1 of the survey) to collect feedback on the early PQ-flows that were created in step 2, refine the flows, and
4. Conduct a follow up online survey (henceforth referred to as Phase 2 of the survey) to evaluate the refined PQ-flows from step 3.
Steps 1 and 2 were helpful in designing the basic PQ-flows for the 10 ASFR categories, while steps 3 and 4 were feedback-collection exercises which helped to refine the basic designs and to remove any bias that might have been passed into them. We now explain each of these steps in more detail.

5.2.1 Interview with the Software Architects

Our prior interviews with SAs [see the interviews described in Chapter 4] provided a basic unstructured list of PQs for each ASFR category. The author of this dissertation and one of her colleague therefore manually analyzed the PQs to identify a candidate logical sequence of questions. For instance, in case of Audit Trail category, we found that the PQ “What kind of notification mechanisms are needed to notify the end user about the analytics?” should be asked only if the client first answered “Yes” to the PQ “Should warning or alert messages of the analytics be sent to the end users?” Similarly, if the domain under consideration is Finance, it is not relevant to ask the PQ “Should reports be built over the audit?” This is because as per our interview participants “if the domain is finance, it is obvious that reports are needed”.

Once the basic PQ-flow structures for the ten ASFR categories were identified, we carried out a series of consultations with a senior SA who had more than 20 years of experience in Banking, Healthcare and Insurance domains. This was done to collect the SA’s feedback on the proposed PQ-flows. The consultation included a joint review of the basic PQ-flows for each ASFR category and a walk-through session in which the SA played the role of both client and SA and provided insights into the contextual settings under which he might ask a particular PQ as well as the effect of the client’s answers on the PQ-flow sequence. This exploratory exercise yielded ten refined PQ-flows that formed our initial PQ-flow catalog.

5.2.2 The Online Survey

We used the online survey method [103] to validate and improve the initial PQ-flow structure created in step 2. The survey was essential to reduce possible bias of the author and the senior SA. The use of a survey method enabled the inclusion of large number of participants.
which in turn enabled us to recognize trends in responses as the design of the PQ-flows was incrementally improved. The objective of our survey was to evaluate the initial PQ-flows, identify missing questions to improve the completeness of the PQ-flows, modify or remove invalid PQs, and improve their logical order. The number of questions in the online survey was kept as small as possible in an effort to encourage participants to complete the entire survey. The survey was implemented using Survey Monkey (https://www.surveymonkey.com/). The questionnaire is presented in Appendix B on page 125. We piloted the questionnaire with two SAs and integrated their feedback. The feedback was used primarily to improve user interactions (i.e. through improved instructions). As the survey questions were only slightly modified, we included the responses from the pilot study in the final analysis. The survey targeted professionals who have worked in the capacity of SAs in organizations ranging from multi-national companies to start-ups. As an incentive for participating in the study, we offered to share the final report of the results from our work. The questionnaire captured demographic information concerning the SA’s experience and primary business domain. As depicted in Table 5.1, over 73% of our participants had more than ten years of IT experience, while over 40% had at least five years’ experience as a SA. The primary business domains of the respondents included Energy, Resources and Utilities, Media and Information Services, Banking, Financial Services and Insurance (BFSI), Education, Content Management, Healthcare, Retail, Automotive, Telecommunication, Analytics, Recruitment and ERP for Manufacturing, Real estate, Hospitals. For each of the ten ASFR categories, the survey presented a diagram, similar to those shown in Figure 5.2 and 5.3 depicting the PQ-flows. The participants were then asked to evaluate the PQ-flow and to identify missing PQs (if any) and those that should be modified, deleted or re-sequenced. The participants’ responses were then used to refine the PQ-flow for each ASFR category. We would like to highlight that in [35] we have reported the results for 5 ASFR categories because while we were working on paper [35], we had analyzed the survey responses for only those 5 categories.

**TABLE 5.1 DEMOGRAPHIC DETAILS OF THE RESPONDENTS**

<table>
<thead>
<tr>
<th>TOTAL YEARS OF IT EXPERIENCE</th>
<th>TOTAL YEARS OF EXPERIENCE AS A SA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Choices</strong></td>
<td><strong>Response</strong></td>
</tr>
<tr>
<td>&lt; 1 year</td>
<td>0.00 %</td>
</tr>
<tr>
<td>1 to 5 years</td>
<td>10.53 %</td>
</tr>
<tr>
<td>5 to 10 years</td>
<td>15.79 %</td>
</tr>
<tr>
<td>&gt; 10 years</td>
<td>73.68 %</td>
</tr>
</tbody>
</table>
FIGURE 5.2 PQ-FLOW FOR AUDIT TRAIL

FIGURE 5.3 PQ-FLOW FOR BUSINESS PROCESS ‘STATE’ ALERT
5.2.2.1 Online Survey Phase 1 and Analysis

Phase 1 of the online survey was carried out over a period of 3 weeks (03rd July – 24th July 2015). We received a total of 22 responses. The survey was qualitative in nature and therefore once the survey responses were received, we performed qualitative data analysis [104] following the guidelines of Charmaz’ Constructive Grounded Theory method [32]. The data analysis was performed by three researchers (the author of this dissertation, and two of the authors’ colleagues in her organization). Based on the findings from our data analysis, we again refined the PQ-flows for each of the ASFR categories. This was done as follows: the three researchers analysed all the responses and then for each ASFR category, organized the PQs in sequence and logically appended them to the existing PQ-flows. While merging the feedback from different SAs, there were some minor cases of conflict such as cases where a missing PQ was specified by different SAs in different ways and/or at different levels of granularity. In such cases the author discussed with her colleagues and supervisors and chose the most appropriate PQ form to append. Also, there were several instances of agreements. For example, for the Workflow category, two of the Participants (P3 and P7) suggested we exclude the PQ: Do you want to have support for sub-workflow? They gave the following explanation:

P3: "How does a user know that they need a sub workflow? It is more of a technical implementation. This PQ can be removed."

P7: “A decade back, it might be an issue, but with many frameworks offering this as out-of-shelf, I don't think we should really focus on this."

We rejected some of the feedback comments for the reasons explained below.

1. Some suggestions for adding new PQs were not accepted because they pertained to design considerations rather than architecturally significant PQs. For example, for Report category, Participant P 12 suggested that we add the PQ: "What are the filters needed for the report?

2. Some suggestions for deleting PQs were not accepted either. For instance, Participant P16 suggested deleting several PQs from Audit Trail: "Delete PQ 12 through PQ 15 as these don’t strictly fall in audit trail category. Audit trails are typically “passive”. PQ 12 through 15 is venturing into monitoring and further into actionizing." PQ 12 through PQ 15 for Audit Trail talked about dynamically monitoring the data transactions/flows to detect abnormal patterns and notification to be sent for such patterns. However, Participant P4 mentioned the following for the same PQs: “I think PQ 12 through PQ 15 should be modified to include any analytics requirements that the client might have instead of only looking for abnormal patterns. The audited data can provide a mine of information”. Analysing these two feedback comments, we included the suggestion by P4. Instead of deleting PQ12 through PQ15, we modified them as per Participant P4’s suggestion.

3. Certain suggestions for adding new PQs were rejected because those PQs were already covered in the existing PQ-flows at different levels of granularity. We
attempted as much as possible to maintain similar level of granularity across all PQs. The modified PQ-flows were then evaluated in Phase 2 of the online survey, using the same questionnaire (Appendix B, page 125) as the one in Phase 1. This time, we showed the refined PQ-flows to an entirely different set of participants.

5.2.2.2 Online Survey Phase 2 and Analysis – Enriching the PQ-flows

Phase 2 of the online survey was conducted over a period of 2 weeks (31st July – 14th August 2015). The responses were analysed using the same qualitative analysis techniques used in Phase 1. We received 10 responses in Phase 2. This time, the analysis of responses indicated very few requests for modifications, additions, deletions, or re-organization of the PQ-flows. This suggests that the existing catalogue of PQ-flows adequately covered most of the important architectural concerns of the new set of participants.

Figure 5.2 depicts the PQ-flows for Audit Trail and figure 5.3 for Business Process ‘State’ Alert. Table 5.2 depicts the count for the number of PQs added to, edited and deleted from the original PQ list (before the beginning of Phase 1 of the survey) for each of the five ASFR categories at the completion of Phase 2 of the survey. The catalogue of all the ten PQ-flows is presented in Appendix C on page 129.

<table>
<thead>
<tr>
<th>Category</th>
<th>No. of PQs added</th>
<th>No. of PQs edited</th>
<th>No. of PQs deleted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audit Trail</td>
<td>7</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Batch Processing</td>
<td>4</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>BPSA</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Report</td>
<td>12</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Workflow</td>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Localization/Multilingual</td>
<td>5</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Search</td>
<td>4</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Print</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Online Help</td>
<td>6</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Third Party Interaction</td>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
5.3 Threats to Validity

We used multiple research techniques (i.e. interviews and online survey) for construction of PQ-flows. We evaluated the threats to validity in this multi-method research by using the checklist in [29]. As our design of PQ-flows is exploratory in nature and relies on refining the PQ-flows at each phase by collecting practitioners’ feedback, the generalizability of our PQ-flow structures is our most important concern. To what extent can the observations of the interview study participants and the two surveys’ participants be considered representative for other project organizations? We cannot claim universal generalizability of the PQ-flows that we derived by using our participants’ responses. However, it is reasonable to expect similar experiences in projects and organizations working in the application domains in which our practitioners work. For example, other organizations in sectors that must ensure compliance of their IT-solutions to business sector-specific regulations might have observations similar to those in this case study. More research is needed to collect evidence for or against this generalization.

We are also aware of the inherent uncertainty of interview techniques and of survey methods [29]. First uncertainty is about the extent to which the participant’s answered our questions truthfully. As in the previous chapter [section 4.7, page 52], we minimized this uncertainty by involving volunteers, so that they respond to our survey out of their free will and without pressure. Second, the author might have included her bias in the data collection and analysis process. To reduce this threat, we used Yin’s recommendations [24]. For example, we ensured that our survey respondents are drawn from diverse application domains, which let us evaluate the PQ-flows from diverse perspectives. Additionally, we had the data analysis report reviewed by some of the survey respondents (some of whom provided feedback on clarifying the concepts, but this did not change the analysis).

5.4 Conclusion

In this chapter we present the results from a study with experienced SAs to identify, document, and organize PQs for ten different areas of functionality into structured flows. The PQ-flows are aimed at assisting the BAs in asking the right questions to the clients so as to extract details needed by the SAs to make informed architectural decisions.

The work presented in this chapter has implications for both practice and research. First, we found an inexpensive down-to-earth solution that seems effective for enriching SRSs with architecturally relevant information. Using this approach, BAs can produce richer specifications that contain the details that SAs need for making architectural decisions. This could lead to less communication cycles between SAs and clients as there would be little need for follow-up clarifications in later project stages concerning ASFRs. Second, our approach can facilitate creation of a repository of PQ-flows for each ASFR category which could generate a body of knowledge similar to the Design Patterns of Gamma et al. [102] which could be used by the BAs to enhance the quality and completeness of SRSs.
Future research directions could involve more rigorous studies of PQ-flows in practice, and exploring the use of PQ-flows for emergent domains such as Internet of Things wherein the architectural landscape is still evolving and regulations are nascent at best.
6
Contextualizing PQ-flows

In chapter 5 we presented the results of a study with SAs to identify reusable PQs for ten areas of functionality and organize them into structured flows. In this chapter, we focus on the application of a specific PQ-flow within a project i.e. we present the automated approach we have developed for determining the relevance of a PQ-flow for a SRS, and automated annotation of relevant PQs with information found in the SRS. This chapter is largely based on the publications [33, 35]. Therein, we first present our initial experiments on automated identification, classification of ASFRs and annotating them with relevant information from the SRS. We next performed two studies to evaluate the usefulness of the approach in situ wherein we collected feedback from BAs and SAs.

The goal of the experiments is only to check the feasibility of automating the contextualization of PQ-flows. We do not claim direct usage of the automation part in real life projects. We acknowledge that more experiments are needed to improve the accuracy of the algorithm.

6.1 Introduction

In chapter 5 we presented the creation of a catalogue of PQ-flows. In this chapter we focus on the application of a specific PQ-flow within a project. We have devised an automated support for determining relevance of a PQ-flow for a SRS, and automated annotation of relevant PQs with information found in the SRS. We would like to highlight that the technical implementation of the automated support was done by the junior colleagues of the author of this dissertation. The author of this dissertation guided the juniors.

We have two goals in this work:

Our first goal (see Goal 1 in Figure 6.1 on page 64) is to determine when a flow is relevant to the project i.e. determining if there is any evidence in the existing requirements suggesting that Audit Trail, Workflow, or any other ASFR category is relevant for a certain project context.

Our second goal (see Goal 2 in Figure 6.1) is to identify requirements that already contain answers sought by questions in the relevant PQ-flow (see Figures 5.2 and 5.3 on page 58 for examples of the questions pertaining to ASFRs Audit Trail and Business Process ‘State’ Alert respectively) and then to annotate the PQ-flow diagram accordingly (see Figure 6.2 on page 75 for annotated PQ-flow). The BAs can leverage this information to determine what is already known, what knowledge is missing, and what effect this would have on the flow of the PQs. For example, if the following requirement is included in the SRS: “For regulatory
compliance, an Audit Trail of all the transactions should be maintained", then there is a clear indication that the audit is for regulatory compliance. In this case, there would be no need to ask the PQ: “Is the audit for internal compliance or for regulatory compliance?”

We further evaluated our approach for its usefulness in situ. For this, we applied our approach on two studies with real life SRS and collected feedback from BAs and SAs working on the projects of our studies. Please not that this evaluation is for the contextualization approach (i.e. the automation part). The evaluation of the concept of PQs, PQ-flows and the KB is presented in detail in chapter 7.

6.2 Contextualizing the PQ-flows

As already indicated in Chapter 3 (The Related Work chapter), many techniques for identification, classification and mining various types of requirements have been proposed [36]. However, the focus of previously published techniques has primarily been on NFRs. Based on the expert interviews, we have confirmed that many requirements which are purely functional in nature have architectural impact and we focus on such requirements. While writing this dissertation, this is the first time the concept of PQ-flows and their contextualization is introduced.

Figure 6.1 depicts our overall approach at a high level for contextualizing the PQ-flows. It shows the initial step of scanning the SRS in order to identify ASFRs, recommending the use of relevant PQ-flows, and then annotating the PQ-flow diagram with relevant requirements which already contain details sought by the specific questions in the PQ-flow.

![Figure 6.1 Contextualizing the PQ-flow – High Level](image)

We opted for a two-tier process for identification and classification of ASFRs. In the first-tier, our algorithm classified a given statement as an ASFR or non-ASFR. In the second-tier, the statements identified as ASFRs were taken as input and were then classified into various ASFR categories. We selected a two-tier process because we had more than 7000 instances of non-ASFRs as compared to only around 1000 instances of ASFRs. Since we were interested only in
ASFRs, subjecting all the 7000 non-ASFR instances to our multi-label classifier would have been an overkill. Also, previous studies have shown that building a two-tier classifier ultimately leads to more accurate results in scenarios where the first task is a binary classification (ASFR or non-ASFR) task when compared to the second task which is much fine grained (in terms of different ASFR categories).

We evaluated the performance of the algorithms using the following four metrics: Recall, Precision, F-score and Specificity.

We define each of them here:

**Recall:** Recall measures the fraction of actual ASFR instances which are correctly assigned.

\[
\text{Recall} = \frac{|\{\text{Correctly classified instances}\} \cap \{\text{Classified instances}\}|}{|\{\text{Correctly classified instances}\}|}
\]

**Precision:** Precision measures the fraction of classified instances which are correct.

\[
\text{Precision} = \frac{|\{\text{Correctly classified instances}\} \cap \{\text{Classified instances}\}|}{|\{\text{Classified instances}\}|}
\]

**F-Score:** F-score is the harmonic mean of recall and precision.

\[
\text{F-score} = \frac{2 \times (\text{precision} \times \text{recall})}{\text{precision} + \text{recall}}
\]

**Specificity:** Specificity measures the extent to which the algorithm correctly rejected instances that were not of the specified category. A specificity close to 1 means that the algorithm correctly rejected almost all negative instances of a category. Specificity is also referred to as True Negative Rate.

\[
\text{Specificity} = \frac{\text{True Negative}}{\text{True Negative} + \text{False Positive}}
\]

The formal definitions of these metrics are provided in [123, 124].

In the following three subsections we discuss the three steps of Figure 6.1 in detail.
6.2.1 Automated Detection of ASFRs

As indicated in this chapter’s Introduction, the first goal is to determine those PQ-flows that are relevant for a given project context. To this end we have trained a classifier to recognize requirements related to each of the ten ASFR categories. If such requirements are found in a SRS, then we assume the corresponding PQ-flow is relevant. We first performed a preliminary proof-of-concept experiment with 450 FRs. We next performed improved experiment on a larger dataset comprising of a total of 8669 requirements. In the sub sections that follow, we present details of each of these experiment.

6.2.1.1 Preliminary Experiment

The preliminary proof-of-concept experiment that we conducted is reported in [33]. In this experiment, we trained a classifier to detect 10 different categories of ASFRs. The classifier was trained using 450 FRs including a total of 246 ASFRs dispersed across 10 categories (i.e. Audit Trail, Batch Processing, Business Process ‘State’ Alert (BPSA), Localization/Multilingual, Payment, Print, Report, Search, Third Party Interaction (TPI) and Workflow). We reported our results with Naïve Bayes Classifier (NB) [108] as it outperformed the other classifiers (Decision Tree [109], Support Vector Machine [110], Nearest Neighbor [111] and Meta classifier – AdaBoost [112]) in the case of our dataset and the optimization parameters chosen. The NB classifier is a probabilistic classifier based on Bayes theorem and follows the assumption that the contribution of an individual feature towards deciding the probability of a particular class is independent of other features in that dataset instance. For example, in the case of detecting an ASFR class, the contribution of the term ‘log’ is independent of the contribution of the term ‘notification’. NB tends to overestimate the probability of the selected class because it ignores any dependencies among features. However, since we use it only to predict class membership and not the probability of class membership, overestimation of this probability does not have a large impact on inaccuracy.

