July 2013

Maps of Lessons Learnt in Requirements Engineering

Ibtehal Noorwali
The University of Western Ontario

Supervisor
Nazim H. Madhavji
The University of Western Ontario

Graduate Program in Computer Science

A thesis submitted in partial fulfillment of the requirements for the degree in Master of Science

MAPS OF LESSONS LEARNT IN REQUIREMENTS ENGINEERING

(Thesis format: Monograph)

by

Ibtehal Noorwali

Graduate Program in Computer Science

A thesis submitted in partial fulfillment

of the requirements for the degree of

Master of Science

The School of Graduate and Postdoctoral Studies

The University of Western Ontario

London, Ontario, Canada

Abstract

Both researchers and practitioners have emphasized the importance of learning from past experiences and its consequential impact on project time, cost, and quality. However, from the survey we conducted of requirements engineering (RE) practitioners, over 70% of the respondents stated that they seldom use RE lessons in the RE process, though 85% of these would use such lessons if readily available. Our observation, however, is that RE lessons are scattered, mainly implicitly, in the literature and practice, which obviously, does not help the situation. We, therefore, present "maps" of RE lessons which would highlight weak (dark) and strong (bright) areas of RE (and hence RE theories). Such maps would thus be: (a) a driver for research to "light up" the darker areas of RE and (b) a guide for practice to benefit from the brighter areas. To achieve this goal, we populated the maps with over 200 RE lessons elicited from literature and practice using a systematic literature review and survey. The results show that approximately 80% of the elicited lessons are implicit and that approximately 70% of the lessons deal with the elicitation, analysis, and specification RE phases only. The RE Lesson Maps, elicited lessons, and the results from populating the maps provide novel scientific groundings for lessons learnt in RE as this topic has not yet been systematically studied in the field.

Keywords: lessons learnt in requirements engineering, lesson maps, software engineering, software quality, empirical study.

Acknowledgements

First and foremost, my sincere thanks go to Professor Nazim Madhavji for his continuous guidance, supervision, and encouragement during the course of my research.

I would also like to thank my friends and colleagues: Avi Walia, Quanjun Yin, and Rashed Nekvi for their support, helpful comments and invaluable input throughout my studies.

I would like to thank the Department of Computer Science at the University of Western Ontario for the computing infrastructure and facilities provided during my graduate studies.

Special thanks to Saudi Arabia's Ministry of Higher Education, The Saudi Arabian Cultural Bureau, and Umm Al-Qura University for fully funding my graduate studies.

Last but not least, I would like to thank my mother, father, siblings, and close friends for their never-ending support, love and encouragement, which without, I would not have been able to complete this work.

Publications

Reference: Noorwali, I. and Madhavji, N. H. (2013a). Lessons learnt in requirements engineering: A research preview. In Doerr, J. and Opdahl, A., editors, Proceedings of the 19th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 13), LNCS 7830, pages 119124, Essen, Germany. Springer-Verlag Berlin Heidelberg.

Abstract: "Those who cannot remember the past are condemned to repeat it" – George Santayana. From the survey we conducted of requirements engineering (RE) practitioners, over 70% seldom use RE lessons in the RE process, though 85% of these would use such lessons if readily available. Our observation, however, is that, RE lessons are scattered, mainly implicitly, in the literature and practice, which, obviously, does not help the situation. Approximately 90% of the survey participants stated that not utilising RE lessons has significant negative impact on product quality, productivity, project delays and cost overruns. We propose "maps (or profiles) of RE lessons which, once populated, would highlight weak (dark) and strong (bright) areas of RE (and hence RE theories). Such maps would thus be: (a) a driver for research to light up the darker areas of RE and (b) a guide for practice to benefit from the brighter areas. The key contribution of this work is the concept of "maps" of RE lessons.

This REFSQ 2013 paper is jointly authored by Noorwali and Madhavji and the contribution by both authors is significant. For the full paper, please see Appendix E.

Table of Contents

List of Figures

List of Tables

List of Appendices

Chapter 1: Introduction

1.1 Motivation

The importance of lessons learnt (LL) has been stressed upon in the software engineering literature [Abdelhamid and Madnick, 1990, Basili et al., 2002, Boehm, 2006] as well as in the software industry [Kaner et al., 2001]. The benefits of lessons learnt for improving processes, products, and services have been widely cited, e.g.: "we provide a set of lessons learned that should help future groups to learn from and improve on a quarter of century of experiences" [Basili et al., 2002] and "past knowledge is birthed from lessons learned that should be made available and accessible to different parties for different occasions" [Lee, 2008].

With requirements engineering (RE) as an established field, one can assume that lessons are being learnt in this field as well. However, our analysis of the requirements engineering literature indicates that lessons learnt are rarely explicitly described and often they are lacking contextual information. This analysis also indicates that many RE lessons are scattered and implicitly described in the literature and so this makes it arduous to apply readily in software projects.

Our main concern is about the possible negative impact of the state of LL in the field of RE. The concerns are twofold: a. Is it that the scientific and practicing RE communities are not widely and effectively using LL from the collective past experiences? b. There is not much scientific basis and technological support for LL.

To validate this concern, especially about RE LL in industry, we conducted a survey of 50 RE practitioners from the US, Canada [Noorwali and Madhavji, 2012] (see Appendix A), Europe and Asia. The findings are alarming. Around 82% of the respondents indicated that RE LL are important for their organisations' RE processes. However, 72% stated that RE LL are only occasionally/hardly ever used in their organisations. Moreover, 90% indicated that RE LL are shared within the organisation through informal sharing. The statistics for the difficulty of finding, gathering, eliciting and getting access to RE LL from various sources were as follows:

Table 1.1: Percentage of respondents who indicated that accessing lessons learnt is 'easy' and 'very easy' from various sources

Source	Easy	Very Easy
Development projects	24%	7%
People	21%	23%
Websites	18%	3%
Books	32%	0%
Technical reports	16%	0%

Around 85% of the respondents indicated that they will use RE LL in their projects if they were made readily available. Finally, the percentage of the respondents who said that the level of impact is significant for the different attributes is significant, is as follows: Productivity loss: 85%, project delays: 90%, cost overruns: 93%, product quality problems: 95%, repeating mistakes: 98%, opportunities lost: 60%, project failures: 62%, Customer dissatisfaction: 83%.

1.2 Purpose of the Study

The findings from section 1.1 present the following two questions that this study seeks to answer:

- What is the state of LL in RE?
- What scientific and technological basis can be created to promote the use of LL in RE?

Therefore, the purpose of this study is:

To understand and determine the state of lessons learnt in RE and promote their use by creating a scientific basis for the structuring and organization of lessons embodied in the concept of "Lesson Maps" and populating them with lessons elicited from the literature and practice. We, therefore, conduct a solution-building and knowledge-seeking study in an attempt to achieve this purpose. The solution building study involves presenting a formal representation of a

RE lesson and concept of RE Lesson Map. The knowledge-seeking empirical study includes a systematic literature review of two years of RE literature and a survey to gather lessons from practice. We chose two years of RE literature across several conferences and journals that amount to approximately 740 papers. This results in significant amount of arduous work analysing these papers to elicit lessons learnt. For the purpose of this thesis, the analysis of two years of literature is considered adequate proof-of-concept study.

1.3 Significance and Originality of Research

The populated lesson maps are anticipated to:

- Improve current RE processes by utilising the lesson maps in projects. This in turn, is anticipated to have an effect on project costs, time, and quality.
- Generate new RE theories by exposing weaker (darker) and stronger (brighter) areas of RE.
- Promulgate further research across the different RE areas to brighten up the 'darker' areas.

While much work has been done in the area of lessons learnt outside and within the software engineering field, no scientific work has been done on LL in RE. To our knowledge, the concept of a RE Lesson Map is a novel one and an attempt to provide a significant body of knowledge of RE lessons has not yet been made.

1.4 Thesis Structure

The thesis is structured as follows: Chapter 2 discusses the related works from the research literature; Chapter 3 presents the general concept of a lesson and lesson map including the definitions, representation and attributes; Chapter 4 is where we apply the concepts of a lesson and lesson map to RE specifically; Chapter 5 describes the empirical study we conducted to elicit lessons from the literature and practice detailing the research questions, research methods and threats to validity; Chapter 6 includes the results of the empirical study (i.e. the elicited lessons from literature and practice); Chapter 7 gives examples of some populated RE Lesson Maps; Chapter 8 discusses the implications of this study on research and practice; Chapter 9 summarizes the thesis by discussing the limitations of the study and future work in this area.

Chapter 2: Related Work

The concept of lessons learnt has received significant attention in many fields such as management [Lee, 2008], education [Bodycott and Walker, 2001], medicine [Rogers et al., 2001], policymaking [McLaughlin, 1987], and biotechnology [Olson and Lyles, 2011], to name a few. Both researchers and practitioners have dealt with lessons learnt in their respective fields in different ways. Their focus was mainly on the following two dimensions: (i) collecting and sharing the lessons they learnt through mediums such as publications and technical reports [Olson and Lyles, 2011, Bodycott and Walker, 2001, Rogers et al., 2001], and (ii) studying and proposing methods, processes and solutions for dealing with and managing lessons learnt in projects, teams, and organizations. Based on our extensive literature search, we categorised the study of LL into three disciplines: the study of lessons learnt in non-software engineering fields, software engineering in general, and finally lessons learnt in requirements engineering, discussed in sections 2.1, 2.2, and 2.3 respectively. Section 2.4 presents the analysis and research gap.

2.1 Lessons Learnt in Non-Software Engineering Fields

Our literature analysis led us to the realisation that LL have received more attention in non-SE fields and in SE in general than in RE specifically. Within the dimension of managing LL, Huber [Huber, 1991] analyses the relative literature on organisational learning and assesses the current state of research on each of the following four organisational learning constructs: (1) knowledge acquisition, (2) information distribution, (3) information interpretation, and (4) organisational memory. Knowledge acquisition refers to the formal methods used to acquire information or knowledge such as customer surveys, research and development activities, performance reviews, and analyses of competitor's products. He categorises the processes through which organisations acquire information into five processes: (1) congenital learning, (2) experiential learning, (3) vicarious learning, (4) grafting, and (5) searching. Information distribution determines both the occurrence and breadth of organizational learning. Information interpretation is defined as “the process through which information is given meaning” and “the process of translating events and developing shared understandings and conceptual schemes”. Finally, organisational memory refers to the methods through which information is stored and retrieved. He concludes that, while much work has been done in the area of information distribution, there is a dearth of research in the areas of knowledge acquisition, information interpretation and organizational memory.