As mentioned earlier, the dataset in our proof-of-concept experiment comprises 450 FR statements selected from 30 requirements specification documents (from the Insurance domain), by using the random sampling technique [113]. As a part of our supervised learning mechanism, two of the researchers (the author of this dissertation and one of her colleagues) manually analysed the 450 FR statements to label each statement as an ASFR or a non-ASFR. Further, each ASFR was manually tagged into one of the 10 ASFR categories. This formed the answer set against which the identification and classification results could be compared. The manual identification and classification took approximately 7.5 hours of each person’s time. We used 10-fold cross-validation [114] for training and testing using random stratified sampling. We used our own custom code for dividing the dataset into ten equal buckets to ensure that each of the bucket contained an approximately representative percentage of instances as present in the total dataset. During each iteration, a single bucket was treated as test data and the classifier was trained on the remaining nine buckets. Ten such iterations were performed until each bucket had been tested.
ASFR Identification

In this phase, each of the 450 FR statements in the dataset is classified into either an ASFR statement or a non-ASFR statement. We used WEKA (Waikato Environment for Knowledge Analysis) data mining software [115] for pre-processing the statements. This involves removing stop words that do not provide any relevant information on the statement’s lexical content (for example conjunctions and prepositions). The statements are then reduced into a set of keywords. The keywords are reduced to their stemmed form using Lovin’s stemming algorithm [116], to reduce all the words with the same root (or, if prefixes are left untouched, the same stem) to a common form, usually by stripping each word of its derivational and inflectional suffixes. The resultant keywords form the feature attributes. We used Information gain [117] to rank these features and retrieve the top 40 feature attribute. Information gain is a measure of the reduction in entropy of the class variable after the value for the feature is observed. In other words, information gain for classification is a measure of how common a feature is in a particular class compared to how common it is in all other classes. Examples of top ranked feature attributes include batch, search, communication, language, approval, print, email, workflow, audit, acknowledgment, payment. These 40 feature attributes are then used as an input to train the NB classifier. The NB probability model derived from these 40 attributes is then used for testing. The output of the testing phase involves flagging each FR statement as an ASFR or a non-ASFR. This training and testing mechanism is performed for the entire dataset by 10-fold cross validation. For ASFR identification, we achieved a precision of 0.77 and a recall of 0.81.

ASFR Classification

The second task involved classifying the identified ASFRs into specific categories. Classification at this level of granularity is necessary for determining which PQ-flows are relevant to a given project, and as a precursor for attaching PQs to a given ASFR. For ASFR classification, the ASFRs obtained from the ASFR identification step are taken. The mechanisms for pre-processing, training and testing are the same as those explained for ASFR identification above. We used a Multinomial Naïve Bayes classifier [118] for ASFR classification. Table 6.1 reports the two metrics of recall and precision for each ASFR category individually.
TABLE 6.1: CLASSIFICATION RESULTS FOR EACH ASFR CATEGORY

<table>
<thead>
<tr>
<th>ASFR Category</th>
<th>Requirements Count</th>
<th>Precision (%)</th>
<th>Recall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audit trail</td>
<td>21</td>
<td>0.67</td>
<td>0.48</td>
</tr>
<tr>
<td>Batch processing</td>
<td>10</td>
<td>0.60</td>
<td>0.60</td>
</tr>
<tr>
<td>BPSA</td>
<td>60</td>
<td>0.63</td>
<td>0.92</td>
</tr>
<tr>
<td>Localization/Multilingual</td>
<td>10</td>
<td>1.00</td>
<td>0.90</td>
</tr>
<tr>
<td>Payment</td>
<td>21</td>
<td>0.60</td>
<td>0.57</td>
</tr>
<tr>
<td>Print</td>
<td>24</td>
<td>0.75</td>
<td>0.50</td>
</tr>
<tr>
<td>Report</td>
<td>7</td>
<td>0.50</td>
<td>0.29</td>
</tr>
<tr>
<td>Search</td>
<td>21</td>
<td>0.58</td>
<td>0.52</td>
</tr>
<tr>
<td>Third party interaction</td>
<td>7</td>
<td>1.00</td>
<td>0.57</td>
</tr>
<tr>
<td>Workflow</td>
<td>23</td>
<td>0.79</td>
<td>0.65</td>
</tr>
</tbody>
</table>

Table 6.2 depicts a confusion matrix for the classification results. The correct classifications (true positives) are depicted on the diagonal, and have been highlighted in the table. It should be noted that we have considered the original 450 requirements statement for precision and recall calculation. Also, the 450 FR statements did not contain ASFRs of category ‘Online help’ and ‘Licensing’.

The best results were achieved for ‘Localization’ category of requirements, for which we achieved 0.90 recall at 1.00 precision. Overall, recall ranged from 0.28 to 0.90 and precision from 0.50 to 1.00. While from these results it is evident that there is much room for improvement, these classification results provide initial evidence that the ASFR categories identified by our experts can be automatically classified. While analysing the results, we found that ‘Report’ category had the lowest recall (0.29%) because we had very few statements in our dataset belonging to the category ‘Report’ (only 7 statements). However, prior work has shown that increasing the size of the training set can improve the accuracy of requirements classification results [36]. Furthermore, we observed that this approach suffered from the major limitation that each requirement was classified into exactly one class, whereas approximately 20% of requirements in the studied SRS were manually classified as belonging to two or more classes. For example, the requirement stating that the “system should generate reports on the complaint register on a monthly basis through a batch process” clearly provides information about both Reporting and Batch Processing. To address these issues, we retrained the classifier on a far larger number of requirements and utilized a
classifier which supports multi-label classification. In the following sub-section we present results from the experiments we did on the larger dataset.

**TABLE 6.2: CONFUSION MATRIX DEPICTING CLASSIFICATION RESULTS**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>K</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>55</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>12</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>2</td>
<td>12</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>2</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>15</td>
<td>23</td>
<td></td>
</tr>
</tbody>
</table>

Classified as:
- A = Audit trail
- B = Batch processing
- C = Localization/Multilingual
- D = BPSA
- E = Payment
- F = Print
- G = Report
- H = Search
- I = Third party interaction
- J = Workflow

**6.2.1.2 Training on a Larger Set of ASFRs**

To increase the reliability of our approach and to scale it up for use in large industrial projects, we retrained the classifier against 8,669 requirement statements dispersed across 114 distinct SRSs from the insurance domain.

Three of the researchers (the author of this dissertation and two of her colleagues) manually analyzed the 8,669 requirement statements to label each statement as an ASFR or a non-ASFR and then classify each ASFR statement into one or more of the ASFR categories. Each researcher individually reviewed and classified a third of the requirements. If there was any doubt about the class of a requirement, all three researchers discussed together to arrive at a consensus. Furthermore, the researchers periodically reviewed each other’s work to ensure ongoing agreement. The resulting dataset formed the ground truth against which the identification and classification results could be compared. The task took a total of 60 hours to complete i.e. on an average 20 hours per person.

Out of the 8,669 requirements, the counts attributed to each ASFR category ranged from 2 for Online Help to 424 for Localization/Multilingual. Since Online Help had negligible instances (i.e. only 2), we did not include it for the ASFR categorization step. Table 6.3 shows the requirements count for each of the ASFR category.
### Table 6.3 Requirements Count for Each ASFR Category

<table>
<thead>
<tr>
<th>ASFR Category</th>
<th>Requirements Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audit Trail</td>
<td>89</td>
</tr>
<tr>
<td>Batch Process</td>
<td>89</td>
</tr>
<tr>
<td>BPSA</td>
<td>246</td>
</tr>
<tr>
<td>Localization/Multilingual</td>
<td>424</td>
</tr>
<tr>
<td>Payment</td>
<td>26</td>
</tr>
<tr>
<td>Print</td>
<td>164</td>
</tr>
<tr>
<td>Report</td>
<td>90</td>
</tr>
<tr>
<td>Search</td>
<td>142</td>
</tr>
<tr>
<td>Storage Mechanism</td>
<td>41</td>
</tr>
<tr>
<td>TPI</td>
<td>281</td>
</tr>
<tr>
<td>Workflow</td>
<td>81</td>
</tr>
</tbody>
</table>

### ASFR Identification

As before, our classification approach includes two steps. As mentioned in Section 6.2.1.1, in the first step, we simply classify each requirement as ASFR or non-ASFR. In our preliminary experiment (Section 6.2.1.1), we found that Naïve Bayes (NB) performed better when compared with J48 decision tree, Support Vector Machine, Nearest Neighbor and AdaBoost, and this matches results observed in other studies related to classification problems [119]. We therefore adopted NB for the binary classification task.

Because instances of ASFRs and non-ASFRs were imbalanced (i.e. 1434 ASFRs and 7235 non-ASFRs), we adopted stratified random sub-sampling. The majority class (i.e. the non-ASFRs) was under-sampled in each iteration. A disadvantage associated with under-sampling is that it is known to result in loss of information [120]. To counter this, we repeated the experiment such that all non-ASFRs were included in the training dataset at least once. We used 5-fold cross validation to divide the data into five evenly sized buckets, such that instances of each ASFR and non-ASFR were distributed evenly across the buckets. In each iteration, one bucket was set aside for testing and the remaining four buckets were used to create a training set. The sub-sampling was applied to equalize the instances of ASFRs and non-ASFRs and then used to train the NB classifier. The trained classifier was then used to classify the requirements in the test set. This process was repeated 5 times, until all data had been classified. The cross validation was 5-fold because the number of instances of non-ASFRs is 5 times the number of instance of ASFRs in our dataset.
As in Section 6.2.1.1, all requirements were pre-processed in WEKA to remove stop words, and to stem each word to its morphological root using Lovins’ stemming algorithm. The resulting terms formed the feature attributes. In order to select feature attributes that would help in classification, we used the technique of Information Gain in WEKA to rank these features according to their relevance to classification. We then selected the top 100 feature attributes for our classification experiments.

With a larger training set, for the ASFR identification task, we achieved an average recall of 0.81 and an average precision of 0.69. When compared to the preliminary experiment (recall = 0.81, precision = 0.77), we observed that the recall remains the same while there is a drop in precision. This indicates that re-training (using a larger training set) was effective in allowing our approach to scale up to the scale of an industrial project.

ASFR Classification

In Section 6.2.1.1, we presented our preliminary experiment on classifying ASFRs using a smaller dataset. The classification had a limitation that it classified a statement exactly into one class. However, as an ASFR can belong to more than one class, in this step, we investigated classifiers which are capable of performing multi-label classification. In particular, we performed initial exploratory investigations with the RAndom k labELsets classifier (RAkEL) [121], NB Classifier [108] and Nearest Neighbor Classifier [111] using the Mulan software package [122]. Mulan has specialized algorithms to handle multi-label classification problems. In this dissertation, we report our results with RAkEL since it outperformed the other classifiers in the case of our dataset and the optimization parameters chosen. Moreover, during the manual analysis of our dataset, we discovered some label correlations in our dataset. For instance, labels ‘Workflow’ and ‘Third Party Interaction’ occur together for certain statements. RAkEL also incorporates such correlations during the classification task. RAkEL is an enhancement of the Label Powerset (LP) method that incorporates label correlations by considering each subset of labels (called labelsets) that exists in the training set, as a different class value in a single-label classification task. However, if the training set has a large number of labelsets, LP suffers from higher computational cost and the dataset can be skewed if some labelsets have limited training examples. Moreover, during the testing phase, LP can predict only the labelsets learned from the training set. RAkEL overcomes the limitations of LP by randomly breaking the initial set of labels into a number of small-sized labelsets and then training a multi-label classifier on the training set with those labelsets using LP. The k in RAkEL is a parameter for the size of the labelsets.

We present the results for each of the evaluated ASFR categories in Table 6.4, using four metrics: Recall, Precision, F-score and Specificity.
### TABLE 6.4 CLASSIFICATION RESULTS FOR EACH ASFR CATEGORY AT STATEMENT LEVEL

<table>
<thead>
<tr>
<th>ASFR Category</th>
<th>#Statements</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
<th>Specificity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audit Trail</td>
<td>89</td>
<td>0.81</td>
<td>0.85</td>
<td>0.82</td>
<td>0.98</td>
</tr>
<tr>
<td>Batch Process</td>
<td>89</td>
<td>0.90</td>
<td>0.96</td>
<td>0.93</td>
<td>0.99</td>
</tr>
<tr>
<td>BPSA</td>
<td>246</td>
<td>0.76</td>
<td>0.60</td>
<td>0.67</td>
<td>0.96</td>
</tr>
<tr>
<td>Localization/Multilingual</td>
<td>424</td>
<td>0.93</td>
<td>0.95</td>
<td>0.94</td>
<td>0.97</td>
</tr>
<tr>
<td>Payment</td>
<td>26</td>
<td>0.50</td>
<td>0.20</td>
<td>0.26</td>
<td>0.99</td>
</tr>
<tr>
<td>Print</td>
<td>164</td>
<td>0.90</td>
<td>0.96</td>
<td>0.93</td>
<td>0.98</td>
</tr>
<tr>
<td>Report</td>
<td>90</td>
<td>0.97</td>
<td>0.83</td>
<td>0.89</td>
<td>0.99</td>
</tr>
<tr>
<td>Search</td>
<td>142</td>
<td>0.95</td>
<td>0.88</td>
<td>0.91</td>
<td>0.99</td>
</tr>
<tr>
<td>Storage Mechanism</td>
<td>41</td>
<td>0.36</td>
<td>0.12</td>
<td>0.17</td>
<td>0.99</td>
</tr>
<tr>
<td>TPI</td>
<td>281</td>
<td>0.76</td>
<td>0.62</td>
<td>0.68</td>
<td>0.95</td>
</tr>
<tr>
<td>Workflow</td>
<td>81</td>
<td>0.78</td>
<td>0.52</td>
<td>0.58</td>
<td>0.99</td>
</tr>
<tr>
<td>AVERAGE</td>
<td></td>
<td><strong>0.79</strong></td>
<td><strong>0.69</strong></td>
<td><strong>0.71</strong></td>
<td><strong>0.98</strong></td>
</tr>
</tbody>
</table>

**Analysis of Results**

The results reported in Table 6.4 show that the greatest accuracy was observed for *Batch Processing*, *Print*, *Localization/ Multilingual* and *Search*, which all achieved F-Scores above 0.90. Overall, F-score values ranged from 0.17 to 0.94. A consistently good specificity of 0.95 and above for all the categories indicates that the classifier successfully avoided many misclassifications.

The lower accuracy for some categories such as *Storage Mechanism* and *Payment* can be attributed to the fact that they had very few instances (26 and 41 respectively). For some categories such as *BPSA* (*Business Process “State” Alert*) and *TPI* (*Third Party Interaction*), accuracy was also relatively low despite the fact that the categories were well represented at 246 and 281 respectively. We observed that requirements in these two categories tended to belong to a higher-than-average number of class associations. For instance, the requirement statement “There should be a monthly batch process that generates a report of premium defaulters and then triggers an email reminder for the defaulter policy holders. This requirement is only for Denmark” falls into 4 ASFR categories namely Batch Process, BPSI, Localization/Multilingual and Report. Our analysis of the results suggested that the results from the classifier were less accurate for such requirements.
In table 6.5 we show a comparison of the classification results (Precision, Recall and F-score values) for our preliminary experiment versus the experiment on larger data set. Since the ASFR category – *Storage Mechanism* was not present in the preliminary experiment, we exclude the same from this comparison.

<table>
<thead>
<tr>
<th>ASFR Category</th>
<th>Preliminary Experiment</th>
<th>Experiment on Larger Data Set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Statements</td>
<td>Precision (%)</td>
</tr>
<tr>
<td>Audit trail</td>
<td>21</td>
<td>0.67</td>
</tr>
<tr>
<td>Batch</td>
<td>10</td>
<td>0.60</td>
</tr>
<tr>
<td>BPSA</td>
<td>60</td>
<td>0.62</td>
</tr>
<tr>
<td>Localization</td>
<td>10</td>
<td>1.00</td>
</tr>
<tr>
<td>Payment</td>
<td>60</td>
<td>0.60</td>
</tr>
<tr>
<td>Print</td>
<td>21</td>
<td>0.75</td>
</tr>
<tr>
<td>Report</td>
<td>7</td>
<td>0.50</td>
</tr>
<tr>
<td>Search</td>
<td>21</td>
<td>0.58</td>
</tr>
<tr>
<td>Third party interaction</td>
<td>7</td>
<td>1.00</td>
</tr>
<tr>
<td>Workflow</td>
<td>23</td>
<td>0.79</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>0.71</strong></td>
<td><strong>0.60</strong></td>
</tr>
</tbody>
</table>

As is evident from table 6.5, our experiment with larger dataset achieved an average precision, recall and F-Score of 0.83, 0.74 and 0.76 respectively which is a good improvement on the earlier average precision, recall and F-score of 0.71, 0.60 and 0.67 respectively that we obtained in our preliminary experiment.

### 6.2.2 Recommending PQ-flows Relevant for an SRS Document

A specific PQ-flow is recommended to a BA for use in a project if, and only if, the current SRS contains one or more ASFRs of the respective category. For example, the Audit Trail PQ-flow would be recommended if at least one audit-related requirement is found in the SRS. We therefore analyzed the classification results at the SRS level. We refer to this approach as Relevance algorithm.
Based on the previously established ground truth and results from section 6.2.1.2, we determined whether a given PQ-flow was relevant to each one of the 114 SRS documents. We evaluated the relevance algorithm using Precision, Recall, Specificity, and F-score results, accordingly.

Table 6.6 depicts results from the relevance algorithm for each of the five ASFR categories in the 114 SRS documents. The results from our analysis show that we were able to quite accurately determine when to recommend a PQ-flow for use in a project. Precision ranged from 0.92 (Workflow) to 0.98 (BPSA) while recall ranged from 0.79 (Workflow) to 0.10 (Batch Processing). This indicates that the approach is capable of retrieving most of the relevant SRSs for the PQ-flows for the ASFR categories. Specificity ranged from 0.93 (BPSA) to 0.98 (Workflow), indicating that there were few cases in which we failed to recommend relevant PQ-flows, or made incorrect recommendations.