On a similar note, Wellman [Wellman, 2007] describes the advantages and disadvantages of the four methods that organisations use to manage lessons learned: Culture, Old Pros, Archives, and Processes. Culture is the set of behaviours and operating principles that nearly everyone knows but which are not written. Old Pros refer to those people in an organisation who have been around long enough for them to gather a great deal of experience. Archives are used to capture and retrieve lessons and Formal Processes can be used as both a repository and disseminator of lessons learned. In summary, the best approach is to use the four methods effectively while recognizing the complexity and challenge of doing so.

Lee [Lee, 2008] discusses some of the essential functions that lessons learnt provide project managers: navigation tool; preventive measures; decision making tool; reserves allocation; portfolio and program management; heuristic methods for solving a problem. He also discusses the reasons behind ineffective lessons learnt and provides guidelines to make lessons learnt more effective for project managers:

- Make sure lessons learnt are done in each phase as deliverables.
- Ensure the lessons learnt pre-format form has all the right prompts to request right details.
- Avoid brainstorming. Organise and solicit input using a structure and chronological approach. This is to enable individuals’ thought processes to be in line with the chrono-

logical events of the project and to think in an orderly fashion that is connected from the beginning through closing.

- Pre-determine which areas of the project are to be assessed and use quantification methods to determine the level of success or failure. Use empirical evidence if necessary.
- Ensure a lessons learnt session is included as one of its agenda during the kick off meeting and emphasises its importance.
- Ensure there is a checklist put in place to ensure each milestone, major deliverable or phase be accompanied by lessons learnt sessions.
- Find ways to bring the team together. Make lessons learnt a compulsory task to be completed before the project team and stakeholders are disbanded from the project. This can be done offsite in conjunction with success celebration.
- Make lessons learnt include key stakeholders and various representatives to ensure the views are comprehensive, balanced, accurate and verified. Enforce attendance list is taken to ensure all participate.
- Use a filter list to check if the information shared is relevant for other similar projects, can be used for project improvement, has generic lessons that can be applied across different industries and has positive impact for the future. Do not waste time on trivial information.
- Avoid referring to specific individuals and NEVER disclose their names. Remember an individual or a group mistake may not be perceived as their mistake. It could come from a chain of reaction.
- A project success is the result of team effort. Culminate the project closure by giving credit and commendation to groups or departments rather than specific individuals.
- Archive lessons learnt documents in an accessible repository.

Patton [Patton, 2001] argues that in order for a lesson to be of high quality, it must be representative of principles extracted from various sources and applicable to future actions. He presents a set of questions for generating high-quality lessons learnt:

- What is meant by a "lesson?"
- What is meant by "learnt?"
- By whom was the lesson learnt?
- What is the evidence supporting each lesson?
- What is the evidence the lesson was learnt?
- What are the contextual boundaries around the lesson (that is, under what conditions does it apply)?
- Is the lesson specific, substantive, and meaningful enough to guide practice in some concrete way?
- Who else is likely to care about this lesson?
- What evidence will they want to see?
- How does this lesson connect with other lessons?

We can see from the previous paragraphs that the focus of the researchers and practitioners was on providing solutions, methods and guidelines for eliciting and managing lessons within projects and organisations. However, outside the fields of management and organisational science, most of the work is concentrated on sharing and disseminating the lessons learnt with others within the field, as discussed in the next paragraph.

Bodycott and Walker [Bodycott and Walker, 2001] discuss the lessons learnt while teaching in a higher education setting in Hong Kong. They focused on issues of language and communication, social and cultural distance, the effect of hierarchy, and teaching strategies. Rogers et al. [Rogers et al., 2001] present the results from applying the lessons learnt from first decade of minimal access surgery. The results were positive as the lesson learnt have helped in focusing attention on technical skills training. In the policymaking field, McLaughlin [McLaughlin, 1987] shares and integrates the lessons learnt from the first and second generation of policy implementation in an attempt to frame the conceptual and instrumental challenge for the third generation of implementation analysts. As a final example, Olson and Lyles [Olson and Lyles, 2011] provide lessons from the antibody industry and apply them to the biofuels industry in an attempt to aid those in biofuels licensing.

2.2 Lessons Learnt in Software Engineering

In the field of Software Engineering (SE), lessons learnt have also received some attention. As in other fields, research on lessons learnt in SE can be roughly categorized into two categories. The first is research related to discovering and sharing lessons learnt from specific cases with a wider audience. The literature in this category can be further divided into two subcategories: (i) an explicit mention of lessons learnt and (ii) an implicit mention of lessons learnt. The same categorisation applies to the literature on requirements engineering. However, because we are interested in lessons learnt in requirements engineering in particular, we will only discuss the literature on explicit lessons learnt in software engineering in general, then both subcategories will be discussed extensively in the RE section.

The literature that explicitly states and discusses lessons learnt from projects, organisations, and individual experiences is found throughout the SE literature but not in abundance. Examples of such studies include Basili's et al. [Basili et al., 2002] paper in which the authors "give some lessons learned on what they did right, what they did wrong, and what others can learn from our experiences" from 25 years of process improvement at NASA's Software Engineering Laboratory. The lessons are organised and presented in a chronological order. In a similar fashion, Boehm [Boehm, 2006] tries to identify the major lessons he learnt from his experience in the software engineering industry during the 20th and 21st centuries.

While the previous two papers put forth lessons learnt of a more general nature, other papers share lessons learnt from a specific project or application of specific tools and/or methods. In a study by both Basili and Boehm and others [Reifer et al., 2004], the authors share twelve important lessons about COTS-Based Systems (CBS) maintenance. They state each lesson along with its source and implications, thus giving it a rough structure as we will discuss in Chapter 3.

Dick and Woods [Dick and Woods, 1997] describe the positive and negative lessons learnt from the application of the VDM and B methods for the specification, design, and implementation of an administration subsystem. In another study, the experience factory [Basili et al., 1994] is tailored and used in an Experience Management System (EMS) developed by the authors then applied to three experience bases in three different organisations [Lindvall et al., 2001]. The authors then share their lessons learnt and experience from the applying the EMS to three organisations: Q-Labs, the Fraunhofer Center, Maryland (FC-MD) and Johnson Control, Inc. (JCI).

The second category of research on lessons learnt in SE includes studies that propose and develop methods, processes, and prototypes, and tools for lessons learnt. Given the software engineering community's tool-building mindset, the work done in this category is more extensive than in the first category. The literature is rich with studies on tools, methods, prototypes, frameworks and systems for eliciting, storing, and managing lessons learnt.

Weber has done extensive work in the area of Lessons Learnt Systems. In one study, Weber and others [Weber et al., 2001] survey forty LL systems found on the WWW that are maintained by various government and other organisations and detail their capabilities and limitations. They first categorised the processes that LL systems are designed to support then they categorised the systems themselves. They identify five main subprocess that a LL system must support: (1) collect, (2) verify, (3) store, (4) disseminate, and (5) reuse. The 'collect' subprocess can be performed in five different ways: passively, reactively, after an action, proactively actively and interactively. The verification subprocess is usually done by experts to ensure the correctness, completeness and uniqueness of the lessons. Storing lessons refers to the representation (e.g. level of abstraction) and indexing of lessons, formatting, and the repository's framework. Lesson representations can be structured, semi- structured, or in different media. Lesson dissemination can be done in 5 ways: (1) passive dissemination, (2) active casting, (3) broadcasting, (4) proactive dissemination and, (5) reactive dissemination. Finally, the reuse subprocess includes three categories: (1) browsable recommendation, (2) executable recommendation, and (3) outcome reuse. A categorisation of available LL systems is then presented. The authors categorised the systems according to content, roles, duration, orientation, architecture, organisation type, confidentiality and size. Some of the surveyed LL systems include: Berkeley Lab LL Program, DOE Corporate LL Collections, Air Combat Command Center for LL, Marine Corps LL System, Eureka (Xerox), and RECALL.

In a subsequent study by Weber and Aha [Weber and Aha, 2002], they introduce, describe, and empirically evaluate the monitored distribution (MD) approach for the active delivery of lessons learnt in the context of a decision support tool for planning military missions. The results of the empirical evaluation show that this just-in-time information delivery approach, embedded in a decision support system (DSS) for plan authoring, significantly improved plan execution performance measures.

On a similar note, Andrade et al. [Andrade et al., 2007] provide a framework for safety-critical lessons learnt and present a prototype lesson learnt system for safety-critical software with a focus on failures and negative experiences. In a more recent study by the same first author [Andrade et al., 2013], an architectural model for software testing lesson learnt systems is proposed. Birk and Tautz [Birk and Tautz, 1998], in their technical report, describe how to structure and represent lessons learnt. They also present a technical infrastructure for storing, disseminating, and applying them. Their method is developed and evaluated through three case studies: (i) Systematic Inspections, (ii) Requirements Engineering, and (iii) GQM-Based Measurement.

Abdel-Hamid and Madnick [Abdelhamid and Madnick, 1990] develop a post mortem diagnostic tool for conducting a postmortem diagnostic analysis of a software project to learn from failures and discern lessons. They analyse both the failures they would like to avoid in the future and the successes they want to improve upon from NASA's DE-A software project, which was established to design, implement, and test a software system for processing telemetry data and providing attitude determination and control for the DE-A satellite. Their approach, however, is concerned with the managerial aspect of software engineering rather than technical one.

Different structured approaches for eliciting and identifying lessons have been proposed, such as the approximate reasoning-based approach [Vandeville and Shaikh, 1999], Case-Based Reasoning (CBR) approach [Sary and Mackey, 1995], and the Experience Factory Framework [Basili et al., 1994].

2.3 Lessons Learnt in Requirements Engineering

Although lessons learnt have received attention in software engineering in general, unfortunately, the same case cannot be said to hold true for lessons learnt in requirements engineering in particular. As we've seen in Section 1, the results from our survey indicate that lessons learnt haven't received much attention in requirements engineering practice either; around 82% of the respondents indicated that requirements engineering LL are important for their organisation's

RE processes. However, 72% stated that RE LL are only occasionally/hardly ever used in their organisation [Noorwali and Madhavji, 2012]. Our findings from the literature search support our survey's results. We will discuss the related work done on lessons learnt in requirements engineering using the same categorisations used for discussion in the SE section.

With regard to sharing lessons learnt from projects, organisations, and personal experiences, some papers discuss some lessons in RE explicitly [Daneva, 2004, Damian, 2007, Berenbach, 2012], but lessons learnt are mainly implicit in the literature [Lauesen and Kuhail, 2012, Gibson et al., 2011, Kong and Hayes, 2011, Boulila et al., 2011]. Out of the 209 lessons we elicited from two year literature, only 47 lessons were mentioned explicitly (i.e., 22%), as we will discuss at length in Chapter 5.

Under the category of LL supporting tools, processes and methods, software requirements patterns [Withall, 2007] are the closest to lessons learnt. A requirements software pattern is "a guide to writing a particular type of requirement. It explains how to tackle that type of requirement, what to say, what to worry about, and it suggests additional requirements you might need to write too". It enables requirements engineers to reuse requirements that are recurrent by providing them with a systematic means to specify new requirements. A recurring lesson learnt can eventually become a requirement pattern if it satisfies a pattern's criteria.

The technical infrastructure described in section 2.2 for storing, disseminating, and applying lessons has been developed using a requirements engineering case study among three others to develop and evaluate the method, thus, rendering it suitable for use in a requirements engineering setting [Birk and Tautz, 1998].

2.4 Analysis and Research Gap

The concept of lessons learnt is common throughout a variety of disciplines [Lee, 2008, Bodycott and Walker, 2001, Rogers et al., 2001, Olson and Lyles, 2011], as we've seen in earlier sections, denoting its importance. Researchers' and practitioners' emphasis on the importance of learning from past experiences [Boehm, 2006, Abdelhamid and Madnick, 1990] further supports the argument for lessons learnt. Moreover, the results from the survey we conducted indicate that approximately 90% of the survey participants stated that not utilising RE lessons has significant negative impact on product quality, productivity, project delays and cost overruns [Noorwali and Madhavji, 2012].

Despite this emphasis on the importance of lessons learnt and the abundance of research done on the topic in non-SE disciplines and in SE, lessons learnt in requirements engineering are scarce, scattered and mainly implicit, making the state of lessons learnt in RE rather vague. This implicit and vague state of lessons make accessing and reusing lessons learnt in RE processes a difficult task. In addition, current LL-oriented tools, systems and processes are mainly organisational-level making it difficult to utilise these tools and methods on an individual and team level. The focus of this study, therefore, is to fill in this gap by proposing the concept of maps of lessons learnt in requirements engineering and populating these maps with lessons elicited from the RE literature and practice in an attempt to understand the current state of lessons learnt across the different RE subprocesses and to provide practitioners with a solution that is anticipated to have impact on project costs, time, and quality. To our knowledge, such an attempt has not yet been made and our concept of RE Lesson Maps is novel.

Chapter 3: The Concept of a Lesson and Lesson Map

This chapter discusses the concepts of a lesson and lesson map in a generic sense. In section 3.1, we discuss the concept of a lesson by listing the definitions of a lesson from various dictionaries and literary sources (section 3.1.1) and describing its representation (section 3.1.2). Section 3.2, presents the concept of a lesson map.

3.1 Lesson Concept

The term 'lesson' has numerous definitions. In this section, we list the definitions from well-known sources. The definitions are categorised based on the following: (i) literature definitions and (ii) dictionary definitions. For our research, we use the definitions from the literature as they are more encompassing and relevant to our work.

3.1.1 Lesson Definition

Literature Definitions

- A "good work practice" or innovative approach that is captured and shared to promote repeat application. A lesson learned may also be an adverse work practice or experience that is captured and shared to avoid recurrence [DOE-STD-7501-99, 1999].

- A lesson learnt is the knowledge acquired from an innovation or an adverse experience that causes a worker or an organization to improve a process or activity to work safer, more efficiently, or with higher quality [Weber et al., 2001].

- A lesson learnt is a recorded experience of value; a conclusion drawn from analysis of feedback information on past and/or current programs, policies, systems and processes. Lessons may show successes or innovative techniques, or they may show deficiencies or problems to be avoided. A lesson may be: 1. An informal policy or procedure. 2. Something you want to repeat. 3. A solution to a problem, or a corrective action. 4. How to avoid repeating an error. 5. Something you never want to do (again) [Weber et al., 2001].

- A lesson learnt is a knowledge or understanding gained by experience. The experience may be positive, as in a successful test or mission, or negative, as in a mishap or failure. Successes are also considered sources of lessons learnt. A lesson must be significant in that it has a real or assumed impact on operations; valid in that is factually and technically correct; and applicable in that it identifies a specific design, process, or decision that reduces or eliminates the potential for failures and mishaps, or reinforces a positive result [Secchi et al., 1999].
- Lessons learned (LL) are part of knowledge gained from experiences during a project and in the post mortem phase. It was initially conceived of guidelines, checklists or tips of what went right or wrong in every event worth mentioning, in projects activities [Jalili et al., 2011].

Dictionary Definitions

- An occurrence, example, rebuke, or punishment, that serves or should serve to warn or encourage. [Pearsall and Trumble, 1995]
- An experience or event that serves as a warning or encouragement. [Oxford, 2013b]
- An experience, especially an unpleasant one, that makes you more careful in the future. [Summers, 1987]
- Lesson is an experience which teaches you how to behave better in a similar situation in the future. [Cambridge, 2013a]
- Lesson is a punishment or bad experience that teaches you something. [Macmillan, 2013a]

- Lesson is an experience, especially an unpleasant one, that somebody can learn from so that it does not happen again in the future. [Oxford, 2013a]
- Something that you have learned or that must be learned. [Oxford, 2004]
- Something, especially a piece of wisdom, learnt by study or experience. [Allen, 2003]
- An instructive or warning example. [Allen, 2003]
- An instructive example. [Merriam-Webster, 2013]
- A thing inculcated by experience or study. [Pearsall and Trumble, 1995]
- That which is learned or taught by an express effort; instruction derived from precept, experience, observation, or deduction; a precept; a doctrine; as, to take or give a lesson in drawing. A smooth and pleasing lesson. [Webster, 1913]
- A lesson is a useful piece of information learnt though experience. [Cambridge, 2013b]
- Something that needs to be learned (or the event through which it is learned) for the sake of one's safety, well-being, etc. [Webster, 2013]
- Something from which useful knowledge or principles can be learnt. [Collins, 2013]
- Something from which a person learns or should learn; an instructive example. [Dictionary.com, 2013a]
- A useful piece of practical wisdom acquired by experience or study. [Dictionary.com, 2013a]
- A lesson is a thing learnt by experience. [Oxford, 2013b]
- Something that you learn from life, an event, or an experience. [Macmillan, 2013a]
- Lesson is something learned by study or experience. [Merriam-Webster, 2013]

3.1.2 Lesson Representation

Many researchers have given formal representations [Birk and Tautz, 1998, Weber et al., 2001, Wangenheim et al., 1998] for a lesson using 'attributes' that are suitable for a specific discipline or task. Researchers have used different terms to refer to a lesson's attributes. Some have used the term attribute [Weber et al., 2001, Andrade et al., 2007], descriptor [Andrade et al., 2013], and feature or field [Sary and Mackey, 1995]. In this study, we will use the term attribute to refer to the 'fields used to contain the different pieces of information related to a lesson that are needed to fully represent a lesson learnt '.

It is important to note that organisations use different formats (i.e., different combinations of attributes) to represent lessons in their LL systems [Weber et al., 2001]. This indicates that a universal representation for lessons learnt does not exist, but representations and attributes are tweaked according to the needs of the organisation, domain, and/or task employing lessons learnt.

In this section, we list the different attributes used to represent lessons across different disciplines, mainly software engineering. The attributes are categorised into two categories: (i) generic attributes, (ii) extended attributes. These attributes are also used for representing lessons in RE. The attributes and their values with relation to RE are detailed in Chapter 4.

Generic Attributes

The generic attributes are needed to represent a lesson regardless of organization, field, and/or task. While the list is not exhaustive of what is available in the literature, we have chosen those attributes that are relevent to most contexts in software engineering and specifically RE, and left out those that are meant for a specific domain or area. Each attribute is supported by a citation to the corresponding literature.

- Lesson [DOE-STD-7501-99, 1999]: This is the text describing the lesson learnt.

- Rationale/Justification [Wangenheim et al., 1998]: This explains the reason behind the given lesson and why it is important.

- Impact [Weber et al., 2001]: The anticipated or observed impact of the stated lesson that may be either positive or negative.

- Repeatability [Andrade et al., 2007]: This attribute specifies the properties of the lesson which makes it repeatable and the context in which it is repeatable including any contexts in which it may *not* be repeatable. This gives a level of confidence to the lesson. A lesson derived from several experiments, for example, has a higher level of confidence than a lesson derived from a single case study.

- Source [Reifer et al., 2004]: The source of the lesson can include: experience reports, project memos, presentation slides, the minutes from project post-mortem meetings or GQM feedback sessions, the results from project or technology evaluations, interviews, questionnaires, surveys, brainstorming meetings, capturing of ad-hoc statements during project work [Birk and Tautz, 1998], people, literature, books, etc.

- Target Object [Birk and Tautz, 1998]: The target object of a lesson learnt is the artifact that the lesson learnt is about. It may be a process, a product, a technique or method, some policy, etc.

- Type [Weber et al., 2001]: Type indicates whether a lesson is positive (one learnt from a successful past experience) or a negative lesson (one learnt from an unsuccessful past experience).

- Application domain [Wangenheim et al., 1998]: the application domain from which a specific lesson was derived. Examples of the values of this attribute in our case, for example, are critical systems, security systems, information system, etc.

- Project size [Birk and Tautz, 1998]: the project size indicates the size of the project from which the lesson was derived (e.g., number of people, number of LOC or function-points, person-years, etc.).

- Software process [Sommerville, 2011, Andrade et al., 2013]: Some researchers have referred to this as a development model. We will use the term 'process' in our study. This attribute indicates the type(s) of software process(es) used in the project from which the lesson has been derived. A software process is defined as: A systemic approach that is used in software engineering. It is a sequence of activities that lead to the production of a software product (e.g. agile, waterfall, incremental development, spiral, code-and-fix, rapid prototyping, unified process, extreme programming etc.).

- Phase [Andrade et al., 2007, Andrade et al., 2013]: While some use the term 'activity' or 'life cycle phase', we will use 'phase' here. In software engineering, possible values for a phase may be system requirements definition, system design, software requirements definition, software design, human-interface design, implementation, modelling/simulation, testing, deployment, usage, maintenance. In RE, phases refer to the different RE activities such as elicitation, specification, validation, etc.