<table>
<thead>
<tr>
<th>Category</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Score</th>
<th>Specificity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audit Trail</td>
<td>35</td>
<td>33</td>
<td>32</td>
<td>0.97</td>
<td>0.91</td>
<td>0.98</td>
<td>0.94</td>
</tr>
<tr>
<td>Batch Processing</td>
<td>28</td>
<td>29</td>
<td>28</td>
<td>0.97</td>
<td>1.00</td>
<td>0.99</td>
<td>0.98</td>
</tr>
<tr>
<td>BPSA</td>
<td>59</td>
<td>53</td>
<td>52</td>
<td>0.98</td>
<td>0.88</td>
<td>0.97</td>
<td>0.93</td>
</tr>
<tr>
<td>Report</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>0.96</td>
<td>0.92</td>
<td>0.99</td>
<td>0.94</td>
</tr>
<tr>
<td>Workflow</td>
<td>14</td>
<td>12</td>
<td>11</td>
<td>0.92</td>
<td>0.79</td>
<td>0.99</td>
<td>0.85</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td></td>
<td>0.96</td>
<td>0.90</td>
<td>0.98</td>
<td>0.92</td>
</tr>
</tbody>
</table>

### 6.2.3 Annotating the PQ-Flows

Once relevant PQ-flows are identified by the relevance algorithm, the BA can follow the flow of questions in order to elicit architecturally significant information. However, a naïve use of the PQ-flow could lead the BA to asking questions which are already addressed in the SRS. In this phase of our approach we therefore leverage text mining techniques to identify existing requirements which could be associated with various nodes in the PQ-flow. For illustrative purposes, in Figure 6.2, we present the Batch Processing PQ-flow with requirements automatically retrieved from one of the SRS and associated with a specific PQ. Some of the example requirements in Figure 6.2 are obfuscated for confidentiality reasons. In this section we describe the techniques we used to associate requirements with specific PQs. It is worth noting that this task is even more challenging than classifying ASFRs by category. In effect, it represents an even more fine-grained task in which specific ASFRs are associated with individual PQs in the PQ-flows. For this task, we used the standard Vector Space Model (VSM) [29] to compute the similarity between a PQ and each requirement in a given SRS. The process is repeated for all the PQs in a PQ-flow for each ASFR category.
FIGURE 6.2 ANNOTATED PQ FLOW FOR BATCH PROCESSING

**Batch Processing**

Process starts when the batch is received and an effort is made to process it as efficiently as possible. The batch is divided into smaller units that are processed in a sequential manner.

1. **Receiving and Staging**
   - The batch is received and staged for processing.
   - The batch is divided into smaller units for processing.
   - The units are processed in a sequential manner.

2. **Data Extraction**
   - The data is extracted from the units.
   - The data is formatted and organized for processing.
   - The data is validated for accuracy and completeness.

3. **Data Processing**
   - The data is processed through various stages of the process.
   - The data is updated and consolidated.
   - The data is stored for future reference.

4. **Quality Control**
   - The data is reviewed for quality and accuracy.
   - The data is approved for further processing.

5. **Output Generation**
   - The output is generated from the processed data.
   - The output is formatted and organized for distribution.

6. **Distribution and Reporting**
   - The output is distributed to the relevant parties.
   - The output is reported to the management for review.

7. **Closure**
   - The process is completed and the batch is closed.
   - The batch is archived for future reference.

**Notes**

- Fig. 6.2: PQ flow for batch processing.
- The flowchart shows the sequence of operations involved in processing a batch of data.
- The flowchart is annotated with the relevant steps and notes for better understanding.

**Abbreviations**

- PQ: Process Quality
- Batch: A collection of data that is processed as a single unit.
For similarity score computation, VSM uses the following formula:

\[
\text{Sim}(\text{ASFR}_i, PQ) = \frac{\text{ASFR}_i \cdot PQ}{\|\text{ASFR}_i\| \|PQ\|} = \frac{\sum_{j=1}^{N} w_{j,a} w_{j,p}}{\sqrt{\sum_{j=1}^{N} w_{j,a}^2} \sqrt{\sum_{j=1}^{N} w_{j,p}^2}}
\]

Where, 'N' represents the total number of unique words, 'a' represents ASFR statement, 'p' represents probing question, \(w_{j,a}\) represents the list of frequencies of words present in \(i\)\(^{th}\) index of the ASFR statement and \(w_{j,p}\) represents the list of frequencies of words present in PQs. A cosine similarity of 1 indicates the closest match and a cosine similarity of 0 means that the two strings are totally different. We acknowledge that this is a weak similarity measure as this uses only the frequency of words to compute similarity. However, as mentioned in the beginning of this chapter, our intention was to check if it is feasible to automate the annotation of PQ-flows. The results endorse that it is feasible and in our future work we will explore better techniques to achieve this.

### 6.2.3.1 Analysis of Results for Annotating PQ-flows

To evaluate the results, three of the researchers (the author of this dissertation and two of her colleagues) manually analysed all possible associations between PQ and ASFR statements. Each researcher individually reviewed all the associations. We periodically reviewed each other’s work to ensure ongoing agreement about the association. The resulting dataset formed the answer set against which the association results could be compared. The task took a total of 80 hours to complete. Results are reported in Table 6.7 using the standard metrics: Precision (P), Recall (R), F-score, and Specificity (S).

<table>
<thead>
<tr>
<th>Category</th>
<th>#PQs</th>
<th>#ASFR Count</th>
<th>Potential associations</th>
<th>Relevant associations</th>
<th>Retrieved associations</th>
<th>Retrieved &amp; relevant associations</th>
<th>P</th>
<th>R</th>
<th>F-score</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audit Trail</td>
<td>16</td>
<td>89</td>
<td>1376</td>
<td>22</td>
<td>68</td>
<td>18</td>
<td>0.27</td>
<td>0.82</td>
<td>0.40</td>
<td>0.96</td>
</tr>
<tr>
<td>Batch Processing</td>
<td>15</td>
<td>90</td>
<td>1350</td>
<td>207</td>
<td>674</td>
<td>162</td>
<td>0.24</td>
<td>0.78</td>
<td>0.37</td>
<td>0.55</td>
</tr>
<tr>
<td>BPSA</td>
<td>12</td>
<td>246</td>
<td>2952</td>
<td>90</td>
<td>434</td>
<td>52</td>
<td>0.12</td>
<td>0.58</td>
<td>0.20</td>
<td>0.87</td>
</tr>
<tr>
<td>Report</td>
<td>22</td>
<td>90</td>
<td>1980</td>
<td>63</td>
<td>609</td>
<td>49</td>
<td>0.08</td>
<td>0.78</td>
<td>0.15</td>
<td>0.71</td>
</tr>
<tr>
<td>Workflow</td>
<td>18</td>
<td>84</td>
<td>1512</td>
<td>52</td>
<td>59</td>
<td>31</td>
<td>0.58</td>
<td>0.65</td>
<td>0.61</td>
<td>0.98</td>
</tr>
</tbody>
</table>

On an average, the precision was 0.28 while the recall was above 0.70 indicating that our approach was able to retrieve many of the relevant statements for annotation. The lowest recall was observed in case of BPSA (0.58). For BPSA, we observed that VSM picked up comparatively more irrelevant terms for similarity matching. Further, the PQs we used for our experiments were taken verbatim from the SAs. However, our analysis revealed that
rephrasing the PQs could improve the results further. For example, a PQ for BPSA “What are the roles that can configure the categories/types?”, if rephrased to make it more precise, such as “Who are the stakeholders (roles) that can configure the categories/types of the alert message?” would help in retrieving more relevant statements. This can be a direction of future work.

6.3 Evaluating the PQ-Flows

As indicated in the Introduction section of this chapter (pp. 63-64), we performed two preliminary studies to evaluate the usefulness of the contextualization approach in situ. By doing so, we wanted to get a preliminary feedback from real professionals and learn from their opinion. Each study involved a BA, a pseudo-client and a SA. In both studies, the BA and SA were experienced team members responsible for delivering large projects in the Insurance and Banking domains. The pseudo-client had experience working closely with clients to understand their requirements and to architect systems. In his professional carrier of 20 years, he had served in the roles of both BA and SA in different organizations.

Study 1 involved the SRS document from an ongoing insurance project containing a total of 226 requirements, while Study 2 was based on the SRS from an ongoing banking project containing a total of 2855 requirements. We ran the Relevance algorithm (see section 6.2.2) for each of the five ASFR categories against the two SRSs to find out the ASFR categories relevant to that SRS. For the SRS in Study 1, the relevance algorithm identified 4 requirements of Audit Trail and 2 requirements of Business Process ‘State’ Alert. For the SRS in Study 2, the relevance algorithm identified 17 requirements of Report, 22 of Business Process ‘State’ Alert, 1 of Workflow and 6 of Batch Processing. We then shared the relevant ASFR categories (their name, definition and an example) with the participants of Study 1 and Study 2 respectively and asked them to select the category they were most experienced and comfortable with. The participants of Study 1 chose Audit Trail while the participants of Study 2 chose Report.

In each of the studies, we ran VSM to analyse the respective SRS in order to generate annotated PQ-flows for Audit Trail and Report (as explained in section 6.2.3). For Audit Trail, VSM annotated seven of the 16 PQs while for Report, VSM annotated 21 of the 22 PQs. We manually analysed these annotations and found the following:

For the SRS in Study 1 (Audit Trail), seven of the PQs were annotated. Of these, PQs were fully answered in two cases, while meaningful information was provided for three additional PQs which enabled better or more informed questions to be asked. The remaining two cases represented false positive annotations. Further, we manually analysed the SRS and found that there was no additional relevant information in the SRS that should have been used to annotate the PQs.

For the SRS in Study 2 (Report), out of the 21 annotated PQs, three PQs were fully answered, while meaningful information, which enabled better or more informed questions to be asked, was provided for two additional PQs. The remaining 16 cases represented PQs with false positive annotations. These results are not surprising given the relatively low precision reported for Report annotations in Table 6.7. Further, we manually analysed the SRS and
found that there were two additional requirements in the SRS that should have been included in the annotation but were excluded.

We provided the automatically annotated PQ-flows to the respective BAs of the two studies. For each study, the respective BA was asked to follow the PQ-flow and accordingly ask PQs to the pseudo-client and record the client’s response. In the flow, if a PQ was annotated (indicating that either an answer to that PQ is already present in the SRS or the SRS contains some information related to that PQ), the BA was asked to review the annotation and decide whether to skip that PQ (in case it completely answers the PQ) or ask a better or more informed question (in case the SRS contains some related information).

Based on the responses received from the pseudo-client, the BA was asked to augment the original SRS with the answers received for the PQs. For Audit Trail, 13 new requirements were added and for Report, 15 new requirements were added.

After the completion of this activity, in each study, the BA was sent a questionnaire to gather feedback on the applicability and usefulness of the PQ-flow. Further, we shared the original and the augmented version of the SRS with the SA and asked them to evaluate the usefulness of the additional requirements. We summarize part of the feedback below. The questionnaire for the BA and the SA can be viewed at Appendix D on page 137.

It should be noted that (1) the author of this dissertation and one of her supervisors explained the approach and the evaluation study process to the study participants but neither of them were present when the actual evaluation study was conducted, (2) the participants for the evaluation study were selected on the basis of their domain knowledge and project management experience endorsed by their supervisors who recommended them to us.

### 6.3.1 Feedback from the BAs

To our question on the applicability of the PQ-flows in the requirements elicitation process, the BA from Study 1 stated:

“It (the PQ-flow) fits well into the elicitation process as it covered the key architectural concerns that a software architect will have. I as a BA am not able to elicit them on my own due to lack of expertise in the architectural domain. So this is a really good approach”.

To our question on the usefulness of the hierarchical structure of the PQ-flows (versus just having a collection), the BA from Study 1 stated:

“Yes, having a hierarchical structure really helps in specifying each problem area. For example, having a flow structure to the data audit aspect helps clarify in stages, the volume of the data, its type and output. Having it as a collection instead could possibly lead to several overlapping or confounding responses which could potentially require follow-up questions”

and the BA from Study 2 said:

“PQs that depended on other PQs were well represented in hierarchical structure. This helped in traversing to the right question without wasting much of time in thinking what to ask next?”
To our question on the extent to which the BA could imagine integrating the information acquired through using the PQ-flows into the SRS, the BA from Study 2 said:

“I would think that a good fit would exist between the information gained in the PQ-flows and the requirements specification. It would make the requirements specification much richer in terms of architectural knowledge.”

When asked whether they would use PQ-flows in future projects, the BA from Study 1 said:

“PQ-flows offer a sound structured framework to collect information and therefore I would definitely like to use this approach in future projects” and the BA from Study 2 said “I would think that this approach would apply to any data-intensive project, which is any typical project. PQ-flows offers a sound structured framework to collect information”.

Further, on the question on the change in the order of PQs the BA would like to see to make the PQs more helpful to them, both the BAs responded saying that the current order of the PQs is “good enough”.

6.3.2 Feedback from the SA

To our question on the value seen by the SA on using this approach, the SA from Study 1 said:

“I think this approach is quite exhaustive to make client as well as vendor think in depth before requirements are put forth and are understood clearly. It would avoid the SA make any assumptions about unstated or unclear requirements and also will save us a lot of time that gets wasted going back and forth to the client for clarifications”.

The SA from Study 2 said:

“I think, this approach shall help us to ensure that the client does not miss out key features that he may not think otherwise. The flow shall serve as a template for similar requirements. The client shall view us as a good process oriented and experienced organization”.

When asked about the kind of effort needed to get this level of architectural detail in the absence of such an approach, the SA from Study 1 said:

“Effort would be quite high and more than that user acceptance testing will raise questions as to why things were not clarified earlier in requirements phase. Series of change requests will follow after implementation.”

Further, the SA from Study 2 mentioned:

“Yes, this approach will help very much in bridging the gaps that exists between requirements and architecture”.

On the question “Do you feel a BA equipped to ask these questions is useful or would you prefer asking these questions on your own”, the SA from Study 1 said:
“The BA would need some amount of training which looks very much doable to me. Once trained, this approach would definitely be very useful and would save a lot of time which we spend going back to the BA or client for clarifications”.

The SA from Study 2 said:

“I am not sure the knowledge BA will have in terms of domain, overall understanding of client process flow as well as the organizational structure with roles and responsibilities. If well equipped with this then can carry out such exercise and would be very useful. An important aspect that is required in an individual who does this job is to do lateral thinking and should be analytical in nature”.

Further when asked, if such an approach helps in bridging the gap that exist between requirements and architecture, the SA from Study 1 said “Yes, very much!!”

The feedback we received from the BAs and SAs indicate that our approach seems applicable and useful in real world IT project settings.

6.4 Threats to Validity

Regarding the evaluation of the classification techniques chosen in the tool design, we used a large set of SRSs (114), comprising over 8000 statements. Although this cannot serve for forming universal claims, it injects realism [31] in our analysis as the SRSs have been delivered in large real-life industrial projects with elaborate complex functionalities delivered by team sizes greater than 100 to hundreds of thousands of users.

The feedback from the BA and SA about the utility of our approach could be ascribed to the ‘good participant role’ in which the participant attempts to discern the experimenter’s hypotheses and to confirm them [125]. The participant does not want to “ruin” the experiment. This threat cannot be eliminated, but our future plan of conducting a technical action research should mitigate this risk. Another threat is the possibility of asking leading questions i.e. Question(s) deliberately designed to make the respondents think in a certain way. We mitigated this threat by staying away when the actual evaluation study was conducted. In this way we ensured that we did not bias the process or outcome in any way. Next threat is to ask whether the feedback would be different if we interview a different set of BAs and SAs. We cannot claim generalizability of the responses that we received from the validation study participants. However, it is reasonable to repeat this experiment with other subjects, in similarly realistic circumstances, because these initial results are promising. However, we acknowledge that more research is needed to collect evidence for or against this generalization.

6.5 Conclusion

In this chapter we presented the machine learning techniques that we deployed to contextualize the use of the PQ-flows and to guide BAs through the process of asking pertinent questions that elicit architecturally significant details. The BAs can leverage this
information to determine what is already known, what knowledge is missing, and what effect the missing knowledge would have on the flow of the PQs. Results from our initial study showed that the use of PQ-flows by BAs can help to unearth many architectural details and can lead to more complete and comprehensive SRSs enriched with architectural information. The study also suggested that BAs would need initial training to effectively use this approach.

We are in the process of conducting experiments on larger and more varied datasets to evaluate the scalability of our approach and to improve its accuracy. However, these experiments are outside the scope of this dissertation. On the basis on our current experiments, we note that the training process took 140 hours, but once trained, it took us 13-15 minutes to process a large SRS containing 8669 individual requirements. We plan to further address performance as we transition our approach to industrial practice.
Empirical Evaluation of the Approach

This chapter presents the design, the execution and the results of a systematic empirical evaluation of our approach. In particular, our goal is to investigate four aspects namely the ease of use, effectiveness, relevance and generalizability of our approach. Our evaluation study found out that the ease of use of our approach can be improved by introducing some form of a training module. Regarding the relevance of our approach, we observed that the SAs found our approach to be relevant. However, the relevance of the existing ASFR categories and PQ-flows is heavily dependent on factors such as the project domain, geography and technology.

7.1 Introduction

In chapter 4, 5 and 6, we presented our approach that assists BAs in asking architecturally relevant questions to the client and produce an architecturally richer specification. We explained the process of arriving at the 15 categories of ASFRs (Chapter 4) and their corresponding PQ-flows (Chapter 5). Drawing on methodological sources in empirical software engineering [24 - 26], the empirical evaluation of our approach is an important research phase in this PhD project. Its role is to gauge (1) the ease of use, effectiveness and relevance of the approach, and (2) the generalizability of the approach. In our research design, we first evaluate the ease of use, effectiveness and relevance from the perspective of practicing BAs. By referring to the ease of use concept originally published by FD Davis [28], we measure ease of use by gauging how easy it is for the BAs to use the approach as a part of requirements gathering, do they find it easy to adapt to this new way or do they consider this as a paradigm shift that they are not able to relate to. By effectiveness of PQs, we intend to investigate the degree to which the PQs are successful in producing a desired result i.e. assist the BAs in unearthing architectural information from the client during requirements gathering. By relevance we mean to investigate whether the BAs find the approach important to their requirement gathering practices and would add value to it. Furthermore, regarding examination of generalizability, we include the perspectives of SAs. We need to test generalizability of the ASFRs and PQ-flows. The target of generalization is the set of all cases in which the communication between BAs and SAs mostly takes place through SRS, and expertise is not shared between them. In other words, we hope that the answer to this is applicable to all such cases, with due allowance made for uncertainty in the answer. For
examining generalizability, the strategies described by Wieringa and Daneva [66] would be considered.