- Practice: A practice is a specific way for achieving a particular goal. For example, in RE, a practice is a specific way for achieving a particular software development goal and it may be applied to one or more subprocesses (e.g., prototyping, using checklists, use case modelling, prioritisation via voting, tracing using a requirements tool, win-win model of negotiation, elicitation via interviews, using a prioritisation framework, etc.) [Sommerville and Sawyer, 1997]. In testing, on the other hand, example practices include using reviews and inspection and specifying formal entry and exit criteria [Chillarege, 1999]. While the literature on the representation of lessons learnt does not discuss the use of 'practice' as an attribute, we've added it to our list of attributes as it will be useful in trying to identify all the lessons under, for example, prototyping for elicitation. Thus, playing an important role in achieving our goal of understanding the state of lessons learnt across RE. Likewise, it can be useful in other disciplines.

- Project date: This indicates the date of the project from which the lesson has been derived. This is important because an older lesson may not be as effective as a newer one in a certain context.
- Recording date [Wangenheim et al., 1998, Andrade et al., 2013]: The date at which the lesson was created/recorded.
- Organisation name [Wangenheim et al., 1998]: The name of the organization from which the lesson was derived.

Extended Attributes

For our study, we added five more 'extended' attributes that, we believe, are needed in our case to help us achieve our research objective of understanding the current state of lessons learnt in RE.

- Expression: Expression indicates whether a lesson was explicitly expressed as a lesson learnt in the literature, or the context and surrounding literature had to be analysed to elicit the lesson. This will aid us in determining the awareness of the RE community towards lessons learnt. The results will be discussed in detail in Chapter 6.
- Year: Indicates the year of the publication from which the lesson was elicited. This is essential for comparing the state of lessons learnt in the different years.
- Journal: The name of the journal from which the lesson was elicited.
- Conference: The name of the conference in which the publication containing the lesson learnt was presented.
- Workshop: The name of the workshop in which the publication containing the lesson learnt was presented. These last three attributes will help us determine where the majority of lessons learnt are coming from.

Extended attributes can be removed and added depending on the need for them in a specific context. Other extended attributes that may be added depending on the environment include: department; project name; author, etc.

3.2 Lesson Map Concept

3.2.1 Map Definition

To shed any ambiguity on the use of the term ‘map’, we provide a number of definitions from a variety of dictionaries. A ‘map’ is defined as:

- An image of an area that shows the positions of things such as countries, rivers, cities, and streets. Used about other types of images that show the positions of things. [Macmillan, 2013b]
- A maplike delineation, representation, or reflection of anything. [Dictionary.com, 2013b]
- A diagram or collection of data showing the spatial arrangement or distribution of something over an area [Oxford, 2013c]

In our case, a ‘map depicts an arrangement and distribution of lessons learnt over an “area” according to specific attributes. An area is denoted by the set of attributes of interest. For example, “elicitation” intersected with “real-time” systems denotes a set of lessons learnt relevant to “elicitation” and “real-time” systems. This idea facilitates a user to define his or her own areas of interest in selecting lessons learnt. In this respect, such a “map” is not a “road map” that shows one the path to follow to reach a destination; rather, it is akin to maps in atlases.

3.2.2 The Concept of a Lesson Map

With reference to the definitions of a map in section 3.2.1, a lesson map is based on two types of elements:

- Content. The content of the map is the lessons learnt. In our study, the content is the lessons elicited from literature and practice (discussed in Chapter 5 and 6).

- Context. The context includes specific attributes selected by the user. In our specific case, the context consists of any combination of the context attributes we presented in section 3.1: source, target object, type, application domain, project size, practice, phase, software process, project date, recording date, organisation name, rationale, impact, and/or repeatability.

In principle, therefore, it is possible to produce many permutations of lesson maps, e.g.: practices; practices X phases; practices X phases X application domains; project size X phases X sources; application domain X process type; etc. The actual rendering of a map in various permutations is a matter of technological support, which is outside the scope of this work.

After populating a map with some lessons learnt, it can be indicative of the state of lessons learnt in RE (in a project, organisation, body of knowledge, etc.) identified by scarce (dark) and abundant (bright) areas of the map (see Table 3.1). Let us assume that context attributes X and Y (selected by the user) are values for the attributes 'phase' and 'practice', respectively, where they are depicted here as a table but could be depicted in another form (e.g. hierarchically). LL1, LL2, etc., are the lessons learnt relating to specific phases and practices. Examples of dark areas (low number of lessons learnt) are: X3Y2 and X4Y2 and of bright areas (high frequency of lessons learnt) are: X1Y1and X2Y3.

Table 3.1: An example map with context attributes X and Y.

	X1	X2	X3	X4	...
Y1	LL1, LL2, LL3	LL7, LL8	LL13	LL6	
Y2	LL16	LL4, LL5			
Y3	LL17, LL18	LL9, LL10, LL11, LL12	LL14, LL15, LL6	LL6	

Chapter 4: The Concept of Lessons and Lesson Maps in RE

In this chapter we apply the concepts explained in Chapter 3 to the RE field. Section 4.1 discusses the representation of a RE Lesson and the values of the attributes in detail. In section 4.2, we describe the RE Lesson Map. Finally, in section 4.3, we describe the validation processes used for validating the RE Lesson and RE Lesson Map.

4.1 RE Lesson

A RE Lesson has the attributes discussed in Chapter 3: lesson, rationale/justification, impact, repeatability, source, target object, type, application domain, project size, software process, phase, practice, project date, recording date, organisation name, expression, year, journal, conference, workshop.

In this section, we describe the possible values for each attribute specific to the RE field. It is important to note that the values of the attributes may fall into one of the following categories: (i) attributes that have a set of countable, fixed, concise values (e.g., type, phase, expression), which we will call 'category A', (ii) attributes that have uncountable, long, explanatory values (e.g., rationale/justification, impact), which we will call 'category B' and (iii) attributes that do not have a fixed set of values, as new values may emerge depending on the domain, project, organization, etc., but are usually relatively short (e.g., project size, application domain, practice, project date, organisation name), which we will call 'category C'.
Note: In the following, the values that have one, two, or three citations were found in only that number of papers. The values that have three or more citations were found in many papers, but we are listing only a few of them here.

- Lesson: The values of this attribute are of category B; it contains the text of the lesson itself.

- Rationale/Justification: The values of this attribute are of category B; it contains the text that explains the reason behind the given lesson and why it is important.

- Impact: The values of this attribute are of category B; it contains the text that describes the anticipated or observed impact of the stated lesson that may be either positive or negative.

- Source: The values of this attribute are of category C. In Chapter 3, we stated that the source of the lesson can include: experience reports, project memos, presentation slides, the minutes from project post-mortem meetings or GQM feedback sessions, the results from project or technology evaluations, interviews, questionnaires, surveys, brainstorming meetings, capturing of ad-hoc statements during project work, people, literature, books, etc. However, because our lessons come mainly from the literature, we used this attribute to indicate whether the lesson came from a controlled experiment, case study, industrial experience, etc. The following list enumerates all the sources we have found during our lesson elicitation process. It is important to note, however, that the sources have been gathered and used according to the exact terms used in the source papers.

  - Controlled experiment [Niknafs and Berry, 2012, Calefato et al., 2012, El-Sharkawy and Schmid, 2011, Pires et al., 2011]
  - Experiment [Schmidt et al., 2012, Ohashi et al., 2011, Chen et al., 2011, Kong and Hayes, 2011]
  - Evaluating experiment [Yang et al., 2012, Yi et al., 2012, Gross et al., 2012]
  - Exploratory experiment [Li et al., 2011]
  - Quasi experiment [Uusitalo et al., 2011, Gonzales and Leroy, 2011]
  - Case study [Behnam et al., 2012, Penzenstadler and Eckhardt, 2012, Borges et al., 2011, Kof and Penzenstadler, 2011]

  - Evaluating case study [Charrada et al., 2012, Gordon and Breaux, 2012, Maxwell et al., 2012a]
  - Evaluating exploratory case study [Niu and Mahmoud, 2012]
  - Confirmatory case study [Shaker et al., 2012]
  - Retrospective case study analysis [Morales-Ramirez et al., 2012]
  - Pilot case study [Veerappa and Letier, 2011]
  - Explanatory case study [Bjarnason et al., 2011]
  - Industrial experience [Tawhid et al., 2012, Chernak, 2012, Nolan et al., 2011, Vogl et al., 2011]
  - Workshop [Massacci et al., 2012]
  - Survey [Anh et al., 2012, Poort et al., 2012, Wever and Maiden, 2011]
  - Questionnaire [van Tuijl et al., 2011, Gross and Doerr, 2012, Bjarnason et al., 2012]
  - Pilot Study [Sharma and Biswas, 2012]
  - Illustrative example [Lutz et al., 2012, Cleland-Huang et al., 2012]
  - Simulation [Cleland-Huang et al., 2012]
  - Qualitative study [Sim and Alspaugh, 2011]
  - Systematic review [Dieste and Juristo, 2011]
  - End-user study [Kamalrudin et al., 2011]
  - Evaluating example [Merten et al., 2011]
  - Document analysis study [Reggio et al., 2011]
  - Field assessment [Daramola et al., 2011]

- Repeatability: The values of this attribute in our study are of category C. Because literature was the main source of our lessons, this attribute contains information from the paper on whether the results of the lesson, for example, are first time results of a controlled experiment or confirm existing results. It depends on the value of the 'source' attribute. Example values include: first-time case study results, first-time controlled experiment results, first-time industry results, 4 studies, etc.

- Target Object: The values of this attribute in our study are of category C. The target object of a lesson learnt is the artifact that the lesson learnt is about. It may be a process, a product, a technique or method, some policy, etc. The following list includes all the values we found during our study:
  - Technique/method [Berenbach, 2012, Teka et al., 2012, Markov et al., 2011, Jones, 2011]
  - Tool [Svensson et al., 2011, Mashkoor and Jacquot, 2011, Smialek et al., 2012, Hussain et al., 2012]
  - Policy [Gordon and Breaux, 2012, Maxwell et al., 2012a, Schmidt et al., 2011, Isaacs and Berry, 2011]
  - Artifact: requirements [Gross and Doerr, 2012]
  - RE analysts [Niknafs and Berry, 2012]
  - People [Wever and Maiden, 2011, Massey et al., 2011]
  - Language [Luna et al., 2011, Gordon and Breaux, 2011, Teruel et al., 2011, Pasquale and Spoletini, 2011]
- Type [Weber et al., 2001]: The values of this attribute are of category A. The values may be one of the following:
  - Positive [Weber et al., 2001]: A lesson learnt from a successful experience or study that yielded positive results.
  - Negative [Weber et al., 2001]: A lesson learnt from an unsuccessful experience or study that yielded negative results.

  - Neutral: A lesson that is neither positive nor negative, but the knowledge learnt can be actively utilised.