7.2 Research Questions

In order to empirically evaluate our approach, we set out to find answers to the following research questions (RQs):

**RQ 1:** To what extent does a BA perceive it easy to use the approach?

**RQ 1.1:** How easy or difficult is it for the BAs understand the PQs on their own?

**RQ 1.2:** What kind of effort / training is needed so that the BA can start using the approach on their own?

**RQ 2:** How do the PQ-flows of the 10 categories help improve architectural relevance of requirements in a SRS?

**RQ 2.1:** To what extent are the questions in each category architecturally relevant (no superfluous questions), and

**RQ 2.2:** To what extent are all architecturally relevant questions for each category asked?

**RQ 2.3:** To what extent are all ASFR categories architecturally relevant for the system being specified?

We conducted two empirical studies: (1) to answer RQ1 (henceforth referred to as Study 1) and (2) to answer RQ2 (henceforth referred to as Study 2). The two studies, though different in terms of participants and execution style, build upon each other. Each study’s research process is organized into three main phases: Design, Execution and Analysis [5]. In the next section, we detail each of the studies.

7.3 Research Methodology and Research Plan

7.3.1 Study 1

In this section we detail the research methodology, threats to validity and results of Study 1.

7.3.1.1 Research Methodology

**Design.** We compared the research methodologies that are most relevant to studies in SE [5]. We chose a qualitative interview-based evaluation research method and followed R. Yin’s guidelines for case study design [6]. We chose interviews to obtain a detail-rich, holistic and contextualized description from the participants about the approach. The interview technique was selected for two reasons: (1) it is suitable for inquiry like ours, and (2) the resulting data offers a robust alternative [6] to more traditional survey methods. We
triangulated the data collected from multiple sources (e.g. participants with varied domain expertise, years of experience, educational background). As we wanted to collect BAs’ feedback, we designed our interview study by (1) composing an interview questionnaire to help the participant structure her response (2) testing the questionnaire with an experienced researcher and implement changes to improve it; (3) doing a pilot interview to check the applicability of the questionnaire in a real-life context; (4) carrying out in-depth interviews according to the finalized questionnaire.

**Execution.** All the BAs were informed in advance of our research goals and the interview process. The interviews were on a one-to-one basis. The author of this dissertation conducted all the interviews. Table 7.1 presents details of the interview participants. The 10 ASFR categories were shared with participating BAs and they were asked to choose one category that they are most familiar with and one that they are least familiar with. For the two chosen categories, we shared the corresponding PQ-flows and the interview questionnaire. The duration of each interview was between 30 and 60 minutes. The questionnaire included sections designed to collect information about the BA’s (i) experience and application domain (ii) understanding of ASFRs and PQ-flows. The questionnaire is presented in Appendix E on page 139.

**TABLE 7.1: DETAILS OF PARTICIPANTS FOR STUDY 1**

<table>
<thead>
<tr>
<th>Participant ID</th>
<th>Application Domain</th>
<th>Country</th>
<th>Total Years of Experience as a BA</th>
<th>Educational Background</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Insurance</td>
<td>India</td>
<td>4</td>
<td>B.COM, MBA</td>
</tr>
<tr>
<td>P2</td>
<td>Banking</td>
<td>India</td>
<td>3</td>
<td>BE, MBA</td>
</tr>
<tr>
<td>P3</td>
<td>Telecom</td>
<td>Israel</td>
<td>13</td>
<td>BSc.</td>
</tr>
<tr>
<td>P4</td>
<td>Insurance</td>
<td>India</td>
<td>6</td>
<td>BE, MBA</td>
</tr>
<tr>
<td>P5</td>
<td>Insurance</td>
<td>India</td>
<td>6</td>
<td>BSc., MBA</td>
</tr>
<tr>
<td>P6</td>
<td>BFSI</td>
<td>India</td>
<td>2</td>
<td>B.COM, MBA</td>
</tr>
<tr>
<td>P7</td>
<td>Telecom</td>
<td>India</td>
<td>18</td>
<td>BSc., MBA</td>
</tr>
<tr>
<td>P8</td>
<td>Telecom</td>
<td>Israel</td>
<td>10</td>
<td>BSc.</td>
</tr>
<tr>
<td>P9</td>
<td>Telecom</td>
<td>India</td>
<td>7</td>
<td>BCA, MCA</td>
</tr>
<tr>
<td>P10</td>
<td>Telecom</td>
<td>India</td>
<td>3</td>
<td>BE, MBA</td>
</tr>
</tbody>
</table>

B.COM – Bachelors of Commerce, MBA – Masters of Business Administration, BE – Bachelors of Engineering, BSc. – Bachelors of Science, BCA – Bachelors of Computer Application, MCA – Masters of Computer Application

**Analysis.** We used qualitative coding of our data [7], which helps us classify the various reasons as to why BAs perceive a particular category and/or PQs as more difficult or easier than others.

**7.3.1.2 Threats to Validity**

Regarding Study 1, we devised measures to counter the following validity threats [5]:

(1) **Researcher’s bias:** as the author of this dissertation is the one who was involved in creation of the PQ-flows, there is an elevated risk of passing bias into data collection and
analysis. To reduce this risk, the author let the BA select the category to discuss and freely explain the kind of difficulties felt. The author took conscious steps to avoid providing any unnecessary information or explanation, except those that the BA asked explicitly.

(2) **Interviewee background:** BAs could vary in terms of collaboration relationships they established with their respective SAs in a project. Some BAs might be more exposed to SAs’ work than others. We think however that this threat is minimal because our participants worked in organizations that have standard project delivery process; where knowledge sharing standards and tools are instrumental in keeping SDLC processes consistent across projects in the same domain.

### 7.3.1.3 Results

In this section we present our findings regarding our RQ 1.

**RQ 1: To what extent does a BA perceive it easy to use the approach?**

As indicated in Table 7.1, we received responses from 10 BAs. Our results on sub-RQs in RQ 1 are as follows:

**RQ 1.1: How easy or difficult is it for the BAs to understand the PQs on their own?**

In our study, we found that the senior BAs (those with 10 or more years of experience) could understand all the PQs on their own. The BAs at the mid-level of experience (5 to 9 years) needed guidance to understand some questions (on an average 3 questions) and junior BAs (less than 5 years’ experience) needed relatively more guidance (on an average 5 questions).

Moreover, we found that the domain expertise did not really have any influence on the understanding of the PQs, which indicated that our PQs are generic across business information system domains. Another factor that affected the result was the BA’s educational background. This observation was especially relevant for junior BAs (less than 5 years’ experience). Junior BAs with a computer science (CS) background found it easier to understand the PQs than the junior BAs with non-CS background. The BAs attributed this difference to the fact that the BAs with CS background had already taken software architecture course as a part of their educational curriculum and therefore they are familiar with the PQs vocabulary. This indicates that our approach can be used to improve the level of architectural understanding of junior BAs with non-CS background.

**RQ 1.2: What kind of effort / training is needed so that the BA can start using the approach on their own?**

From the responses received from the BAs, we observed that providing some form of guidance to further elaborate the PQs would improve the understandability of the PQs.

In the words of participant P9, “elaboration of PQs is important. When we ask questions to the client, they sometime ask us to elaborate it or sometimes they ask counter questions to us. So, we should be well versed with what the PQ is all about.”
Another participant P7 mentioned, “We cannot use PQs as a black box. We need to know what it entails”.

Based on the analysis of the interview responses, we found that a guidance mechanism for elaborating PQs could take multiple forms: (1) a one hour self-training module for junior BAs, (2) an embedded self-training module in the tool, (3) a user manual with ‘how to start working’ steps detailed, (4) some kind of look up facility in the tool as and when required.

7.3.2 Study 2
In this section, we detail the research methodology, threats to validity and results of Study 2.

7.3.2.1 Research Methodology

Design. Our research plan is to ask volunteer BAs, pseudo-clients and SAs (5 each) to simulate a process in which requirements are specified by the BA in consultation with the pseudo-client and is used by the SA to design software. The goal is to observe and analyse simulations in which BAs and SAs use our approach, and to use these observations to answer RQ 2. The design includes a volunteer BA and a pseudo-client who simulates a RE process using our approach, and a volunteer SA who assesses how much the resultant SRS contributes in explicating the hidden architectural requirements. On the basis of a post-simulation interview with the SAs, we collect their reflections on their experience. Our post-simulation interview questionnaire is developed using the same steps as in Study 1 and is presented in Appendix E on page 139.

Execution. We sent the study participation request through email to BAs, pseudo-clients and SAs explaining them our study goal and process. The BAs and SAs were experienced team members responsible for delivering large projects in varied domains. Any person with more than 20 years of experience working closely with clients to understand their requirements and to architect systems qualified as a pseudo-client. Based on the domain expertise, we grouped the BAs, SAs and pseudo-client. We created 5 such tuples. The details of the participants in each tuple is presented in Table 7.2. The study execution included the following steps:

(1) We provided the list of 10 ASFR categories along with their description to each of the participating BA.

(2) We asked each BA to choose an ASFR category of her choice based on her expertise and familiarity with the category. If a category was chosen by a BA, we marked the same in the provided list. We told the BAs that for us to ensure validation of more number of categories, we would prefer them to choose a category that is not previously chosen by other BAs. However, we clearly mentioned that this is just our preference and they are free to choose otherwise. To our delight, each BA chose a different ASFR category. The category chosen by each of the 5 BAs is indicated in Table 7.2.
(3) The PQ-flow corresponding to the chosen ASFR category was provided to the BA along with detailed instructions on how to use it.

(4) At the meeting between the BA and the pseudo-client, we provided one sample requirement from real life SRS document corresponding to the domain and the chosen ASFR category. The selected requirement corresponding to each of the chosen ASFR category is also presented in Table 7.2. The BA used the PQ-flow to assist the pseudo-client in elaborating the selected requirement with architectural details and create a much detailed requirement.

(5) This detailed requirement specific to the chosen ASFR category along with the complete SRS from which this requirement was taken was then given to the participating SA. The SA assessed the elaborated requirement and gave feedback on whether the PQ-flows helped in detailing the requirement further by providing architectural details pertinent to the requirement. In addition to the assessment of PQ-flows, we also provided each SA with the list of all the 15 ASFR categories so that they can comment on the relevance of each of the category.

### TABLE 7.2 DETAILS OF PARTICIPANTS FOR STUDY 2

<table>
<thead>
<tr>
<th>Tuple Id</th>
<th>BA Exp. (years)</th>
<th>SA Exp. (Years)</th>
<th>Pseudo-client Exp. (Yrs)</th>
<th>Domain</th>
<th>Chosen ASFR Category</th>
<th>Requirement from SRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>7</td>
<td>10</td>
<td>21</td>
<td>Insurance</td>
<td>Audit Trail</td>
<td>All changes shall be logged so you can see who added/edited/deleted a client and when.</td>
</tr>
<tr>
<td>T2</td>
<td>6</td>
<td>9</td>
<td>23</td>
<td>Insurance</td>
<td>Business Process ‘State’ Alert</td>
<td>A notification shall appear when the user logs onto the system (to inform that the client has converted the quotation to a policy).</td>
</tr>
<tr>
<td>T3</td>
<td>6</td>
<td>12</td>
<td>27</td>
<td>Banking</td>
<td>Report</td>
<td>The system shall create a detailed client report at the end of each quarter.</td>
</tr>
<tr>
<td>T4</td>
<td>5</td>
<td>8</td>
<td>20</td>
<td>Banking</td>
<td>Batch Processing</td>
<td>On a daily basis, there shall be a batch process that activates the future to-dos, changes the urgency level, and close to-dos.</td>
</tr>
<tr>
<td>T5</td>
<td>8</td>
<td>8</td>
<td>21</td>
<td>Healthcare</td>
<td>Search</td>
<td>The system must search in the systems mentioned in the &lt;&lt;document name&gt;&gt; and identify the patient record.</td>
</tr>
</tbody>
</table>

**Analysis.** As we collected participants’ reflections in the form of qualitative data, we used the coding method similar to Study 1. We expected it to yield codes that explain why the approach worked according to the participant or why they would (or would not) use the approach in their next project and what improvements are needed in the approach to make it practically more relevant.
7.3.2.2 Threats to Validity

Regarding Study 2, our most important concern is that the simulation is dependent on BA, SA and pseudo-client tuples and as we are relying on volunteers in each role, the relationship between the three is not known to us. However, we relied on professional code of conduct and even if the BA, SA and the pseudo-client have prior working history, we asked them to avoid referring to it during the simulation. Following [8], while a simulation in practical settings may be hard to generalize to other context, its key value is in experiencing what in a method works and why it works (or why not). We take this simulation as a pilot and expect the learning from it to be instrumental in improving our approach and its application scenario.

7.3.2.3 Results

In this section, we present our findings regarding our RQ 2.

RQ 2: Can the PQ-flows help improve architectural relevance of requirements in a SRS?

Following are our findings based on the responses we received from each of the five SAs who participated in the simulation study:

From the analysis of the responses, we found that even though all the SAs found the approach to be very relevant to unearth implicit architectural concerns, the opinion of the SAs on the relevance of an ASFR category or a PQ was highly influenced by the domain of expertise and the current project the respective SA was working on. Each SA viewed the ASFR category or the specific PQ from the angle of how relevant it is in her/his current project context. None of the SAs said that the existing ASFRs and/or the PQ-flows are irrelevant. However, they did comment on the relative significance of few of the PQs and ASFRs based on their current project context, changing business trends and the advancements in technology. For example, the SA in T1 with experience in Insurance domain mentioned the following about an ASFR category:

“If you see the category User behaviour Analysis. This is too specific to applications with consumer-oriented front end. Generally an Insurance application for Indian clientele would not worry much about this category as even today in India as you know most of the insurance policies are bought off line from agents. So we would not worry much about user experience in this case”

From this response we also gauged that in addition to domain, the geography in which project needs to be deployed also has an impact on the relevance of an ASFR Category and/or PQ. For example, a law prevalent in certain geography may advocate certain architectural considerations.

Also, all of the SAs mentioned that the PQs and ASFR categories would get updated based on the latest trends in the business sector and technology sector. For example, SA in T5 with experience in healthcare domain mentioned the following:

“With the advent of, say for example, eHealth, mHealth, Telehealth, we would see new ASFR categories and therefore new PQs. Some of the existing PQs may become irrelevant as well”.
Further, SA in T3 talked about block chain technology and how is it reshaping the banking sector and therefore how new ASFR categories and PQs would be needed to address the new architectural needs.

From the responses received from the SAs, it is clear that our approach comprising of ASFR categories and PQ-flows to assist BAs in unearthing implicit architectural concerns during requirements elicitation is relevant. However, the relevance of the existing ASFR categories and the corresponding PQ-flows is heavily dependent on factors such as the project domain, geography and technology and this is subject to evolution based on the specific project context. We consider this feedback important as it sheds light on the possible directions that our work can take to make the repository of ASFR Category and PQs-flows more useful to its users.

7.4 Conclusion

In this chapter, we presented the most important findings of the empirical evaluation of our solution approach. This evaluation was intended to gauge the ease of use, effectiveness, relevance and generalizability of our approach. From the responses received from the BAs, we conclude that the tool based on the approach should have an embedded self-training module for the BAs. Further, each PQ should include some explanatory text along with it as a ready reckoner for the BA. The BA can look up this text while gathering requirements.

Next, regarding relevance, we observed that the SAs found our approach to be relevant. However, the relevance of the existing ASFR categories and PQ-flows is heavily dependent on factors such as the project domain, geography and technology. For example, the ASFR category User Behaviour Analysis is not critical for an Insurance application for Indian clientele as in India most of the insurance policies are bought off line from agents. This finding also provides answers to the question on generalizability of our approach. The approach itself is generalizable, but the ASFRs and PQ-flows are not. The set of ASFRs and PQ-flows would evolve depending on the project context and therefore a KB (detailed in chapter 8) with knowledge about the ASFR categories and PQ-flows would support such an evolution.
In this chapter, we present the knowledge base (KB) that we have created for managing ASFRs and their corresponding PQ-flows. The KB has the seed knowledge of the 15 ASFR categories and their corresponding PQ-flows. This seed knowledge is extendable; we expect it to be enhanced further by contributions from SAs.

8.1 Introduction

In chapters 4 and 5, we explained the process of arriving at the 15 categories of ASFRs and their corresponding PQ-flows. In this chapter, we introduce the concept of an evolvable KB of PQs and PQ-flows that would allow SAs to add or update the ASFR categories and PQ-flows based on their specific project settings and needs. For this, we have extended an existing framework developed in the organization in which the author of this dissertation work. The existing framework is the so-called Knowledge-assisted Ontology-based Requirements Evolution (K-RE) method and its complementary toolset to address the knowledge and collaboration needs of RE stakeholders [38, 154]. The author of this dissertation is a part of the team that developed this framework.

In this chapter, we would first briefly explain K-RE and then in detail would explain how we have extended it to support creation of a KB for ASFRs and PQs. We then present a validation study intended to validate the KB for its ease of maintenance and how well it contributes to making the ASFRs and PQs generalizable.

8.2 The Knowledge-assisted Ontology-based Requirements Evolution (K-RE) Framework

Most of the RE methods way back in 2008-2009 treated the RE exercise as something that began from scratch and assumed a ‘clean slate’ approach which outlined a series of steps to define, analyse, specify and validate requirements collaboratively with relevant stakeholders. Different projects in any given domain has a lot of commonalities in addition to variabilities. Starting the RE exercise from scratch each time do not take advantage of these commonalities and results in a lot of effort and time wastage in the hands of both BAs and domain experts. To remedy this situation, we had proposed a method and framework to enable Knowledge
assisted Requirements Evolution (K-RE). K-RE starts with a seed requirement specification in place as opposed to the tradition ‘clean slate’ approach. The seed contains structured domain knowledge as represented by core requirements knowledge elements such as the core business events, actions and decisions (as captured in business processes), and constraints. Each time a new software application is to be developed; we start with this seed specification and ‘evolve’ it by way of altering and/or adding to the core to get to the final requirement specification. This is done in a semantically assisted way. The semantic assistance comes from ontologies that can be created, maintained and evolved collaboratively and a context-sensitive alert mechanism that provides online alerts as a BA evolves her specification from the seed. Each new exercise of requirements definition thus, becomes an evolution of a pre-existing structured domain knowledge-base tailored to suit specific projects [154].