- Application domain: The values of this attribute in our study are of category C; it indicates the application domain from which a specific lesson was derived. The following application domains were derived from our study:

  - Automotive [Post et al., 2012]
  - Medical [Charrada et al., 2012]
  - Software intensive [Niu and Mahmoud, 2012]
  - Healthcare [Maxwell et al., 2012a]
  - Weather station and graph product line [Yi et al., 2012]
  - University [Niknafs and Berry, 2012]
  - Pharmaceutical industry [Sapkota et al., 2012]
  - Library [Sharma and Biswas, 2012]
  - Student registration and and grading [Sharma and Biswas, 2012]
  - DNA nanotechnology [Lutz et al., 2012]
  - Robotic systems [Cleland-Huang et al., 2012]
  - Aviation security [Braun et al., 2012]
  - Aviation transportation [Tawhid et al., 2012]
  - Banking [Chernak, 2012]
  - Chemical and power plants [Berenbach, 2012]
  - Service-based systems [Torres et al., 2012]
  - Drinking-water production [Engelsman and Wieringa, 2012]
  - Business [Hussain et al., 2012]

– Academic [Hussain et al., 2012]
– Air traffic management [Raspotnig and Opdahl, 2012]
– Socio-technical systems [Morales-Ramirez et al., 2012]
– IT [Berenbach, 2012]
– Online shopping system [Savio and P.C., 2012]
– Industrial automation plant management [Savio and P.C., 2012]
– Finance [Maxwell et al., 2012b]
– Telecommunication [Schneider et al., 2012]
– Safety-critical system [Alrajeh et al., 2012]
– Micro-survey mobile application [Zhu and Herrmann, 2012]
– Electronic and land registry system [Lauesen, 2012]
– Software project management [Wang et al., 2012]
– Drives [Hauksdottir et al., 2012]
– Solar [Hauksdottir et al., 2012]
– System of systems [Penzenstadler and Eckhardt, 2012]
– Software product line (SPL) [Wu et al., 2012]
– Home control devices [Loft et al., 2012]
– Energy [Berenbach et al., 2012]
– Mail sorting system [Berenbach et al., 2012]
– Pump systems [Borges et al., 2011]
– Steam boilers [Kof and Penzenstadler, 2011]
– Datacenter operations [Helferich and Mautsch, 2011]
– Event management [Mahaux et al., 2011]

- Public safety [Adam, 2011]
- Developer asset software platform [Markov et al., 2011]
- Transportation systems [Mashkoor and Jacquot, 2011]
- Information and communication technologies (ICT) [Pitula and Radhakrishna, 2011]
- Nuclear power plants [Uusitalo et al., 2011]
- High-frequency radio-based systems (FAS) [Dietsch et al., 2011]
- Ticket machines [Boulila et al., 2011]
- Aerospace [Nolan et al., 2011]
- Mobile applications [Vogl et al., 2011]
- Embedded systems [Bjarnason et al., 2011]
- Ambulance service systems [Heaven and Letier, 2011]
- Hearing solutions [Waldmann, 2011]
- Rail lock systems [Daramola et al., 2011]
- Steam boiler control systems [Daramola et al., 2011]
- Patient monitoring agents [Morandini et al., 2011]
- Washing machine managers [Morandini et al., 2011]
- ERP vendors [Salfischberger et al., 2011]
- Mobile handsets [Svensson et al., 2011]
- Mobile media management systems [Goulao et al., 2011]
- Office printers production [Marincic et al., 2011]
- Semiconductors [Chopra and Singh, 2011]
- Voting systems [Gibson et al., 2011]

- Project size: The values of this attribute in our study are of category C; it indicates the size of the project from which the lesson was derived (e.g., number of people in a project, number of LOC or function-points, person-years, etc.). Some researchers have chosen specific metrics to identify project size [Andrade et al., 2013]. We list here some examples of project size values we identified for their corresponding lessons during our study:
  - ‘Eleven full-text requirements documents’ [Yang et al., 2012].
  - ‘Six business requirements and a Java code base containing over 500 classes’ [Niu and Mahmoud, 2012].
  - ‘22 features, 196 feature pairs, 6 requires, and 5 excludes’ [Yi et al., 2012].
  - ‘400 technical requirements’ [Boutkova, 2011].
  - ‘The platform functionality included around 170 functions and was encapsulated in 17 CSD’ [Adam, 2011].
  - ‘The company employs about 20 people, of which three are dedicated software developers’ [Dietsch et al., 2011].
- Software process: The values of this attribute in our study are of category C. Although category A may seem more suitable for this attribute, we chose not to limit the values of this attribute to our list as organisations and teams may use new approaches or mixed approaches we have not listed. ‘Software process’ indicates the type(s) of software process(es) used in the project from which the lesson has been derived. It is defined as “a systemic approach that is used to in software engineering. It is a sequence of activities that leads to the production of a software product”. Values for the software process attribute include [Sommerville, 2011]:
  - Agile
  - Waterfall

- Iterative
- Spiral
- Incremental
- Scrum
- Rapid Prototyping
- Adaptive Software Development
- Feature Driven Development
- Extreme Programming
- The Rational Unified Process

- Phase: The values of this attribute in our study are of category A; it refers to the following RE activities [Kotonya and Sommerville, 1998]:
  - Elicitation
  - Analysis
  - Prioritisation
  - Negotiation
  - Specification
  - Documentation
  - Verification
  - Validation
  - Managing
- Practice: The values of this attribute in our study are of category C; it is a specific way for achieving a particular goal. For example, in RE a practice is a specific way for achieving a particular software development goal and it may be applied to one or more

subprocesses (e.g., prototyping, using checklists, use case modelling, prioritisation via voting, tracing using a requirements tool, win-win model of negotiation, elicitation via interviews, using a prioritisation framework, etc.) [Sommerville and Sawyer, 1997]. The following list includes the practices we have derived from our study:

- Tracing [Cleland-Huang et al., 2012, Engelsman and Wieringa, 2012, Gervasi and Zowghi, 2011, Markov et al., 2011]
- Modeling [Raspotnig and Opdahl, 2012, Berenbach et al., 2012, Markov et al., 2011, Mashkoor and Jacquot, 2011]
- Communication [Savio and P.C., 2012, Bjarnason et al., 2011, Waldmann, 2011]
- Prototyping [Poort et al., 2012, Atladottir et al., 2012, Gibson et al., 2011]
- Reuse [Hauksdottir et al., 2012, Ernst et al., 2011, Boutkova and Houdek, 2011]
- Using a prioritisation framework [Kukreja et al., 2012]
- Interviews [Morales-Ramirez et al., 2012, Dieste and Juristo, 2011]
- Brainstorming [Sakhnini et al., 2012, Erra and Scanniello, 2011, Boulila et al., 2011]
- Annotation [Hussain et al., 2012, Sapkota et al., 2012]
- Using use cases [Lauesen and Kuhail, 2012, Mahaux et al., 2011]
- Using specification pattern systems (SPS) [Post et al., 2012]
- Using text-based synchronous communication [Calefato et al., 2012]
- Automatic checking [Knauss and Schneider, 2012, Post et al., 2011]
- Using requirements patterns [Sharma and Biswas, 2012, Behnam et al., 2012, Daramola et al., 2012]
- Using PRISM model checker [Lutz et al., 2012]
- Using task descriptions [Lauesen and Kuhail, 2012]

- Using feature trees [Fricker and Schumacher, 2012]
- Using War Stories Approach [Sim and Alspaugh, 2011]
- Using social networks [Chopra and Singh, 2011]
- Using a framework [Asnar et al., 2011, Salfischberger et al., 2011, Sunindyo et al., 2011]
- Using model-driven development [Goulao et al., 2011]
- Using a modeling language [Morandini et al., 2011]
- Automatic tracing [Kong and Hayes, 2011]
- Using a specification language [Sharma and Biswas, 2011]
- Visualizing traceability information [Merten et al., 2011]
- Using observation techniques [Brill and Knauss, 2011]
- Using quantitative goal models [Heaven and Letier, 2011]
- Using software quality models [Carvallo and Franch, 2011]
- Assisted tracing [Dekhtyar et al., 2011]
- Using templates [Uusitalo et al., 2011, Rauf et al., 2011]
- Story-telling [Boulila et al., 2011]
- Using visual narratives [Dietsch et al., 2011]
- Using metrics [Massey et al., 2011]
- Using CPR analysis [Schmidt et al., 2011]
- Using storyboards [Sutcliffe et al., 2011]
- Using scenarios [Sutcliffe et al., 2011]
- Semi-automated checking [Kamalrudin et al., 2011]
- Protocol analysis [Dieste and Juristo, 2011]

- Project date: The values of this attribute in our study are of category C; it indicates the date of the project from which the lesson has been derived.
- Recording date: The values of this attribute in our study are of category C; it indicates the date at which the lesson was created/recorded.
- Organisation name: The values of this attribute in our study are of category C; it contains the name of the organization from which the lesson was derived.
- Expression: The values of this attribute are of category A. The values may be one of the following:
  - Explicit: The lesson was explicitly expressed as a lesson learnt in the literature. 'Experience' and 'learned from experience' were considered synonyms that indicated an explicit mention of lessons learnt.
  - Implicit: the context and surrounding literature had to be analysed to elicit the lesson.
- Year: The values of this attribute are of category A. The values may be one of the following:
  - 2011
  - 2012
- Journal: The values of this attribute are of category A. The values may be one of the following:
  - RE (Requirements Engineering Journal)
  - EMSE (Empirical Software Engineering Journal)
  - TSE (IEEE Transactions on Software Engineering Journal)

- Conference: The values of this attribute are of category A. The values may be one of the following:
  - IEEE RE (IEEE International Requirements Engineering Conference)
  - REFSQ (International Working Conference on Requirements Engineering: Foundation for Software Quality)
  - ICSE (International Conference on Software Engineering)
- Workshop: The values of this attribute are of category A. The values may be one of the following:
  - EmpiRE (Workshop on Empirical Requirements Engineering)
  - MoDRE (Workshop on Model-Driven Requirements Engineering)
  - REET (Workshop on Requirements Engineering Education and Training)
  - RELAW (Workshop on Requirements Engineering and Law
  - RePa (Workshop on Requirements Patterns)
  - RESS (Workshop on Requirements for Systems, Services, and Systems-of-Systems)
  - TwinPeaks (Workshop on Twin Peaks of Requirements Engineering)
  - MaRK (Workshop on Managing Requirements Knowledge)
  - Mere (Workshop on Multimedia and Enjoyable Requirements Engineering)
  - RE@Runtime (Workshop Requirements@run.time)
  - RESC (Workshop on Requirements Engineering for Social Computing)
  - REVOTE (Workshop on Requirements Engineering for E-Voting Systems)
  - REEW (Requirements Engineering Efficiency Workshop)
  - EPICAL (Workshop on Empirical Research in Requirements Engineering: Challenges and Solutions)

  - RE4SuSy (Requirements Engineering for Sustainable Systems)
  - RePrico (Requirements Prioritisation for Customer Oriented Development Workshop)
  - CreaRE (Creativity in Requirements Engineering Workshop)

The attributes we presented and described are the attributes we used to represent the lessons we derived from literature and practice. However, after eliciting a large number of lessons, we observed that some attributes were almost never filled and that there was a need for further attributes in order to better understand the context of the lesson. For our future work in lessons learnt, we plan to use the following attributes:

- Limitations/Side effects: Although we have an attribute for 'impact', a single attribute proved to be vague and confusing because some lessons may have a combination of both positive and negative effects. Therefore, the 'impact' attribute can be divided into positive impact and negative impact. The decision to apply a lesson or not would be then left to the user's discretion.

- Related lessons: Some lessons are only applicable if another lesson is applied, thus, making them 'lessons learnt on lessons learnt'. Therefore, it is important to add an attribute that specifies whether it is a sub-lesson or a parent-lesson and all the related lessons.

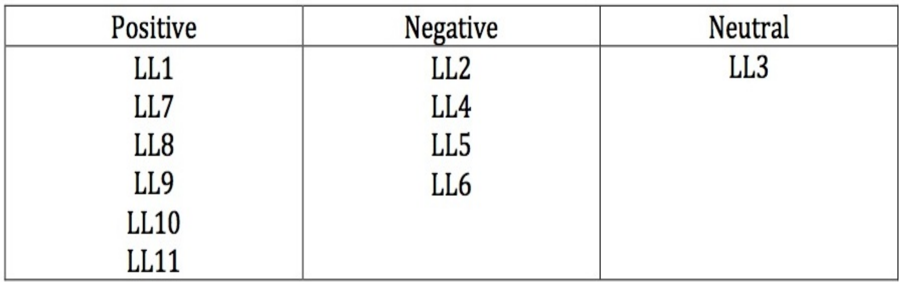
4.2 RE Lesson Map

In an attempt to create a discipline surrounding lessons learnt in RE, we apply the concepts of a map of lessons learnt that we presented to in Chapter 3 to RE. A RE lesson map has two elements:

- Content. The content of the map is the lessons elicited from literature and practice (discussed in Chapter 5 and 6).

- Context. The context is represented by the attributes and values discussed in section 4.1. Different permutations can be created. For example:
  - Type (see Figure 4.1). This Lesson Map depicts the distribution of lessons learnt (LL) according to lesson type (i.e., positive, negative and neutral).
  - Phase X Software Process (see Figure 4.2). This Lesson Map depicts the distribution of the lessons learnt (LL) related to the values of the 'Phase' and 'Software Process' attributes.
  - Target Object X Phase X Expression (see Figure 4.3). This Lesson Map depicts the distribution of the lessons learnt (LL) related to the values of the 'Target Object', 'Phase' and 'Expression' attributes.

The combination of attributes may take as many attribute values as possible (e.g. 4, 5, 6 or more attribute values). We, however, resorted to giving examples of three maps due to the difficulty in representing more complex combinations using a table format. With adequate tool support, the different renderings can be supported (see Appendix D).



Positive	Negative	Neutral
LL1 LL7 LL8 LL9 LL10 LL11	LL2 LL4 LL5 LL6	LL3

Figure 4.1: A Lesson Map with 'Type' attribute

4.3 Validation Processes of the RE Lesson and Lesson Map

The concepts of a lesson and lesson map in general and in RE have gone through several validation processes during our study. The concepts were continuously validated throughout the study by using expert evaluation. Four internal RE experts and one senior RE expert evaluated

	Elicitation	Analysis	Prioritisation	Negotiation	Specification	Documentation	Validation
Agile	LL1 LL2	LL5 LL6 LL7 LL8		LL41	LL20 LL21 LL22 LL26		LL39
Waterfall					LL24 LL25	LL13 LL14 LL15	LL37 LL38
Iterative					LL27 LL28		
Spiral			LL3 LL4				
Incremental	LL31 LL32 LL34	LL19		LL16 LL17			
Scrum					LL29		
Rapid Prototyping		LL44 LL45					LL36
Adaptive Software Development				LL9	LL23 LL35	LL11 LL12	
Feature Driven Development			LL42 LL43				
Extreme Programming					LL30		
The Rational Unified Process		LL10					LL40

Figure 4.2: A Lesson Map with ‘Phase’ and ‘Software Process’ attributes

	Elicitation		Analysis		Prioritisation		Negotiation		Specification		Documentation		Validation	
	Explicit	Implicit	Explicit	Implicit	Explicit	Implicit	Explicit	Implicit	Explicit	Implicit	Explicit	Implicit	Explicit	Implicit
Technique/Method	LL1 LL2	LL3 LL4 LL5 LL6 LL7	LL8										LL19 LL20	
Tool				LL9 LL10 LL11		LL31 LL32 LL33 LL34 LL35								LL15 LL16 LL17 LL18
Policy				LL26 LL27 LL28					LL37 LL38	LL36				
Artifact: requirement			LL29 LL30											
RE Analyst		LL24 LL25												
People				LL12			LL21 LL22 LL23							
Language				LL13 LL14							LL39 LL40	LL41 LL42 LL43		

Figure 4.3: A Lesson Map with 'Target Object' and 'Phase' and 'Expression' attributes

the concepts for their completeness, validity, and potential problems.

In addition, a research preview was published in the Proceedings of the 19th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ13) [Noorwali and Madhavji, 2013a], which was reviewed by four reviewers. The presentation of the paper led to further improvement through discussions and feedback from the participants at REFSQ.

Chapter 5: The Empirical Study on RE Lessons

In order to fulfill our study's overall research purpose of understanding and determining the state of lessons learnt in RE and promoting its use, we adopted a study method that is both solution-building and knowledge seeking. While the concept of "Lesson Maps" presented in Chapters 3 and 4 constituted the solution-building part of this study, this chapter describes the knowledge-seeking part of our study (i.e., empirical study).

We conducted two studies, a systematic literature review [Kitchenham, 2004] and survey, to elicit lessons from literature and practice, which in turn, was used to populate the lesson maps. This chapter describes the systematic literature review and survey we conducted, which includes the research goal and questions (section 5.1), the research methods used (section 5.2), and finally, the threats to validity (section 5.3).

5.1 Research Goal and Questions

In section 1.1 we discussed the results from an initial survey we conducted [Noorwali and Madhavji, 2012] (see Appendix A for survey). For convenience, we will restate the results here: we found that around 82% of the respondents indicated that RE LL are important for their organisation's RE processes. However, 72% stated that RE LL are only occasionally/hardly ever used in their organisations. The statistics for the difficulty of finding, gathering, eliciting and getting access to RE LL from various sources were as follows:

Table 5.1: Percentage of respondents who indicated that accessing lessons learnt is 'easy' and 'very easy' from various sources

Source	Easy	Very Easy
Development projects	24%	7%
People	21%	23%
Websites	18%	3%
Books	32%	0%
Technical reports	16%	0%
Peer-reviewed scietific literature	25%	0%

Around 85% of the respondents indicated that they will use RE LL in their projects if they were made readily available. These numbers indicate that accessing lessons learnt from the different sources is difficult and that LL would be used if made readily available. Therefore, this study aims to better understand where this difficulty is stemming from and provide a concrete step towards solving it by making lessons learnt readily available. Although the survey responses indicate that accessing lessons learnt is difficult from most sources, we have restricted our study to understanding the state of lessons learnt in the scientific literature due to time constraints. Thus, our overall research goal is:

"To determine the current state of lessons learnt in RE and to make these lessons learnt accessible and readily available to practitioners."

While the process of eliciting the lessons and populating the lesson maps will make these lessons accessible and readily available to practitioners, to understand the *current state* of lessons learnt in RE (both literature and practice), we have divided our research questions into two categories: (i) the first targets the RE literature and (ii) the second targets RE industry. The following research questions address the state of lessons learnt in the RE literature:

RQ 1.1 *What are the elicited lessons learnt from the RE literature?*

RQ 1.2 *How many lessons learnt are in the RE literature?*

RQ 1.3 *How are the lessons learnt expressed in the RE literature (explicitly or implicitly)?*

RQ 1.4 *Where (in which area of RE) are the lessons from the RE literature concentrated?*

RQ 1.5 *What is the quality of the lessons learnt elicited from the RE literature?*

In order to expand the sources from which we elicited lessons by including RE practice as well as the RE literature, we have set a number of research questions for this area:

RQ 2.1 *What are there lessons learnt from industry/practice?*

RQ 2.2 *Where (which area of RE) are the lessons from practice concentrated?*

RQ 2.3 *What is the quality of the lessons learnt elicited from industry/practice?*

RQ 2.4 *How do the lessons learnt from industry compare with the lessons learnt from literature?*

It is, however, important to note that the study addressing these questions is in its initial stages and the results we have received to date is small (details in section 5.2.2).

5.2 Research Methods

In order to investigate the aforementioned research questions, we conducted two knowledge-seeking empirical studies: (i) a systematic literature review (SLR) [Kitchenham, 2004] to elicit lessons from the literature and (ii) a survey [Yin, 2003] to elicit lessons from practice. In this section, we describe the research procedures in detail for the SLR (section 5.2.1) and survey (section 5.2.2).

5.2.1 Systematic Literature Review

According to Kitchenham, a systematic literature review is "a means of identifying, evaluating and interpreting all available research relevant to a particular research question, or topic area, or phenomenon of interest. Individual studies contributing to a systematic review are called primary studies; a systematic review is a form of secondary study" [Kitchenham, 2004]. The systematic literature review process includes three phases: (i) plan review, (ii) conduct review, and (iii) document review. Each phase embodies several stages (see Figure 5.1). The SLR we conducted followed the original guidelines provided by Kitchenham [Kitchenham, 2004].