### 8.2.1 Foundations of K-RE

K-RE organizes knowledge along four distinct contexts: (1) environment, (2) problem domain, (3) generic requirements, and (4) RE process. The semantic assistance in K-RE comes from the inference rules operating on the four ontologies that represent these knowledge contexts. The framework also incorporates abstractions from various knowledge modelling paradigms like feature models [25], business process models [26], data models, and use case models [27], to capture and organize knowledge. For more information on the framework, we refer interested readers to two publications [38, 154].

The four ontologies in K-RE are: (1) Environmental Context Ontology, (2) Generic Requirements Ontology, (3) RE Process Ontology and (4) Problem Domain Ontology. We next explain each of the ontologies briefly. In Figure 8.1, we reproduce from [38] the example instances of the ontologies depicted using the Unified Modeling Language (UML) class diagram notation. Below, we present these ontologies as follows:

**Environmental Context Ontology:** This ontology is designed to capture the environment for which software requirements are to be defined. The abstractions such as *Actor, Action, Domain, Line of Business, Client, and Geography* are used to capture this information.

**Generic Requirements Ontology:** This ontology comprises of requirements knowledge elements such as business goals, features, business processes and sub processes, business constraints (laws of the land, organizational policies), use cases, and business entities. The Generic Requirements Ontology provides these abstractions [155].

**RE Process Ontology:** This contains abstractions specific to the RE process, for e.g., Agile Method has requirement representation in the form of user story, iteration mechanism as sprint, and tracking mechanism as burn down whereas Waterfall Method would have requirements representation in the form of business processes, sub processes, business rules, use cases etc.

**Problem Domain Ontology:** This ontology provides abstractions to capture the essence of the problem domain. The abstractions such as *BusinessEvent, BusinessType, Party, and BusinessAction* are used to capture this information.
The K-RE framework is supported by a toolset called the K-RE tool. It provides a wiki-like user interface for knowledge contributor to contribute domain knowledge, knowledge curator to maintain the correctness and consistency of the KB and for BA to access and configure the knowledge. Next we explain the 3 main roles in K-RE.

**Knowledge contributor:** This role is taken by an experienced BA who contributes generic and specific domain knowledge to the repository.

**Knowledge curator:** As a Subject Matter Expert (SME), the curator monitors and ensures quality of knowledge. She ensures that the knowledge content is correct and current. Curator acts as the reviewer and monitors the contributions made by the knowledge contributor.

**Requirement analyst:** This is an end user of the K-RE who configures available domain knowledge as per the project environment parameters and scope.
We had developed K-RE as a Web-based tool with a centralized application server and database server which is accessible to multiple clients over the Internet. RDF-OWL [156, 157] ontologies and a relational data store [158] was used to store knowledge models and their instances. The design incorporated collaborative aspects of Web 2.0 for a participatory information sharing and collaboration among the RE stakeholders. The semantic assistance provided by K-RE used OpenNLP – NLP toolkit [159], WordNet lexical database [160], and RDFOWL ontologies along with Semantic Web Rule Language (SWRL) rules.

With this background knowledge about K-RE, we next explain how we have extended K-RE for creating a KB for ASFR categories and PQ-flows.

8.3 Extending K-RE to accommodate the Knowledge of ASFRs and PQ-flows

In order to accommodate the knowledge of ASFRs and PQ-flows in K-RE, we have augmented the Generic Requirements Ontology (explained in Section 8.2.1) to include two more elements namely ASFR category and PQs. Figure 8.2 shows the augmented Generic Requirements Ontology. The concepts highlighted in blue font are the new additions.

The concept ‘ASFR Category’ would store the various categories of ASFRs such as Print, Audit Trail, Batch Process and so on. The concept ‘PQs’ would store the PQs corresponding to each of the identified ‘ASFR Category’.

We also modified the existing roles of Knowledge Contributor and Knowledge Curator.

In Knowledge Contributor role, we added two sub roles, namely, Requirements Analysts and Software Architect. The Software Architect role would be taken up by an experienced SA who would contribute her knowledge about the ASFR categories and the corresponding PQs to the KB. The Requirements Analyst would contribute to the remaining concepts in the ontology.

In the Knowledge Curator role, we added two sub roles, namely, Requirements Knowledge Curator and Architectural Knowledge Curator. The Requirements Knowledge Curator would be a SME from the requirements field and would curate the contributed knowledge pertinent to various requirements artefacts. The Architectural Knowledge Curator would curate the contributed knowledge pertinent to ASFR categories and PQs.
Figure 8.3 shows the process map that explains how these roles will work together in a process to update the knowledge repository.

As is evident from the Figure 8.3, in K-RE, the SA first creates the KB by contributing ASFR categories and corresponding PQ-flows. Once the knowledge is contributed by the SA, it goes to the workflow of the Architectural Knowledge Curator. The Architectural Knowledge curator curates the knowledge contributed by the SA. Once the knowledge is curated, it is made available to the Requirement Analyst so that she can use it during the requirements gathering exercise to ask architecturally relevant questions to the client and produce an architecturally richer SRS.
8.4 Sample Screenshots of the Augmented K-RE Framework

Figures 8.4 and 8.5 present screenshots of the augmented K-RE framework that we have developed for managing the KB of ASFR categories and PQ-flows. The screenshot in Figure 8.4 depicts the user interface for managing ASFR categories. Using this interface, a SA can create new ASFR categories, edit existing category names and descriptions or delete existing categories. The screenshot in Figure 8.5 depicts the user interface for creating PQ-flows for a selected category.

FIGURE 8.4: SCREEN SHOT 1 – USER INTERFACE FOR MANAGING ASFR CATEGORIES
8.5 Validation Study

To validate the KB, we carried out a small validation study. The goal of this study was to find answers to the following research questions (RQs).

**RQ 1:** To what extent does a SA perceive it easy to maintain (Add/ Update/ Delete PQs and PQ-flows) the KB?

**RQ 2:** Would such a KB make the approach more generalizable making it applicable to different project domains and settings?

We included five experienced software architects (SAs) to validate the KB. The author of this dissertation showed a demo of the KB to each of the SA separately. In the demo, the author showed the SAs the seed knowledge and demonstrated how new knowledge can be created / updated. Each SA was then asked to do a small exercise wherein she would create a random ASFR (from her domain of expertise) and its corresponding PQ-flow independently. If she has any doubt, she was asked to first look up either the user manual or the tool’s inbuilt assistance mechanism. If neither of them succeed in clearing the SA’s doubt, the SA can ask the author to clarify the doubt. During this exercise, the author remained a silent observer and spoke only in case the SA asked her any questions.

Once the exercise was completed, each SA was sent a questionnaire through email. The questionnaire is presented in Appendix F on page 141. Please note that Question 1 and Question 2 and their sub-questions (from the questionnaire in Appendix F, page 141), were already answered by the author based on her observations during the exercise. The SAs were
asked to comment on these answers if they disagreed with any of the observations of the author. The questionnaire included questions to gauge the ease of maintenance of the KB and validation of whether the KB would contribute to making the approach more generalizable. By referring to the ease of use concept originally published by FD Davis [28], we measure ease of maintenance by gauging how easy it is for the SAs to maintain the KB by keeping the ASFRs and PQ-flows updated to be relevant to the specific project settings.

Next, we summarize the findings from the validation study with respect to each of RQ.

RQ 1: To what extent does a SA perceive it easy to maintain (Add/ Update/ Delete PQs and PQ-flows) the KB?

The author observed that on an average, the SAs took 3 minutes to create a simple PQ-flow and around 6 minutes to create an average complexity PQ flow. None of them attempted a complex PQ-flow with multiple conditions and flows. None of the SAs asked for any clarification from the author. Each of the SA took help of the tool’s in-built assistance two to four times while creating the PQ-flow. The assistance was needed mostly in case of creating or editing conditional flows. The SAs mentioned that the inbuilt assistance was really very helpful in clearing their doubts. None of the users had to consult the user manual provided by the tool indicating that the initial demo provided by the author and the in-built assistance provided by the tool was good enough for the SAs to use the tool independently.

The SAs tried 3 kinds of flows – simple, conditional and fork. The tool supported all these flows. To the question on “Does the tool allow modelling all kinds of flows in the PQ-flow”, one of the SA mentioned:

“The tool seems to be robust enough to handle all kinds of PQ-flows. However, it is not practical to envisage and model all kinds of possible scenarios at this moment. So any shortcomings in the tool would be discovered as and when more SAs use it for creating real KB.”

On similar lines, another SA mentioned:

“I tried some basic flows and it all worked. On field, many more complex scenarios can come in. Given the fact that this is an extension of an existing established tool, I believe it would be able to accommodate any such complex flows. But remains to be validated!”

On the performance of the tool, we mostly got positive responses from all the SAs. The SAs were happy with the speed and look and feel of the tool. One of the SAs mentioned that he found the in-built assistance “too texty” and suggested having some video assistance to make it more attractive.

RQ 2: Would such a KB make the approach more generalizable making it applicable to different project domains and settings?

All the SAs very strongly indicated that this tool has solved the generalizability issue that our approach had. The SAs indicated that with such a tool to manage KB, they can add / update / delete the ASFRs and PQ-flows based on their specific project needs. They acknowledged that
creation of KB is an effort intensive activity but were at the same time appreciative of the returns it would give in the long run. One of the SAs mentioned,

“This approach would require a mind-set change that is difficult but not impossible to achieve in any organization.”

Another SA mentioned that,

“It is very clear that there is an upfront investment in creating a KB. I can see this as a possible hindrance to its adopting. To overcome this, the project management team would have to be convinced about the long term benefits that this approach would bring in.”

8.6 Conclusion

In this chapter we presented the KB that we have created for managing ASFRs and PQ-flows. The KB is an extension of the existing KB used in the organization in which the author of this dissertation is employed. The KB has the seed knowledge of the 15 ASFR categories and their corresponding PQ-flows. This seed knowledge can be enhanced further by contributions from SAs. The evaluation study that we conducted concluded that the tool is easy to maintain and also by having such a KB of ASFR categories and PQ-flows in place, we make our approach generalizable to different project domains and settings. We realize however that the success of the approach would depend largely on the quality of KB that we are able to create. Also, adopting K-RE would require a mind-set change that is difficult to achieve in any organization. The upfront investment in creating a KB could also be a hindrance to adopting this approach.
In this dissertation, we have studied how to enhance communication between BAs and SAs by introducing a KB with architectural knowledge that would be used by BAs during requirements elicitation to produce an architecturally richer SRS. Using an empirical research method, we have developed an initial version of such a KB and have validated the proposed approach to make the knowledge of experienced SAs available to BAs during requirements elicitation, using a series of validation studies. In this chapter, we summarize the contributions of this dissertation in relation to the research goal and questions discussed in Chapter 1. We furthermore highlight open issues and identify topics that we think are relevant for future research on the topic.

9.1 Revisiting the Research Questions

In this section, we revisit each of the RQs answered in this dissertation and summarize the answers yielded by our research.

In this PhD project, we have studied how to make the knowledge of experienced SAs available to BAs during requirements elicitation. As indicated in Chapter 1 (Section 1.5, page 6), to meet the research objective, we have elaborated the central RQ into three more detailed RQs. In this section, we first summarize the answers for the detailed RQs and the sub-questions therein (Section 9.1.1 to 9.1.3). Next, in Section 9.1.4, we revisit our central research question.

9.1.1 How do SAs currently identify architecturally significant information from a SRS? (RQ 1)

To answer RQ 1, we investigated the problem and the available treatments. These are detailed in Chapter 3 and Chapter 4. We used two research methods namely literature review and a case study to find answers to RQ 1. We used a semi-systematic literature study to capture the state-of-the-art in architecturally significant requirements. We further used quantitative semi-structured interviews to supplement the literature review. As part of our research on RQ 1, we executed a qualitative case study with 14 practicing SAs from India, the United States of America and the Netherlands. The key findings of this study are: (1) SRSs often lack crucial architecturally relevant information needed to make informed architectural decisions, (2) Probing Questions (PQs) are an essential mechanism for unearthing underspecified architecturally relevant information, (3) a total of 15 categories of ASFRs, each
with its own set of PQs were identified through the study, (4) and there is a logical way to sequence the PQs during the requirements elicitation process.

9.1.2 How to use the mechanism to equip BAs to ask more architecturally relevant information? How could we design an approach that integrates the mechanism in an organization’s software delivery process? (RQ 2)

Answering RQ 1 led us to understand that PQs are used by SAs as the mechanism to elicit architecturally relevant information. Moreover, the PQs have a logical sequence associated with them. This being known, we headed towards finding answer to our design problem stated in RQ 2. We started with a ‘basic’ design of the PQ-flows and then integrated practitioners’ feedback (using a survey) into the design. This resulted in a refined version of the PQ-flows for which further feedback was elicited. Based on the analysis of the survey findings, PQ-flows for the 15 ASFR categories were created. This study is reported in Chapter 5.

We then designed an automated support for recommending relevant PQ-flows for an SRS and then generating annotated PQ-flows wherein the annotations on the PQs contain possibly relevant requirements from the SRS that already answers some of the PQs. The BAs can leverage this information to determine what is already known, what knowledge is missing, and what effect this would have on the flow of the PQs. We automated the recommendation using existing machine learning techniques. This study is reported in Chapter 6.

Having clearly envisaged the practical scenario that the PQs and PQ-flows would change based on specific project settings, we extended our design further to include a KB comprising of the seed knowledge of the 15 ASFR categories (identified as a part of RQ 1) and the corresponding PQ-flows. This KB is amenable to evolution by contributions from SAs. The SA can contribute new ASFRs, PQ and PQ-flows and / or modify the existing ones based on their specific project and/or business domain needs. This is reported in Chapter 8.

9.1.3 To what extent is the proposed way of working usable, useful and generalizable in practice? (RQ 3)

Our RQ 3 focused on treatment validation wherein our goal was to evaluate the utility and usefulness of the approach. For this, we carried out a series of validation studies, which included both expert-perception based and experimental validation. In Chapter 4, we reported a quantitative evaluation study to validate our theory against an actual SRS document. The goal of this first evaluation was to validate if the SRS documents in fact lack the needed architectural details to make informed architectural decisions. We found that very few (0% to 33 %) architectural details were actually present in the SRS document. This strengthened our working hypothesis that ASFRs do not explicitly state the architectural details and therefore PQs are a necessary mechanism to unearth them.
In Chapter 6, we performed two preliminary studies to evaluate the usefulness of the contextualization approach in situ. We gathered feedback from both the SAs and the BAs. The BAs found the approach to be applicable in the requirements elicitation process (Section 6.3.1, page 78). They found the hierarchical structure of the PQ-flows useful and they were positive on using this approach in future projects. The SAs found the approach to be valuable (Section 6.3.2, page 79). They acknowledged that the effort to get such level of architectural details into SRS would be too high in the absence of such an approach. The feedback we received from the BAs and SAs therefore indicated that our approach seems applicable and useful in real world IT project settings.

In Chapter 7, we presented the empirical evaluation of our solution approach. This evaluation effort was intended to gauge the ease of use, effectiveness, relevance and generalizability of our approach. From the responses received from the BAs and SAs, we concluded that the tool based on the approach should have an embedded self-training module for the BAs. Next, regarding relevance, we observed that the SAs found our approach to be relevant. However, the relevance of the existing ASFR categories and PQ-flows was heavily dependent on factors such as the project domain, geography and technology. This finding also provided answers to the question about the generalizability of our approach. The approach itself is generalizable, but the ASFRs and PQ-flows are not. The set of ASFRs and PQ-flows would evolve depending on the project context and the KB with architectural knowledge (detailed in chapter 8) would support such an evolution.

In Chapter 8, we carried out a validation study to evaluate the KB. The evaluation was to gauge (1) the maintainability of the KB and (2) generalizability potential of the KB. The study that we conducted concluded that the KB is easy to maintain and also by having such a KB of ASFR categories and PQ-flows in place, we make our approach generalizable to different project domains and settings. We realized however that the success of the approach would depend largely on the quality of KB that we are able to create. In addition, adopting such a KB would require a mind-set change that is difficult to achieve in any organization. The upfront investment in creating a KB can also be a hindrance to adopting this approach.

9.2 Contributions

Based on the findings of our work, we posit that we have achieved our research goal successfully. This section presents the main contributions of the dissertation as mentioned below:

C1: Our first contribution is the initial list of 15 categories of ASFRs that form the seed knowledge of our KB.

C2: Our second contribution is the PQ-flow catalog that we created for 10 of the identified ASFR categories. These also form the seed knowledge of our KB.

C3: Our third contribution is the creation of KB for ASFRs and the corresponding PQ-flows to let the SAs evolve the KB with relevant ASFRs and PQ-flows. This is to cater to the fact that the ASFRs and PQ-flows would evolve based on specific project needs.
**C4:** Our fourth contribution is the automated support for recommending relevant PQ-flows for an SRS and then generating annotated PQ-flows wherein the annotations on the PQs contain possibly relevant requirements from the SRS that already answers some of the PQs. The BAs can leverage this information to determine what is already known, what knowledge is missing, and what effect this would have on the flow of the PQs.

These contributions and the research method used to deliver them have been progressively made public in several publications as indicated in Chapter 1 (Section 1.9, page 12)

### 9.3 Directions for Future Work

The work described in this dissertation can be extended by following several possible future research directions.

As indicated earlier, the success of our approach depends largely on the quality of KB that we are able to create. Therefore, an immediate extension of this work could be to enhance the seed knowledge of ASFRs and PQ-flows that currently exists in the KB.

We see many future possibilities with our fourth contribution (C 4). The scope of C4 in this dissertation was limited to establishing that an automated support for recommending ASFRs and PQ-flows is feasible. This was established using basic machine learning algorithms and on a limited dataset. Therefore, a possible future work could be to design bigger experiments using larger and more varied datasets to evaluate the scalability of our approach. This would also help improve the accuracy rates that we have reported in this dissertation. Another direction would be to utilize more advanced machine learning techniques and deep learning algorithms that have proven to produce better results on such classification and recommendation tasks. Traditional machine learning algorithms such as the ones we used in our experiments are based on shallow models trained on sparse features and they suffer from the curse of dimensionality [172]. With the recent success of word embedding, neural network models have achieved superior results on various language-related tasks as compared to traditional machine learning models. Therefore, experiments using Artificial Neural Network (ANN) models such as Convolutional Neural Networks (CNN) [173], Recurrent Neural Networks (RNN) [174], Long Short-Term Memory (LSTM) network [175] and Bidirectional Encoder Representations from Transformers (BERT) [176] is worth a try.

Finally, being a part of such a large IT organization, a Technical Action Research [29] where BAs and SAs use our approach in a real project is also a viable line of research which could lead to refining the approach and improving its generalizability.
Bibliography


Conference on Evaluation and Assessment in Software Engineering (EASE 2013), pp. 97-106. ACM.


[28] Davis, Fred D., Perceived Usefulness, Perceived Ease of Use, And User Acceptance of Information Technology, MIS Quarterly; Sep 1989; 13, 3; ABI/INFORM Global pg. 319


[42] A. Bryman, *Integrating Quantitative and Qualitative Research: How is it Done?* Qualitative research, 2006, 6(1), pp.97-113


[102] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns: Elements of Reusable Object-oriented Software*, Addison-Wesley: Reading, MA, 1995


[105] TATA Consultancy Services (TCS) [https://www.tcs.com/](https://www.tcs.com/) last accessed on 2nd Feb 2020


[115] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten (2009); The WEKA Data Mining Software: An Update, SIGKDD Explorations, Volume 11, Issue 1


In Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018 (pp. 24-34), ACM

APPENDIX A

Interview questions used in Chapter 4. We would like to make a note that section III of the questionnaire was added after obtaining the findings from Phase I of the study (Chapter 4, Section 4.3, page 37). In other words, Phase I of the study had the question only from section I (About You) and Section II (Understanding Architecturally Significant Functional Requirements). Section III (Understanding Requirements Probe) was added for Phase II of the study.

1. About You

Experience

• Total years of IT experience
• Number of years in working as software architect's capacity
• Business sector

Projects

• Out of all the projects that you were involved in, which was/were the most architecturally challenging project(s)?
• In what way would you call this project ‘architecturally challenging’? What all did it involve?
• What was the project size?
• Application type (e.g. business info systems versus embedded systems).
• For this project, where you a software architect on the vendor’s side or on the client’s side?
• What was the employment type (consultant/contractor paid on hourly basis versus full-time employee)?
• Type of contractual agreement - fixed price versus time and material.

2. Understanding Architecturally Significant Functional Requirements

How would you define a functional requirement that has a significant architectural impact?

• Can you give some examples of architecturally significant functional requirements statement?
While deciding that a certain functional requirement may be architecturally significant, what are the different types of impact you sense? Based on these do you identify some buckets?

- Can you give some examples?
- Is this categorization important?

How do you generally recognize a functional requirement that would have a significant architectural impact from a set of functional requirements?

- Can you give some examples?
- Is this way of recognizing a functional requirement with significant architectural impact always reliable?

3. Understanding Requirements Probe

Have you ever experienced a requirements specification in which there were insufficient details about potentially architecturally significant functional requirements?

We are exploring the idea of identifying a set of questions (which we refer to as “probing” questions) - which would be associated with various types of underexplored architecturally significant functional requirements. The idea is to prompt the Business Analyst to ask deeper questions so that the requirement specification contains the information needed by the Software Architect/Designer. Do you think this would be useful?

If your answer is yes, can you give examples of appropriate probing questions? Tell us some examples of the mechanisms/tricks you employ to bring out the architectural implications.

INTERVIEWEE Name:

INTERVIEWER Name:

DATE
APPENDIX B

SURVEY HOSTED ON SURVEY MONKEY

Our recently concluded quantitative study in a large vendor’s organization has indicated that software requirement specifications do not often contain the crucial information needed for making right architectural choices. Intuitive Probing Questions (PQs) by software architects appear to be a mechanism for enriching requirements with the necessary information. Based on our interviews with experienced architects from various business information system domains, we have identified a set of probing questions. The probing questions help in unearthing architecturally significant information from various architecturally significant requirement categories. We are eliciting architectural questions so that we can provide tools and processes for the Business Analysts. The goal is that the Business Analysts would ask these questions and provide a more complete set of requirements to the architect.

We have chosen ten of the architecturally significant requirement categories for this survey. They are: Audit Trail, Batch Processing, Business Process State Alerts, Report, Workflow, Localization/Multilingual, Search, Print, Online Help and Third Party Interaction.

For each of the category, we request you to analyse the diagram and answer the questions that follow.

PART 1

ABOUT YOU

Question 1: Specify the total years of your IT experience.

- Less than 1 year
- 1 to 5 years
- 5 to 10 years
- More than 10 years

Question 2: Specify the number of years you have worked in the capacity of the Software Architect?

- Less than 1 year
- 1 to 5 years
- 5 to 10 years
- More than 10 years

Question 3: What is the primary business domain on which you have worked?
PART 2

For each of the five ASFR category, the corresponding PQ flow diagram was shown and the following questions were asked:

**Question 4:** Do you have experience working with "<<one of the ASFR category>>" kind of requirements? If your answer is "No", you may choose to skip this category and move on to the next category by clicking on the "Next" button at the bottom of the page.

- Yes
- No

**Question 5:** *Missing PQs* - Are any important probing questions missing? If yes, please enter the missing PQs below. For each missing PQ, please provide its immediate precursor. (For example, <missing PQ>: place it after PQ 6).


**Question 6:** *Delete PQs* - Do you think any of the questions should be removed? If Yes, please specify which PQs according to you should be removed and why. (For example, PQ <number> : <explain why>).


**Question 7:** *Modify PQs* - Do you think any of the PQs should be modified? If yes, what according to you would be an improved question for a given PQ? (For example, PQ <number> : <Please explain and suggest an improved question>).


**Question 8:** *Question Flow* - Is the organization of questions in the above diagram logical? If not, please explain how you would change the flow.


**Question 9:** *Overall Quality of Diagram* - For the existing (as-is) diagram, to what extent would the probing questions extract the comprehensive information needed to make architectural decisions?
Almost no critical architectural issues would be unearthed
Some critical architectural issues would be unearthed
Many critical architectural issues would be unearthed
The majority of architectural issues would be unearthed

Question 10: Usefulness - The questions explored in this diagram attempt to explicitly elicit and document architecturally significant knowledge. As an architect, to what extent would you find it helpful if the answers to such questions were documented either informally, or as part of the requirements specification?

- Not at all. I prefer working with unspoken assumptions.
- It would be slightly helpful
- It would be helpful
- It would be extremely helpful

Question 11: Closing comments (if any)
APPENDIX C

Here we present the PQ-flow catalog for the 10 ASFR categories.
WORKFLOW

1. What are the different rules that the workflow needs to include and/or impact?
2. What are the different processes that the workflow needs to include and/or impact?
3. What are the different systems that the workflow needs to include and/or impact?
4. What are the security requirements (access levels) depending upon the sensitivity of the information involved at each stage?
5. Are there any external interfaces involved in this payment gateway?
6. Are the persistence/restart requirements of the workflow?
7. What is the expected volume, frequency and rate of workflow requests?
8. What is the level of nesting and/or recursion?
9. Is auditing of the workflow needed?
10. Is history logging mechanism needed?
11. Are there any regulatory or compliance requirements associated with the workflow?
12. Should multilingual support be considered?
13. Should accessibility support be considered?
14. Should an integrated workflow be considered?
15. What kind of notification mechanism is required for the workflow?
16. Is the infrastructure for notifications in place (for e.g., integrations with existing mail systems, SMS, etc.) or does it need to be designed and implemented?
17. Is an alternative/breakout notification mechanism needed if the primary one fails (for reasons such as mailbox full, network outage, etc.)?
18. Are there any expected exceptions or deviations in the workflow?
19. How to handle such exceptions and deviations?
20. Is it parallel execution workflow or sequential workflow?
21. Is there a requirement for the workflow to fork-out and re-join?
APPENDIX D

**Questionnaire for the Business Analyst**

1. Based on your experience in the pilot session, how well did/would the PQ-flows fit into your elicitation process?

2. Did you find the hierarchical structure of the questions useful? (Versus just having a collection). Please explain.

3. Based on your one-time experience, to what extent could you imagine integrating the information acquired through using the PQ-flows into the requirements specification?

4. Would you use PQ-flows in future projects? Please explain your answer.

5. What would you like to see changed in the order to make the PQs more helpful for Business Analysts?

**Questionnaire for the Software Architect**

1. What value do you see from using this approach?

2. In the absence of such an approach, what kind of effort is required in getting this level of architecture details?

3. Do you feel a business analyst equipped to ask these questions is useful or would you prefer asking these questions on your own?

4. Does this approach help in bridging the gap that exists between requirements and architecture?
APPENDIX E

‘About you’ question common to both the studies

- Total years of IT experience
- Number of years’ experience in working as a software architect /Business Analyst
- Business sector
- Educational Background

Study 1

Q 1. Are all the PQs self-explanatory or do you need explanation?

Q1.1. Please list down the PQs that are not self-explanatory and need further explanation.

Q 1.2. Please suggest how the understandability of these PQs can be improved.

Q 2. What kind of a training module do you think would be required to enable you to use the approach on your own?

Q.2.1 How much of training effort do you think would be required to use this approach on your own?

Q 2.2 How much of a training effort would you be able to put in?

Study 2

Q 1. Are there any superfluous PQs in the given ASFR category? If yes, specify which ones.

Q 2. Are all architecturally relevant questions for the given ASFR category present as a PQ? If no, specify which ones are missing.

Q 3. Are all ASFRs architecturally relevant for the system being specified? If no, specify the missing ones.
APPENDIX F

Q 1. How easy to follow are the steps in the K-RE framework for knowledge management?

Q 1.1. How much time (in minutes) did each user take to execute the steps in the framework?

Q 1.2. How many questions did each user ask in order to get help (from the author of the framework)?

Q 1.3. How many times did each user consult a document (provided by the researcher) on using the framework?

Q 2. Does the tool present useful assistance while working on it?

Q 2.1. How many times each user was able to clear his/her doubts using the tool’s assistance mechanism?

Q 2.2. How many times did the tool assistance save the researcher’s time in explanation?

Q 3. Does the tool allow modelling all kinds of flows in the PQ-flow?

Q 3.1. How many different kinds of flows did the user try?

Q 3.2. How many kinds of flows did the tool not support?

Q 3.3. What is the criticality of the unsupported flow kinds?

Q 3.4. What is the frequency of occurrence of the unsupported flow kinds?

Q 4. Were you able to create/update all the required ASFR categories and PQ-flows?

Q 4.1. If no, please specify the ones you could not and why?

Q 5. How is the performance of the tool in terms of its speed, user experience (look and feel of the tool in terms of colours used, button positioning, placement of different text, font style etc.)?
APPENDIX G

The final list of categories from the interview text during the data analysis phase. Codes are nested represented as super concept > sub concept

1. ASFR Category
2. ASFR Category>Audit Trial
3. ASFR Category>Batch Processing
4. ASFR Category>Localization/Multilingual
5. ASFR Category>Business Process “State” Alert
6. ASFR Category>Data related dialog
7. ASFR Category>Payment
8. ASFR Category>Print
9. ASFR Category>Report
10. ASFR Category>Search
11. ASFR Category>Third Party Interaction
12. ASFR Category>Workflow
13. ASFR Category>Online Help
14. ASFR Category>Licensing
15. ASFR Category>User Behaviour Analysis
16. ASFR Category>Storage Mechanism
17. Probing Questions
18. Probing Questions > Probing Question Types
19. Probing Questions > Probing Question Types > Business Rules
20. Probing Questions > Probing Question Types > strategic technology choices
21. Probing Questions > Probing Question Types > Non-functional aspects
22. Probing Questions > Probing Question Types > Regulatory Compliance
23. Probing Questions > Probing Question Types > Project Context
24. Probing Questions > Probing Question Types > Compound architectural effect
25. Probing Question Reasoning
26. Probing Question Reasoning > Domain Knowledge
27. Probing Question Reasoning > Architect’s knowledge
28. Probing Question Reasoning > Maturity of organization
SIKS DISSERTATION PUBLISHED AS OF 2011

2011

2011-01, Botond Cseke (RUN), Variational Algorithms for Bayesian Inference in Latent Gaussian Models

2011-02, Nick Tinnemeier (UU), Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language

2011-03, Jan Martijn van der Werf (TUE), Compositional Design and Verification of Component-Based Information Systems

2011-04, Hado van Hasselt (UU), Insights in Reinforcement Learning; Formal analysis and empirical evaluation of temporal-difference learning algorithms

2011-05, Base van der Raadt (VU), Enterprise Architecture Coming of Age - Increasing the Performance of an Emerging Discipline.

2011-06, Yiwen Wang (TUE), Semantically-Enhanced Recommendations in Cultural Heritage

2011-07, Yujia Cao (UT), Multimodal Information Presentation for High Load Human Computer Interaction

2011-08, Nieske Vergunst (UU), BDI-based Generation of Robust Task-Oriented Dialogues

2011-09, Tim de Jong (OU), Contextualised Mobile Media for Learning

2011-10, Bart Bogaert (UvT), Cloud Content Contention

2011-11, Dhaval Vyas (UT), Designing for Awareness: An Experience-focused HCI Perspective

2011-12, Carmen Bratosin (TUE), Grid Architecture for Distributed Process Mining

2011-13, Xiaoyu Mao (UvT), Airport under Control. Multiagent Scheduling for Airport Ground Handling

2011-14, Milan Lovric (EUR), Behavioral Finance and Agent-Based Artificial Markets

2011-15, Marijn Koolen (UvA), The Meaning of Structure: the Value of Link Evidence for Information Retrieval

2011-16, Maarten Schadd (UM), Selective Search in Games of Different Complexity

2011-17, Jiyin He (UVA), Exploring Topic Structure: Coherence, Diversity and Relatedness

2011-18, Mark Ponsen (UM), Strategic Decision-Making in complex games

2011-19, Ellen Rusman (OU), The Mind ' s Eye on Personal Profiles

2011-20, Qing Gu (VU), Guiding service-oriented software engineering - A view-based approach

2011-21, Linda Terlouw (TUD), Modularization and Specification of Service-Oriented Systems

2011-22, Junte Zhang (UVA), System Evaluation of Archival Description and Access
<table>
<thead>
<tr>
<th>Year</th>
<th>Title</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011-23</td>
<td>Finding People and their Utterances in Social Media</td>
<td>Wouter Weerkamp (UVA)</td>
</tr>
<tr>
<td>2011-24</td>
<td>Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior</td>
<td>Herwin van Welbergen (UT)</td>
</tr>
<tr>
<td>2011-25</td>
<td>Analysis and Validation of Models for Trust Dynamics</td>
<td>Syed Waqar ul Qounain Jaffry (VU)</td>
</tr>
<tr>
<td>2011-26</td>
<td>Virtual Agents for Human Communication - Emotion Regulation and Involvement-Distance Trade-Offs in Embodied Conversational Agents and Robots</td>
<td>Matthijs Aart Pontier (VU)</td>
</tr>
<tr>
<td>2011-27</td>
<td>Dynamic website optimization through autonomous management of design patterns</td>
<td>Aniel Bhulai (VU)</td>
</tr>
<tr>
<td>2011-28</td>
<td>Effective Focused Retrieval by Exploiting Query Context and Document Structure</td>
<td>Rianne Kaptein (UVA)</td>
</tr>
<tr>
<td>2011-29</td>
<td>Discrimination-aware Classification</td>
<td>Faisal Kamiran (TUE)</td>
</tr>
<tr>
<td>2011-30</td>
<td>Affective Signal Processing (ASP): Unraveling the mystery of emotions</td>
<td>Egon van den Broek (UT)</td>
</tr>
<tr>
<td>2011-31</td>
<td>Computational and Game-Theoretic Approaches for Modeling Bounded Rationality</td>
<td>Ludo Waltman (EUR)</td>
</tr>
<tr>
<td>2011-32</td>
<td>Methodological Advances in Bibliometric Mapping of Science</td>
<td>Nees-Jan van Eck (EUR)</td>
</tr>
<tr>
<td>2011-33</td>
<td>Arguing to Motivate Decisions</td>
<td>Tom van der Weide (UU)</td>
</tr>
<tr>
<td>2011-34</td>
<td>Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations</td>
<td>Paolo Turrini (UU)</td>
</tr>
<tr>
<td>2011-35</td>
<td>Explaining Agent Behavior in Virtual Training</td>
<td>Maaike Harbers (UU)</td>
</tr>
<tr>
<td>2011-36</td>
<td>Experiments in serious game design: a cognitive approach</td>
<td>Erik van der Spek (UU)</td>
</tr>
<tr>
<td>2011-37</td>
<td>Machine Learning for Pairwise Data, Applications for Preference Learning and Supervised Network Inference</td>
<td>Adriana Burlutiu (RUN)</td>
</tr>
<tr>
<td>2011-38</td>
<td>Bee-inspired Distributed Optimization</td>
<td>Nyree Lemmens (UM)</td>
</tr>
<tr>
<td>2011-39</td>
<td>Organizing Adaptation using Agents in Serious Games</td>
<td>Joost Westra (UU)</td>
</tr>
<tr>
<td>2011-40</td>
<td>Architectural Knowledge Management in Global Software Development</td>
<td>Viktor Clerc (VU)</td>
</tr>
<tr>
<td>2011-41</td>
<td>Cryptographically Enforced Distributed Data Access Control</td>
<td>Luan Ibraimi (UT)</td>
</tr>
<tr>
<td>2011-42</td>
<td>Explaining Behavior through Mental State Attribution</td>
<td>Michal Sindlar (UU)</td>
</tr>
<tr>
<td>2011-43</td>
<td>Process Improvement through Software Operation Knowledge</td>
<td>Henk van der Schuur (UU)</td>
</tr>
<tr>
<td>2011-44</td>
<td>Robust Brain-Computer Interfaces</td>
<td>Boris Reuderink (UT)</td>
</tr>
<tr>
<td>2011-45</td>
<td>Statistical Language Models for Alternative Sequence Selection</td>
<td>Herman Stehouwer (UvT)</td>
</tr>
</tbody>
</table>
2011-46, Beibei Hu (TUD), Towards Contextualized Information Delivery: A Rule-based Architecture for the Domain of Mobile Police Work

2011-47, Azizi Bin Ab Aziz (VU), Exploring Computational Models for Intelligent Support of Persons with Depression

2011-48, Mark Ter Maat (UT), Response Selection and Turn-taking for a Sensitive Artificial Listening Agent

2011-49, Andreea Niculescu (UT), Conversational interfaces for task-oriented spoken dialogues: design aspects influencing interaction quality

2012

2012-01, Terry Kakeeto (UvT), Relationship Marketing for SMEs in Uganda

2012-02, Muhammad Umair (VU), Adaptivity, emotion, and Rationality in Human and Ambient Agent Models

2012-03, Adam Vanya (VU), Supporting Architecture Evolution by Mining Software Repositories

2012-04, Jurriaan Souer (UU), Development of Content Management System-based Web Applications

2012-05, Marijn Plomp (UU), Maturing Interorganisational Information Systems

2012-06, Wolfgang Reinhardt (OU), Awareness Support for Knowledge Workers in Research Networks

2012-07, Rianne van Lambalgen (VU), When the Going Gets Tough: Exploring Agent-based Models of Human Performance under Demanding Conditions

2012-08, Gerben de Vries (UVA), Kernel Methods for Vessel Trajectories

2012-09, Ricardo Neisse (UT), Trust and Privacy Management Support for Context-Aware Service Platforms

2012-10, David Smits (TUE), Towards a Generic Distributed Adaptive Hypermedia Environment

2012-11, J.C.B. Rantham Prabhakara (TUE), Process Mining in the Large: Preprocessing, Discovery, and Diagnostics

2012-12, Kees van der Sluijs (TUE), Model Driven Design and Data Integration in Semantic Web Information Systems

2012-13, Suleman Shahid (UvT), Fun and Face: Exploring non-verbal expressions of emotion during playful interactions

2012-14, Evgeny Knutov (TUE), Generic Adaptation Framework for Unifying Adaptive Web-based Systems

2012-15, Natalie van der Wal (VU), Social Agents. Agent-Based Modelling of Integrated Internal and Social Dynamics of Cognitive and Affective Processes.