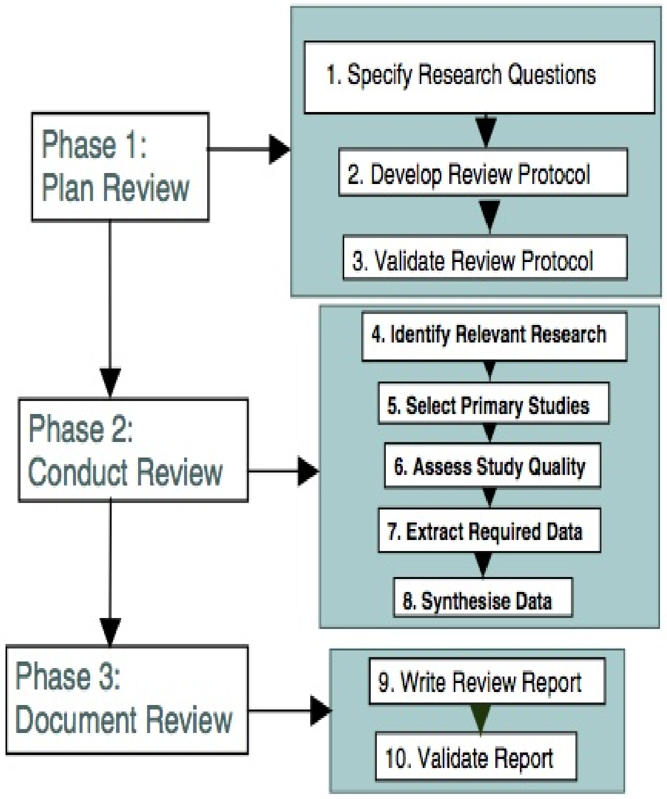


Figure 5.1: Systematic Literature Review Process

Research Questions

The research questions that the review is intended to answer have been discussed in section 5.1, which we will reiterate here: (i) What are the elicited lessons learnt from the RE literature?, (ii) How many lessons learnt are in the RE literature?, (iii) How are the lessons learnt expressed in the RE literature (explicitly or implicitly)?, (iv) Where (in which area of RE) are the lessons from the RE literature concentrated?, and (v) What is the quality of the lessons learnt elicited from the RE literature?

Review Protocol

The components of a protocol include all the elements of the review plus some additional planning information [Kitchenham, 2004]. The elements of the review, which will be discussed in detail in subsequent sections, include: research questions, search process (e.g., search terms, resources to be searched such as databases, journals and conference proceedings), study inclusion and exclusion criteria, quality assessment to assess the studies, data collection strategy, and data analysis/synthesis strategies.

Search Process

The search process was a manual search of RE-related conferences, journals, and workshop papers for years 2011 and 2012. A two-year time period was selected due to time and effort constraints; we intend to extend the years reviewed in our future work. Table 5.2 includes the names of all the selected journals, conferences and workshops. These journals, conferences and workshops were selected because they produce the major RE-related publications. The publications were accessed and downloaded via Western University's databases.

Table 5.2: Selected journals, conference and workshop proceedings.

Title	**Acronym**
Requirements Engineering Journal	RE
Empirical Software Engineering Journal	EMSE
IEEE Transactions on Software Engineering	TSE
Proceedings IEEE International Requirements Engineering Conference	IEEE RE
Proceedings International Conference on Software Engineering	ICSE
Proceedings International Working Conference on Requirements Engineering: Foundation for Software Quality	REFSQ
Proceedings IEEE International Workshop on Empirical Requirements Engineering	EmpiRE
Proceedings IEEE International Workshop on Model-Driven Requirements Engineering	MoDRE
Proceedings IEEE International Workshop on Requirements Engineering Education and Training	REET
Proceedings IEEE International Workshop on Requirements Engineering and Law	RELAW
Proceedings IEEE International Workshop on Requirements Patterns	RePa
Proceedings IEEE International Workshop on Requirements Engineering for Systems, Services, and Systems-of-Systems	RESS
Proceedings IEEE International Workshop on the Twin Peaks of Requirements and Architecture	TwinPeaks
Proceedings International Workshop on Managing Requirements Knowledge	MaRK
Proceedings International Workshop on Multimedia and Enjoyable Requirements Engineering	Mere
Proceedings Workshop Requirements@run.time	RE@Runtime
Proceedings International Workshop on Requirements Engineering for Social Computing	RESC
Proceedings Workshop on Requirements Engineering for E-Voting Systems	REVOTE
Proceedings Requirements Engineering Efficiency Workshop	REEW
Proceedings Workshop on Empirical Research in Requirements Engineering: Challenges and Solutions	EPICAL
Proceedings Requirements Engineering for Sustainable Systems	RE4SuSy
Proceedings Creativity in Requirements Engineering	CreaRE
Proceedings Requirements Prioritization for Customer Oriented Software Development	RePriCo

All publications were reviewed by the author for potentially relevant material.

Inclusion and Exclusion Criteria

Applying the inclusion/exclusion criteria in our study was done in two stages. In the first stage, all publications were reviewed for potential relevance. Potentially relevant here means that a publication may or may not contain a RE lesson(s), implicit or explicit. All articles from January 2011 to December 2012 that contained RE-related content were included and any articles on topics outside RE were excluded. In this case, all articles in each journal and conference and workshop proceedings were considered relevant except for ICSE proceedings and the Empirical Software Engineering journal where only publications with RE-related content were considered relevant.

In the second stage, articles were reviewed for lessons learnt. A lesson(s) learnt was elicited if it satisfied the following criteria:

- One of the literature lesson definitions discussed in Chapter 3 applied.
- There are actionable (either negative or positive), results from an empirical study (controlled experiment, experiment, exploratory experiment, quasi experiment, case study, confirmatory case study, retrospective case study analysis, pilot case study, explanatory case study, survey, workshop, questionnaire, pilot study, systematic review, qualitative study, end-user study, document analysis study), industrial experience, illustrative example, evaluating example, field assessment.

Quality Assessment

The difficulty of assessing the quality of the selected studies has been identified by Kitchenham [Kitchenham, 2004]. In our study, because we are eliciting lessons according to a set of criteria and filling in attributes, the quality assessment of the publications themselves has not been given much attention. We, however, have dedicated a research question for the quality assessment of the *lessons* instead of the selected studies as we will see in detail in Chapter 6. Lessons are assessed for their quality based on their attributes values.

Data Collection

The data extracted from each study that contained a lesson learnt were:

- Lesson ID: A unique ID for each lesson in the format: LLXXX_YY where XXX is a sequence number starting at 001 and YY is the year number (11 or 12).
- Citation: A key to reference the full reference from a local bibliography tool (BibDesk), which will be included as a citation in this document and added accordingly to the list of references.
- Lesson attribute values: All the information needed to fill in the following lesson attributes (see Chapters 3 and 4 for details): Journal, Conference, Workshop, Year, Lesson, Source, Rationale, Impact, Target object, Type, Expression, Application domain, Project size, RE practice, RE Phase, Software Process, Project date, Recording date, Organisation name, Repeatability

Data Analysis

The data was recorded as:

- Elicited lessons learnt from the reviewed literature (addressing RQ 1.1).
- The total number of papers, relevant papers, and elicited lessons from each source for each year. (addressing RQ 1.2).
- The number of explicit and implicit lessons (addressing RQ 1.3).
- The number of lessons for each source, target object, type, application domain, RE practice, RE phase, and software process (addressing RQ 1.4)
- The quality of each lesson (addressing RQ 1.5).

Review Protocol Validation

Because the review protocol is a critical element of the SLR, researchers must seek to get the protocol reviewed by experts. Graduate students typically seek feedback and criticisms from their supervisors. In our case, the review protocol was validated by the author's supervisor before beginning to conduct the SLR. All the elements of the protocol presented above have been reviewed and validated by the supervisor. Further validation will be conducted as part of our future work.

5.2.2 Survey

In section 5.1, we mentioned that part of our overall research goal is to understand the state of lessons learnt in RE. In section 5.2.1, we described in detail the SLR we conducted to address the first set of research questions (Section 5.1) regarding lessons learnt in the RE *literature*. In order to answer the next set of research questions, we have designed and begun conducting another empirical study (a survey) to investigate lessons learnt in practice. Although the study is still in its initial stages and the results are few, we have decided to document the study and initial results as it will be part of our ongoing work.

A survey can be used as research method if the research questions are in the following format: who, what, where, how many, how much? [Yin, 2003]. In addition, lessons learnt from diverse sources by diverse people would likely have less organisational and local-culture bias and so the collective set of lessons gathered would have the diversity of lessons and contexts built it which, as a collection, may be more applicable (or of Interest) in other contexts. A survey, therefore, may aid in gathering lessons from more diverse sources as opposed to interview from a few sources. For these reasons, we chose to conduct a survey as a first step towards addressing the research questions related to RE lessons in practice.

The inclusion and exclusion criteria [Kitchenham et al., 2002] of a survey specify the subjects who will and will not participate in the survey. Because the survey was accepted to the empirical track at the International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'13) in Essen, Germany, all RE participants were invited to participate. They had varying degrees of experience. Subjects with experience outside the RE domain were excluded. The following sections describe the instrument design, data collection and analysis, and participants in detail.

Instrument Design

- The survey was created and conducted via a web-based survey tool (Survey Gizmo).
- The survey consisted of both open-ended and close-ended questions.
- The first page of the survey included the survey introduction, which consisted of facts from a previous study we conducted [Noorwali and Madhavji, 2012] to motivate participants to take the survey. The introduction also included the purpose and a note that all responses are to remain anonymous unless the authors included their contact information. Estimated completion time also noted. Finally, the names and contact information of the survey administrators were included.
- The second page consisted of questions to gather the following participant demographic information:
  - Type of organisation that the participant has substantially worked in,
  - The key roles the participant played in his/her career, and
  - Number of years of work experience.
- The third page included the questions to gather the lesson along with the attributes discussed in Chapters 3 and 4.
- At the end of the third page, the participants were asked to provide their email address if they would like to receive the results of the survey. In addition, they were asked whether they would be willing to be contacted for clarification purposes. Finally, a question was added to ask the participants whether they would like to add another lesson. If yes, they were directed to another template; otherwise, the survey was complete.

- Ambiguous terminology was explained for each question to avoid confusion.
- The participants were provided with a URL to access the survey [Noorwali and Madhavji, 2013b] (Please see Appendix B for the complete survey).

Data Collection

The survey URL was publicised and distributed at the International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'13) to all conference participants. The average number of attendees at REFSQ are between 120-140. We received seven completed responses. Nine participants started the survey without completing it. The responses were stored in the database of the online survey tool we used.

Data Analysis

Due to the qualitative nature of the gathered data, a qualitative analysis (thematic coding technique) [Thomas and Harden, 2008] was conducted. Data was analysed to address the research questions. To address RQ 2.1, the gathered lessons were documented according to the RE Lesson representation discussed in Chapter 4. The number of lessons for each source, target object, type, application domain, RE practice, RE phase, and software process were counted to address RQ 2.2. A quality analysis was performed (Chapter 6) to address RQ 2.3. Finally, a qualitative comparison was performed to compare between the lesson from literature and practice (RQ 2.4).