2012-16, Fiemke Both (VU), Helping people by understanding them - Ambient Agents supporting task execution and depression treatment
2012-17, Amal Elgammal (UvT), Towards a Comprehensive Framework for Business Process Compliance
2012-18, Eltjo Poort (VU), Improving Solution Architecting Practices
2012-19, Helen Schonenberg (TUE), What's Next? Operational Support for Business Process Execution
2012-20, Ali Bahramisharif (RUN), Covert Visual Spatial Attention, a Robust Paradigm for Brain-Computer Interfacing
2012-21, Roberto Cornacchia (TUD), Querying Sparse Matrices for Information Retrieval
2012-22, Thijs Vis (UvT), Intelligence, politie en veiligheidsdienst: verenigbare grootheden?
2012-23, Christian Muehl (UT), Toward Affective Brain-Computer Interfaces: Exploring the Neurophysiology of Affect during Human Media Interaction
2012-24, Laurens van der Werff (UT), Evaluation of Noisy Transcripts for Spoken Document Retrieval
2012-25, Silja Eckartz (UT), Managing the Business Case Development in Inter-Organizational IT Projects: A Methodology and its Application
2012-26, Emile de Maat (UVA), Making Sense of Legal Text
2012-27, Hayrettin Gurkok (UT), Mind the Sheep! User Experience Evaluation & Brain-Computer Interface Games
2012-28, Nancy Pascall (UvT), Engendering Technology Empowering Women
2012-29, Almer Tigelaar (UT), Peer-to-Peer Information Retrieval
2012-30, Alina Pommeranz (TUD), Designing Human-Centered Systems for Reflective Decision Making
2012-31, Emily Bagarukayo (RUN), A Learning by Construction Approach for Higher Order Cognitive Skills Improvement, Building Capacity and Infrastructure
2012-32, Wietske Visser (TUD), Qualitative multi-criteria preference representation and reasoning
2012-33, Rory Sie (OUN), Coalitions in Cooperation Networks (COCOON)
2012-34, Pavol Jancura (RUN), Evolutionary analysis in PPI networks and applications
2012-35, Evert Haasdijk (VU), Never Too Old To Learn -- On-line Evolution of Controllers in Swarm- and Modular Robotics
2012-36, Denis Ssebugwawo (RUN), Analysis and Evaluation of Collaborative Modeling Processes
2012-37, Agnes Nakakawa (RUN), A Collaboration Process for Enterprise Architecture Creation
2012-38, Selmar Smit (VU), Parameter Tuning and Scientific Testing in Evolutionary Algorithms
2012-39, Hassan Fatemi (UT), Risk-aware design of value and coordination networks
2012-40, Agus Gunawan (UvT), Information Access for SMEs in Indonesia
2012-41, Sebastian Kelle (OU), Game Design Patterns for Learning
2012-42, Dominique Verpoorten (OU), Reflection Amplifiers in self-regulated Learning
2012-43 Withdrawn
2012-44, Anna Tordai (VU), On Combining Alignment Techniques
2012-45, Benedikt Kratz (UvT), A Model and Language for Business-aware Transactions
2012-46, Simon Carter (UVA), Exploration and Exploitation of Multilingual Data for Statistical Machine Translation
2012-47, Manos Tsagkias (UVA), Mining Social Media: Tracking Content and Predicting Behavior
2012-48, Jorn Bakker (TUE), Handling Abrupt Changes in Evolving Time-series Data
2012-49, Michael Kaisers (UM), Learning against Learning - Evolutionary dynamics of reinforcement learning algorithms in strategic interactions
2012-50, Steven van Kervel (TUD), Ontology driven Enterprise Information Systems Engineering
2012-51, Jeroen de Jong (TUD), Heuristics in Dynamic Scheduling: a practical framework with a case study in elevator dispatching

2013

2013-01, Viorel Milea (EUR), News Analytics for Financial Decision Support
2013-02, Erietta Liarou (CWI), MonetDB/DataCell: Leveraging the Column-store Database Technology for Efficient and Scalable Stream Processing
2013-03, Szymon Klarman (VU), Reasoning with Contexts in Description Logics
2013-04, Chetan Yadati(TUD), Coordinating autonomous planning and scheduling
2013-05, Dulce Pumareja (UT), Groupware Requirements Evolutions Patterns
2013-06, Romulo Goncalves(CWI), The Data Cyclotron: Juggling Data and Queries for a Data Warehouse Audience
2013-07, Giel van Lankveld (UvT), Quantifying Individual Player Differences
2013-08, Robbert-Jan Merk(VU), Making enemies: cognitive modeling for opponent agents in fighter pilot simulators
2013-09, Fabio Gori (RUN), Metagenomic Data Analysis: Computational Methods and Applications
2013-10, Jeewanie Jayasinghe Arachchige(UvT), A Unified Modeling Framework for Service Design
2013-11, Evangelos Pournaras(TUD), Multi-level Reconfigurable Self-organization in Overlay Services
2013-12, Marian Razavian (VU), Knowledge-driven Migration to Services

2013-13, Mohammad Safiri (UT), Service Tailoring: User-centric creation of integrated IT-based homecare services to support independent living of elderly

2013-14, Jafar Tanha (UVA), Ensemble Approaches to Semi-Supervised Learning Learning

2013-15, Daniel Hennes (UM), Multiagent Learning - Dynamic Games and Applications

2013-16, Eric Kok (UU), Exploring the practical benefits of argumentation in multi-agent deliberation

2013-17 Koen Kok (VU), The PowerMatcher: Smart Coordination for the Smart Electricity Grid

2013-18 Jeroen Janssens (UvT), Outlier Selection and One-Class Classification

2013-19, Renze Steenhuizen (TUD), Coordinated Multi-Agent Planning and Scheduling

2013-20, Katja Hofmann (UvA), Fast and Reliable Online Learning to Rank for Information Retrieval

2013-21, Sander Wubben (UvT), Text-to-text generation by monolingual machine translation

2013-22, Tom Claassen (RUN), Causal Discovery and Logic

2013-23, Patricio de Alencar Silva (UvT), Value Activity Monitoring

2013-24, Haitham Bou Ammar (UM), Automated Transfer in Reinforcement Learning


2013-26, Alireza Zarghami (UT), Architectural Support for Dynamic Homecare Service Provisioning

2013-27, Mohammad Huq (UT), Inference-based Framework Managing Data Provenance

2013-28, Frans van der Sluis (UT), When Complexity becomes Interesting: An Inquiry into the Information eXperience

2013-29, Iwan de Kok (UT), Listening Heads


2013-31, Dinh Khoa Nguyen (UvT), Blueprint Model and Language for Engineering Cloud Applications

2013-32, Kamakshi Rajagopal (OUN), Networking For Learning; The role of Networking in a Lifelong Learner’s Professional Development

2013-33, Qi Gao (TUD), User Modeling and Personalization in the Microblogging Sphere

2013-34, Kien Tjin-Kam-Jet (UT), Distributed Deep Web Search


2013-36, Than Lam Hoang (TUE), Pattern Mining in Data Streams

2013-37, Dirk Borner (OUN), Ambient Learning Displays
2013-38, Eelco den Heijer (VU), Autonomous Evolutionary Art

2013-39, Joop de Jong (TUD), A Method for Enterprise Ontology based Design of Enterprise Information Systems

2013-40, Pim Nijssen (UM), Monte-Carlo Tree Search for Multi-Player Games

2013-41, Jochem Liem (UVA), Supporting the Conceptual Modelling of Dynamic Systems: A Knowledge Engineering Perspective on Qualitative Reasoning

2013-42, Leon Planken (TUD), Algorithms for Simple Temporal Reasoning

2013-43, Marc Bron (UVA), Exploration and Contextualization through Interaction and Concepts

2014

2014-01, Nicola Barile (UU), Studies in Learning Monotone Models from Data

2014-02, Fiona Tuliyano (RUN), Combining System Dynamics with a Domain Modeling Method

2014-03, Sergio Raul Duarte Torres (UT), Information Retrieval for Children: Search Behavior and Solutions

2014-04, Hanna Jochmann-Mannak (UT), Websites for children: search strategies and interface design - Three studies on children's search performance and evaluation

2014-05, Jurriaan van Reijsen (UU), Knowledge Perspectives on Advancing Dynamic Capability

2014-06, Damian Tamburri (VU), Supporting Networked Software Development

2014-07, Arya Adriansyah (TUE), Aligning Observed and Modeled Behavior

2014-08, Samur Araujo (TUD), Data Integration over Distributed and Heterogeneous Data Endpoints

2014-09, Philip Jackson (UvT), Toward Human-Level Artificial Intelligence: Representation and Computation of Meaning in Natural Language

2014-10, Ivan Salvador Razo Zapata (VU), Service Value Networks

2014-11, Janneke van der Zwaan (TUD), An Empathic Virtual Buddy for Social Support

2014-12, Willem van Willigen (VU), Look Ma, No Hands: Aspects of Autonomous Vehicle Control

2014-13, Arlette van Wissen (VU), Agent-Based Support for Behavior Change: Models and Applications in Health and Safety Domains

2014-14, Yangyang Shi (TUD), Language Models with Meta-information

2014-15, Natalya Mogles (VU), Agent-Based Analysis and Support of Human Functioning in Complex Socio-Technical Systems: Applications in Safety and Healthcare

2014-16, Krystyna Milian (VU), Supporting trial recruitment and design by automatically interpreting eligibility criteria
2014-17, Kathrin Dentler (VU), Computing healthcare quality indicators automatically: Secondary Use of Patient Data and Semantic Interoperability

2014-18, Mattijs Ghijsen (UVA), Methods and Models for the Design and Study of Dynamic Agent Organizations

2014-19, Vinicius Ramos (TUE), Adaptive Hypermedia Courses: Qualitative and Quantitative Evaluation and Tool Support

2014-20, Mena Habib (UT), Named Entity Extraction and Disambiguation for Informal Text: The Missing Link

2014-21, Kassidy Clark (TUD), Negotiation and Monitoring in Open Environments

2014-22, Marieke Peeters (UU), Personalized Educational Games - Developing agent-supported scenario-based training

2014-23, Eleftherios Sidirougos (UvA/CWI), Space Efficient Indexes for the Big Data Era

2014-24, Davide Ceolin (VU), Trusting Semi-structured Web Data

2014-25, Martijn Lappenschaar (RUN), New network models for the analysis of disease interaction

2014-26, Tim Baarslag (TUD), What to Bid and When to Stop


2014-28, Anna Chmielowiec (VU), Decentralized k-Clique Matching

2014-29, Jaap Kabbedijk (UU), Variability in Multi-Tenant Enterprise Software

2014-30, Peter de Cock (UVT), Anticipating Criminal Behaviour

2014-31, Leo van Moergestel (UU), Agent Technology in Agile Multiparallel Manufacturing and Product Support

2014-32, Naser Ayat (UvA), On Entity Resolution in Probabilistic Data

2014-33, Tesfa Tegegne (RUN), Service Discovery in eHealth

2014-34, Christina Manteli(VU), The Effect of Governance in Global Software Development: Analyzing Transactive Memory Systems.


2014-36, Joos Buijs (TUE), Flexible Evolutionary Algorithms for Mining Structured Process Models

2014-37, Maral Dadvar (UT), Experts and Machines United Against Cyberbullying

2014-38, Danny Plass-Oude Bos (UT), Making brain-computer interfaces better: improving usability through post-processing
2014-39, Jasmina Maric (UvT), Web Communities, Immigration, and Social Capital
2014-42, Carsten Eijckhof (CWI/TUD), Contextual Multidimensional Relevance Models
2014-43, Kevin Vlaanderen (UU), Supporting Process Improvement using Method Increments
2014-45, Birgit Schmitz (OUN), Mobile Games for Learning: A Pattern-Based Approach
2014-46, Ke Tao (TUD), Social Web Data Analytics: Relevance, Redundancy, Diversity
2014-47, Shangsong Liang (UVA), Fusion and Diversification in Information Retrieval

2015
2015-01, Niels Netten (UvA), Machine Learning for Relevance of Information in Crisis Response
2015-02, Faiza Bukhsh (UvT), Smart auditing: Innovative Compliance Checking in Customs Controls
2015-03, Twan van Laarhoven (RUN), Machine learning for network data
2015-04, Howard Spoelstra (OUN), Collaborations in Open Learning Environments
2015-05, Christoph Bosch (UT), Cryptographically Enforced Search Pattern Hiding
2015-06, Farideh Heidari (TUD), Business Process Quality Computation - Computing Non-Functional Requirements to Improve Business Processes
2015-07, Maria-Hendrike Peetz (UvA), Time-Aware Online Reputation Analysis
2015-08, Jie Jiang (TUD), Organizational Compliance: An agent-based model for designing and evaluating organizational interactions
2015-09, Randy Klaassen (UT), HCI Perspectives on Behavior Change Support Systems
2015-10, Henry Hermans (OUN), OpenU: design of an integrated system to support lifelong learning
2015-11, Yongming Luo (TUE), Designing algorithms for big graph datasets: A study of computing bisimulation and joins
2015-12, Julie M. Birkholz (VU), Modi Operandi of Social Network Dynamics: The Effect of Context on Scientific Collaboration Networks
2015-13, Giuseppe Procaccianti (VU), Energy-Efficient Software
2015-14, Bart van Straalen (UT), A cognitive approach to modeling bad news conversations
2015-15, Klaas Andries de Graaf (VU), Ontology-based Software Architecture Documentation

2015-16, Changyun Wei (UT), Cognitive Coordination for Cooperative Multi-Robot Teamwork

2015-17, Andre van Cleeff (UT), Physical and Digital Security Mechanisms: Properties, Combinations and Trade-offs

2015-18, Holger Pirk (CWI), Waste Not, Want Not! - Managing Relational Data in Asymmetric Memories

2015-19, Bernardo Tabuenca (OUN), Ubiquitous Technology for Lifelong Learners

2015-20, Lois Vanhee (UU), Using Culture and Values to Support Flexible Coordination

2015-21, Sibren Fetter (OUN), Using Peer-Support to Expand and Stabilize Online Learning

2015-22, Zhemin Zhu (UT), Co-occurrence Rate Networks

2015-23, Luit Gazendam (VU), Cataloguer Support in Cultural Heritage


2015-25, Steven Woudenberg (UU), Bayesian Tools for Early Disease Detection

2015-26, Alexander Hogenboom (EUR), Sentiment Analysis of Text Guided by Semantics and Structure

2015-27, Sandor Heman (CWI), Updating compressed column stores

2015-28, Janet Bagorogoza (TiU), KNOWLEDGE MANAGEMENT AND HIGH PERFORMANCE; The Uganda Financial Institutions Model for HPO

2015-29, Hendrik Baier (UM), Monte-Carlo Tree Search Enhancements for One-Player and Two-Player Domains

2015-30, Kiavash Bahreini (OU), Real-time Multimodal Emotion Recognition in E-Learning

2015-31, Yakup Koc (TUD), On the robustness of Power Grids

2015-32, Jerome Gard (UL), Corporate Venture Management in SMEs

2015-33, Frederik Schadd (TUD), Ontology Mapping with Auxiliary Resources

2015-34, Victor de Graaf (UT), Gesocial Recommender Systems


2016

2016-01, Syed Saiden Abbas (RUN), Recognition of Shapes by Humans and Machines
2016-02, Michiel Christiaan Meulendijk (UU), Optimizing medication reviews through decision support: prescribing a better pill to swallow

2016-03, Maya Sappelli (RUN), Knowledge Work in Context: User Centered Knowledge Worker Support

2016-04, Laurens Rietveld (VU), Publishing and Consuming Linked Data

2016-05, Evgeny Sherkhonov (UVA), Expanded Acyclic Queries: Containment and an Application in Explaining Missing Answers

2016-06, Michel Wilson (TUD), Robust scheduling in an uncertain environment

2016-07, Jeroen de Man (VU), Measuring and modeling negative emotions for virtual training

2016-08, Matje van de Camp (TiU), A Link to the Past: Constructing Historical Social Networks from Unstructured Data

2016-09, Archana Nottamkandath (VU), Trusting Crowdsourced Information on Cultural Artefacts

2016-10, George Karafotias (VUA), Parameter Control for Evolutionary Algorithms

2016-11, Anne Schuth (UVA), Search Engines that Learn from Their Users

2016-12, Max Knobbout (UU), Logics for Modelling and Verifying Normative Multi-Agent Systems

2016-13, Nana Baah Gyan (VU), The Web, Speech Technologies and Rural Development in West Africa - An ICT4D Approach

2016-14, Ravi Khadka (UU), Revisiting Legacy Software System Modernization

2016-15, Steffen Michels (RUN), Hybrid Probabilistic Logics - Theoretical Aspects, Algorithms and Experiments

2016-16, Guangliang Li (UVA), Socially Intelligent Autonomous Agents that Learn from Human Reward

2016-17, Berend Weel (VU), Towards Embodied Evolution of Robot Organisms

2016-18, Albert Merono Penuela (VU), Refining Statistical Data on the Web

2016-19, Julia Efremova (Tu/e), Mining Social Structures from Genealogical Data

2016-20, Daan Odijk (UVA), Context & Semantics in News & Web Search

2016-21, Alejandro Moreno Celleri (UT), From Traditional to Interactive Playspaces: Automatic Analysis of Player Behavior in the Interactive Tag Playground

2016-22, Grace Lewis (VU), Software Architecture Strategies for Cyber-Foraging Systems

2016-23, Fei Cai (UVA), Query Auto Completion in Information Retrieval

2016-24, Brend Wanders (UT), Repurposing and Probabilistic Integration of Data; An Iterative and data model independent approach
2016-25, Julia Kiseleva (TU/e), Using Contextual Information to Understand Searching and Browsing Behavior

2016-26, Dilhan Thilakaratne (VU), In or Out of Control: Exploring Computational Models to Study the Role of Human Awareness and Control in Behavioural Choices, with Applications in Aviation and Energy Management Domains

2016-27, Wen Li (TUD), Understanding Geo-spatial Information on Social Media

2016-28, Mingxin Zhang (TUD), Large-scale Agent-based Social Simulation - A study on epidemic prediction and control

2016-29, Nicolas Honing (TUD), Peak reduction in decentralised electricity systems - Markets and prices for flexible planning

2016-30, Ruud Mattheij (UvT), The Eyes Have It

2016-31, Mohammad Khelghati (UT), Deep web content monitoring

2016-32, Eelco Vriezekolk (UT), Assessing Telecommunication Service Availability Risks for Crisis Organisations

2016-33, Peter Bloem (UVA), Single Sample Statistics, exercises in learning from just one example

2016-34, Dennis Schunselaar (TUE), Configurable Process Trees: Elicitation, Analysis, and Enactment

2016-35, Zhaochun Ren (UVA), Monitoring Social Media: Summarization, Classification and Recommendation

2016-36, Daphne Karreman (UT), Beyond R2D2: The design of nonverbal interaction behavior optimized for robot-specific morphologies

2016-37, Giovanni Sileno (UvA), Aligning Law and Action - a conceptual and computational inquiry

2016-38, Andrea Minuto (UT), MATERIALS THAT MATTER - Smart Materials meet Art & Interaction Design

2016-39, Merijn Bruijnes (UT), Believable Suspect Agents; Response and Interpersonal Style Selection for an Artificial Suspect

2016-40, Christian Detweiler (TUD), Accounting for Values in Design

2016-41, Thomas King (TUD), Governing Governance: A Formal Framework for Analysing Institutional Design and Enactment Governance

2016-42, Spyros Martzoukos (UVA), Combinatorial and Compositional Aspects of Bilingual Aligned Corpora

2016-43, Saskia Koldijk (RUN), Context-Aware Support for Stress Self-Management: From Theory to Practice

2016-44, Thibault Sellam (UVA), Automatic Assistants for Database Exploration

2016-45, Bram van de Laar (UT), Experiencing Brain-Computer Interface Control
2016-46, Jorge Gallego Perez (UT), Robots to Make you Happy

2016-47, Christina Weber (UL), Real-time foresight - Preparedness for dynamic innovation networks

2016-48, Tanja Buttler (TUD), Collecting Lessons Learned

2016-49, Gleb Polevoy (TUD), Participation and Interaction in Projects. A Game-Theoretic Analysis

2016-50, Yan Wang (UVT), The Bridge of Dreams: Towards a Method for Operational Performance Alignment in IT-enabled Service Supply Chains

2017

2017-01, Jan-Jaap Oerlemans (UL), Investigating Cybercrime

2017-02, Sjoerd Timmer (UU), Designing and Understanding Forensic Bayesian Networks using Argumentation

2017-03, Daniel Harold Telgen (UU), Grid Manufacturing; A Cyber-Physical Approach with Autonomous Products and Reconfigurable Manufacturing Machines

2017-04, Mrunal Gawade (CWI), MULTI-CORE PARALLELISM IN A COLUMN-STORE

2017-05, Mahdieh Shadi (UVA), Collaboration Behavior

2017-06, Damir Vandic (EUR), Intelligent Information Systems for Web Product Search

2017-07, Roel Bertens (UU), Insight in Information: from Abstract to Anomaly

2017-08, Rob Konijn (VU), Detecting Interesting Differences: Data Mining in Health Insurance Data using Outlier Detection and Subgroup Discovery

2017-09, Dong Nguyen (UT), Text as Social and Cultural Data: A Computational Perspective on Variation in Text

2017-10, Robby van Delden (UT), (Steering) Interactive Play Behavior

2017-11, Florian Kunneman (RUN), Modelling patterns of time and emotion in Twitter #anticipointment

2017-12, Sander Leemans (TUE), Robust Process Mining with Guarantees

2017-13, Gijs Huisman (UT), Social Touch Technology - Extending the reach of social touch through haptic technology

2017-14, Shoshannah Tekofsky (UvT), You Are Who You Play You Are: Modelling Player Traits from Video Game Behavior

2017-15, Peter Berck, Radboud University (RUN) Memory-Based Text Correction

2017-16, Aleksandr Chuklin (UVA), Understanding and Modeling Users of Modern Search Engines

2017-17, Daniel Dimov (UL), Crowdsourced Online Dispute Resolution
2017-18, Ridho Reinanda (UVA), Entity Associations for Search

2017-19, Jeroen Vuurens (TUD), Proximity of Terms, Texts and Semantic Vectors in Information Retrieval

2017-20, Mohammadbashir Sedighi (TUD), Fostering Engagement in Knowledge Sharing: The Role of Perceived Benefits, Costs and Visibility

2017-21, Jeroen Linssen (UT), Meta Matters in Interactive Storytelling and Serious Gaming (A Play on Worlds)

2017-22, Sara Magliacane (VU), Logics for causal inference under uncertainty

2017-23, David Graus (UVA), Entities of Interest--- Discovery in Digital Traces

2017-24, Chang Wang (TUD), Use of Affordances for Efficient Robot Learning

2017-25, Veruska Zamborlini (VU), Knowledge Representation for Clinical Guidelines, with applications to Multimorbidity Analysis and Literature Search

2017-26, Merel Jung (UT), Socially intelligent robots that understand and respond to human touch

2017-27, Michiel Joosse (UT), Investigating Positioning and Gaze Behaviors of Social Robots: People's Preferences, Perceptions and Behaviors

2017-28, John Klein (VU), Architecture Practices for Complex Contexts

2017-29, Adel Alhuraibi (UVT), From IT-Business Strategic Alignment to Performance: A Moderated Mediation Model of Social Innovation, and Enterprise Governance of IT

2017-30, Wilma Latuny (UVT), The Power of Facial Expressions

2017-31, Ben Ruijl (UL), Advances in computational methods for QFT calculations

2017-32, Thaer Samar (RUN), Access to and Retrievability of Content in Web Archives

2017-33, Brigit van Loggem (OU), Towards a Design Rationale for Software Documentation: A Model of Computer-Mediated Activity

2017-34, Maren Scheffel (OUN), The Evaluation Framework for Learning Analytics

2017-35, Martine de Vos (VU), Interpreting natural science spreadsheets

2017-36, Yuanhao Guo (UL), Shape Analysis for Phenotype Characterisation from High-throughput Imaging

2017-37, Alejandro Montes Garcia (TUE), WiBAF: A Within Browser Adaptation Framework that Enables Control over Privacy

2017-38, Alex Kayal (TUD), Normative Social Applications

2017-39, Sara Ahmadi (RUN), Exploiting properties of the human auditory system and compressive sensing methods to increase noise robustness in ASR
2017-40, Altaf Hussain Abro (VUA), Steer your Mind: Computational Exploration of Human Control in Relation to Emotions, Desires and Social Support For applications in human-aware support systems

2017-41, Adnan Manzoor (VUA), Minding a Healthy Lifestyle: An Exploration of Mental Processes and a Smart Environment to Provide Support for a Healthy Lifestyle

2017-42, Elena Sokolova (RUN), Causal discovery from mixed and missing data with applications on ADHD datasets

2017-43, Maaike de Boer (RUN), Semantic Mapping in Video Retrieval

2017-44, Garm Lucassen (UU), Understanding User Stories - Computational Linguistics in Agile Requirements Engineering

2017-45, Bas Testerink (UU), Decentralized Runtime Norm Enforcement

2017-46, Jan Schneider (OU), Sensor-based Learning Support

2017-47, Yie Yang (TUD), Crowd Knowledge Creation Acceleration

2017-48, Angel Suarez (OU), Collaborative inquiry-based learning

2018

2018-01, Han van der Aa (VUA), Comparing and Aligning Process Representations

2018-02, Felix Mannhardt (TUE), Multi-perspective Process Mining

2018-03, Steven Bosems (UT), Causal Models For Well-Being: Knowledge Modeling, Model-Driven Development of Context-Aware Applications, and Behavior Prediction

2018-04, Jordan Janeiro (TUD), Flexible Coordination Support for Diagnosis Teams in Data-Centric Engineering Tasks

2018-05, Hugo Huurdeman (UVA), Supporting the Complex Dynamics of the Information Seeking Process

2018-06, Dan Ionita (UT), Model-Driven Information Security Risk Assessment of Socio-Technical Systems

2018-07, Jieting Luo (UU), A formal account of opportunism in multi-agent systems

2018-08, Rick Smetsers (RUN), Advances in Model Learning for Software Systems

2018-09, Xu Xie (TUD), Data Assimilation in Discrete Event Simulations

2018-10, Julienka Mollee (VUA), Moving forward: supporting physical activity behavior change through intelligent technology

2018-11, Mahdi Sargolzaei (UVA), Enabling Framework for Service-oriented Collaborative Networks

2018-12, Xixi Lu (TUE), Using behavioral context in process mining

2018-13, Seyed Amin Tabatabaei (VUA), Using behavioral context in process mining: Exploring the added value of computational models for increasing the use of renewable energy in the residential sector
2018-14, Bart Joosten (UVT), Detecting Social Signals with Spatiotemporal Gabor Filters
2018-15, Naser Davarzani (UM), Biomarker discovery in heart failure
2018-16, Jaebok Kim (UT), Automatic recognition of engagement and emotion in a group of children
2018-17, Jianpeng Zhang (TUE), On Graph Sample Clustering
2018-18, Henriette Nakad (UL), De Notaris en Private Rechtspraak
2018-19, Minh Duc Pham (VUA), Emergent relational schemas for RDF
2018-20, Manxia Liu (RUN), Time and Bayesian Networks
2018-21, Aad Slootmaker (OUN), EMERGO: a generic platform for authoring and playing scenario-based serious games
2018-22, Eric Fernandes de Mello Araujo (VUA), Contagious: Modeling the Spread of Behaviours, Perceptions and Emotions in Social Networks
2018-23, Kim Schouten (EUR), Semantics-driven Aspect-Based Sentiment Analysis
2018-24, Jered Vroon (UT), Responsive Social Positioning Behaviour for Semi-Autonomous Telepresence Robots
2018-25, Riste Gligorov (VUA), Serious Games in Audio-Visual Collections
2018-26, Roelof de Vries (UT), Theory-Based And Tailor-Made: Motivational Messages for Behavior Change Technology
2018-27, Maikel Leemans (TUE), Hierarchical Process Mining for Scalable Software Analysis
2018-28, Christian Willemse (UT), Social Touch Technologies: How they feel and how they make you feel
2018-29, Yu Gu (UVT), Emotion Recognition from Mandarin Speech
2018-30, Wouter Beek (VU), The "K" in "semantic web" stands for "knowledge": scaling semantics to the web

2019

2019-01, Rob van Eijk (UL), Web privacy measurement in real-time bidding systems. A graph-based approach to RTB system classification
2019-02, Emmanuelle Beauxis- Aussalet (CWI, UU), Statistics and Visualizations for Assessing Class Size Uncertainty
2019-03, Eduardo Gonzalez Lopez de Murillas (TUE), Process Mining on Databases: Extracting Event Data from Real Life Data Sources
2019-04, Ridho Rahmadi (RUN), Finding stable causal structures from clinical data
2019-05, Sebastiaan van Zelst (TUE), Process Mining with Streaming Data
2019-06, Chris Dijkshoorn (VU), Nichesourcing for Improving Access to Linked Cultural Heritage Datasets
2019-07, Soude Fazeli (TUD), Recommender Systems in Social Learning Platforms
2019-08, Frits de Nijs (TUD), Resource-constrained Multi-agent Markov Decision Processes
2019-09, Fahimeh Alizadeh Moghaddam (UVA), Self-adaptation for energy efficiency in software systems
2019-10, Qing Chuan Ye (EUR), Multi-objective Optimization Methods for Allocation and Prediction
2019-11, Yue Zhao (TUD), Learning Analytics Technology to Understand Learner Behavioral Engagement in MOOCs
2019-12, Jacqueline Heinerman (VU), Better Together
2019-13, Guanliang Chen (TUD), MOOC Analytics: Learner Modeling and Content Generation
2019-14, Daniel Davis (TUD), Large-Scale Learning Analytics: Modeling Learner Behavior & Improving Learning Outcomes in Massive Open Online Courses
2019-15, Erwin Walraven (TUD), Planning under Uncertainty in Constrained and Partially Observable Environments
2019-16, Guangming Li (TUE), Process Mining based on Object-Centric Behavioral Constraint (OCBC) Models
2019-17, Ali Hurriyetoglu (RUN), Extracting actionable information from microtexts
2019-18, Gerard Wagenaar (UU), Artefacts in Agile Team Communication
2019-19, Vincent Koeman (TUD), Tools for Developing Cognitive Agents
2019-20, Chide Groenouwe (UU), Fostering technically augmented human collective intelligence
2019-21, Cong Liu (TUE), Software Data Analytics: Architectural Model Discovery and Design Pattern Detection
2019-22, Martin van den Berg (VU), Improving IT Decisions with Enterprise Architecture
2019-23, Qin Liu (TUD), Intelligent Control Systems: Learning, Interpreting, Verification
2019-24, Anca Dumitrache (VU), Truth in Disagreement- Crowdsourcing Labeled Data for Natural Language Processing
2019-25, Emiel van Miltenburg (UVT), Pragmatic factors in (automatic) image description
2019-26, Prince Singh (UT), An Integration Platform for Synchromodal Transport
2019-27, Alessandra Antonaci (OUN), The Gamification Design Process applied to (Massive) Open Online Courses
2019-28, Esther Kuindersma (UL), Cleared for take-off: Game-based learning to prepare airline pilots for critical situations

2019-29, Daniel Formolo (VU), Using virtual agents for simulation and training of social skills in safety-critical circumstances

2019-30, Vahid Yazdanpanah (UT), Multiagent Industrial Symbiosis Systems

2019-31, Milan Jelisavcic (VUA), Alive and Kicking: Baby Steps in Robotics

2019-32, Chiara Sironi (UM), Monte-Carlo Tree Search for Artificial General Intelligence in Games

2019-33, Anil Yaman (TUE), Evolution of Biologically Inspired Learning in Artificial Neural Networks

2019-34, Negar Ahmadi (TUE), EEG Microstate and Functional Brain Network Features for Classification of Epilepsy and PNES

2019-35, Lisa Facey-Shaw (OUN), Gamification with digital badges in learning programming

2019-36, Kevin Ackermans (OUN), Designing Video-Enhanced Rubrics to Master Complex Skills

2019-37, Jian Fang (TUD), Database Acceleration on FPGAs

2019-38, Ákos Kádár (TiU), Learning visually grounded and multilingual representations

2020-01, Armon Toubman (UL), Calculated Moves: Generating Air Combat Behaviour

2020-02, Macos de Paula Bueno (UL), Unraveling Temporal Processes using Probabilistic Graphical Models

2020-03, Mostafa Deghani (UVA), Learning with Imperfect Supervision for Language Understanding

2020-04, Maarten van Gompel (RUN), Context as Linguistic Bridges

2020-05, Yulong Pei (TU/e), On Local and Global Graph Structure Mining

2020-06, Preethu Rose Anish (UT), Stimulating Architectural Thinking during Requirements Elicitation – An Approach and Tool Support
Requirements specifications often lack the details needed by software architects to make informed architectural decisions. Lacking such details, the architects either make assumptions or go back to business analysts for clarifications or conduct additional stakeholder interviews. This may result in incorrect requirements and project delays.

The goal of this dissertation is to enhance communication between the business analysts and software architects by introducing a knowledge base with architectural knowledge to be used by business analysts during requirements elicitation. Using an empirical approach, we have developed an initial version of such a knowledge base.

Preethu Rose Anish (1982) was born in New Delhi, India. She completed her Bachelor’s degree in Science from Pune University in 2003 and Master’s degree in Computer Application from Bangalore University in 2006. After studies, she started her professional career at the first research unit of Tata Consultancy Services (TCS) - an Indian multinational information technology services, consulting and business solutions company. During her 13 plus years in TCS Research, working as a developer, researcher and scientist, she collaborated on research projects with many renowned academics on requirements engineering, empirical research methods and applications of machine learning. During these collaborations, Preethu discovered a personal drive to understand in depth the scientific rigor with which these academics work. This influenced her to apply for a PhD position in University of Twente, The Netherlands. Being an external PhD student with a full time job, she tried her best to balance the act between her project commitments in TCS and her PhD research work. On the 11\textsuperscript{th} of March 2020, Preethu defended her PhD dissertation at the University of Twente, The Netherlands.