Participants

Due to the small number of responses we received thus far, the participants are likely not representative of the larger RE community. Nevertheless, we will describe the participants' backgrounds, organisations and geographic distribution in this section because a survey's participants play an important role in the design of an empirical study.

Participant's Backgrounds

Despite the small number of respondents, they have held a number of different positions such as managers, requirements analyst/engineers, business analysts, developers, architects, researchers, consultants, and teachers. However, 100% of the participants held a researcher position. Since the question addressing the participant's key role played in his/her career allowed multiple answers, the total percentage exceeded 100%. Table 5.2 includes the details of the percentage distribution of the key roles played during the career of the participants.

Table 5.3: Percentage distribution of the key roles played during the career of the participants.

Role	**Percentage**
Manager	14.2%
Requirements Analysts/Engineer	71.4%
Business Analyst	28.6%
Developer	28.6%
Architect	28.6%
Researcher	100%
Consultant	85.7%
Teacher	42.3%
Other	14.3%

The participant's work experience ranged from 1 to 16+ years. The years of experience were divided into five sections, none being the the minimum number of years of work experience. However, none of the participants had zero year experience. Table 5.3 for details.

Table 5.4: Percentage distribution of years of work experience of the participants.

Number of Years	**Percentage**
None	0.0%
1-4 years	14.3%
5-10 years	14.36%
11-15	0.0%
16+	71.4%

Participants' Organisations

Because we are concerned with software process and project size of the lesson learnt itself rather than the respondents, we did not include a question in the survey that addresses these issues. However, this information was part of the lesson template the participants were asked to complete.

Participants' Geographic Distribution

In our survey, we have not asked participants to indicate their location as it is not relevant to our study. The context attributes which they were asked to fill provided us with all the lesson-related information we need.

5.3 Threats to Validity

This section discusses the threats to validity to our empirical studies and explains how they were addressed. Sections 5.3.1 and 5.3.2 discuss the types of validity for the systematic literature review and survey respectively. The study limitations, however, are discussed in chapter 9 (section 9.1).

5.3.1 Systematic Literature Review

This section discusses the threats to the internal, external, construct, and conclusion validities of the systematic literature review. Methods to deal with these threats are also discussed.

Internal Validity

Internal validity is concerned with the extent to which a causal relationship is warranted [Runeson and Host, 2009]. Therefore, it is only relevant in studies that try to establish a cause-effect relationship. Since our systematic literature review is not concerned with establishing causal relationships, this threat is not applicable in our study.

External Validity

External validity refers to the extent to which a set of results of a study are generalisable to a wider population of persons, settings, and time [Creswell, 2008]. In our case, external validity can be threatened if the lessons are not generalised enough for use in other contexts.

These threats can be mitigated to some degree by obtaining feedback from researchers and practitioners to validate the maps and elicited lessons and by identifying and analysing the context of each lesson. In addition, the documentation of all context attributes (e.g., project size, application domain, project date) of a lesson can help guide a user to decide whether it is applicable to a certain situation, project, or time.

Construct Validity

Construct validity refers to the degree to which the measured constructs are actually measured [Runeson and Host, 2009]. This is a threat to our study because the systematic literature review to elicit lessons was carried out by one researcher. Therefore, the researcher's interpretation of a 'lesson' may be different than that of another researcher. To deal with this threat, we have clearly identified and took into consideration the different definitions of a lesson, which were validated by a senior researcher and by four reviewers at REFSQ'13.

Conclusion Validity

Conclusion validity is concerned with whether the conclusions reached at the end of the study are reasonable or not [Johnson and Christensen, 2007]. This is a threat to our study as the conclusions we draw from the results is based on data from a two-year window only. Although some recurring patterns have been noticed throughout both years, two years may not be enough to generate concrete conclusions. We, therefore, present some emerging patterns (see Chapter 6), which we intend to validate or disprove by eliciting more lessons from literature coming from more years.

5.3.2 Survey

As in the previous section, this section discusses the threats to the internal, external, construct, and conclusion validities of the survey. It is important to note again that this study is still in its initial stages. Therefore, methods to deal with the threats to the different validities have not yet been pursued. However, we discuss how we intend to deal with these threats in our ongoing work.

Internal Validity

Internal validity is concerned with the extent to which a causal relationship is warranted [Runeson and Host, 2009]. Therefore, it is only relevant in studies that try to establish a cause-effect relationship. Since our survey aims to gather data from practitioners and is not concerned with establishing causal relationships, this threat is not applicable in this study.

External Validity

The same threats to external validity that applied to the systematic literature review's results apply to the survey's results. The gathered lessons may not be generalisable to the general population. We intend to mitigate these threats by obtaining expert feedback on the lessons. The lesson's context attributes that were included in the survey are also a method to deal with this threat as the context is described in detail. In addition to the general external validity, it is necessary in the case of the survey to discuss three additional types of external validity threats that may threaten the study: population validity, ecological validity and temporal validity.

Population validity refers to the extent to which the sample is representative of the population as a whole. This threat exists in our study since the survey was conducted at REFSQ'13 only. This limits the variety of subjects who have participated in the survey because REFSQ may attract only a certain portion of the RE community. In addition, the number of responses we have received thus far is very small. To deal with this threat, we plan to distribute the survey to a larger population as part of our ongoing work (e.g. RE conference, via email lists, etc.).

Ecological validity is concerned with the degree to which the study setting represents the real-world situation (i.e. laboratory setting versus real-world setting). Since a survey is not conducted in a specific case, as the case with an experiment or case study, this threat is not relevant to the survey.

Temporal validity is the ability to generalise the results of a study over time. Whether the elicited lessons are generalisable across time or not is not known. This may be measured when the concept of lessons learnt has been promoted and has significantly matured in the RE community.

Construct Validity

Construct validity is critical to the overall validity of the survey. Two types of construct validity are relevant in this case: content validity [Salkind, 2007] and face validity [Bornstein et al., 2004].

Content validity refers to the degree of which the research instrument accurately represents the specific intended domain of content. In our case, we are concerned with the accurate representation of a lesson learnt. To deal with this threat, we have included the definitions of a lesson from reputable dictionaries.

Face validity measures how representative a research project is 'at face value,' and whether it appears to be a good project. In our case, the threat to face validity was mitigated by submitting the survey to the Empirical Track at REFSQ'13, which was reviewed by two reviewers before being accepted for distribution.

Conclusion Validity

The reasonableness of the conclusions drawn at the end of the study is threatened due to the small number of responses we have received. We plan to mitigate this threat by increasing the number of responses to a statistically significant number and validating the conclusions via expert feedback and peer-reviews. While there is uncertainty as to the results more responses will yield, this is something we intend to evaluate.

Chapter 6: Results of the Empirical Study: Elicited Lessons

In this chapter, we present the results of the empirical study we discussed in Chapter 5 (systematic literature review and survey). Section 6.1 includes ten elicited lessons from the literature of years 2011 and 2012 (five for each year) from the systematic literature review we conducted. Although we elicited a total of 209 lessons from the literature, we list only 20 of them here, due to space constraints (for the complete list of lessons, please see Appendix C). Section 6.2 presents the lessons we have thus received from the survey we conducted at REFSQ'13. Finally, Section 6.3 summarises and discusses the results with regard to the research questions (Section 5.1).

It is important to note that, although the elicited lessons followed the definitions and attributes discussed in chapters 3 and 4, and were validated by a senior researcher, they have yet to be validated for usefulness, applicability, and completeness. Therefore, the results of the study should not be generalised before conducting further validation processes in subsequent studies. In addition, due to researcher bias, there may have been some overlooked lessons during the elicitation process.

6.1 RE Lessons from Literature

Table 6.1 shows the results of the systematic literature review. The first column identifies the year; the second column includes the names of the sources that were reviewed; the third column includes the total number of papers reviewed and considered relevant from each source; the fourth column includes the total number of papers selected to be reviewed for lessons learnt; and finally, the fifth column includes the total number of elicited lessons.

Table 6.1: Sources searched for years 2011 and 2012

Year	Source	Total # of Reviewed Papers	Total # of Selected Papers	Total # of Elicited Lessons
2011	RE	19	19	11
	EMSE	25	2	2
	TSE	48	3	2
	IEEE RE	35	35	28
	Workshops at IEEE RE	101	101	32
	REFSQ	20	20	18
	Workshops at REFSQ	13	13	5
	ICSE	128	6	3
	Total	**389**	**199**	**101**
2012	RE	16	16	12
	EMSE	24	3	7
	TSE	82	1	0
	IEEE RE	35	35	35
	Workshops at IEEE RE	54	54	36
	REFSQ	27	27	10
	Workshops at REFSQ	21	21	7
	ICSE	87	5	1
	Total	**346**	**162**	**108**

Recall from Section 5.1, the first research question for the systematic literature review was:

RQ 1.1 *What are the elicited lessons learnt from the RE literature?*

To answer the question, we present, for illustration purposes, in sections 6.1.1 and 6.1.2 a randomly selected sample of elicited lessons (six lessons) from the literature of years 2011 (three lessons) and 2012 (three lessons), respectively. The complete list of lessons are in Appendix C.

For year 2011, the sample lessons are from the RE journal, IEEE RE conference, and RES4 workshop. Likewise, the lessons for year 2012 are from IEEE RE conference, RELAW workshop, and RE journal. The lessons deal with RE phases such as analysis, elicitation, validation and others. They come from application domains such as aviation security, banking, and others.

6.1.1 Lessons from Year 2011

Lesson ID: LL040_11 [Asnar et al., 2011]

Journal: RE

Conference: N/A

Workshop: N/A

Year: 2011

Lesson: Use the Goal-Risk framework for modeling and reasoning about risk during requirements analysis extending the Tropos goal modeling framework.

Source: Case study

Rationale: "Risk analysis is traditionally considered a critical activity for the whole software systems lifecycle. Risks are identified by considering technical aspects (e.g., failures of the system, unavailability of services, etc.) and handled by suitable countermeasures through a refined design. This, however, introduces the problem of reconsidering system requirements." [Asnar et al., 2011]

Impact: "Positive experiences in communicating GR models to analysts and domain experts. This is an important strength for any requirements analysis technique because it empowers domain experts to under- stand and critique proposed models. Moreover, the learning process for experts to understand and use a GR model takes relatively short period (approximately 23 months). The GR framework supports risk analysis during the very early phases of software development. Consequently, it reduces the risk of requirements revision, and consequently the cost of development. In comparison with KAOS, this framework allows analysts to perform qualitative and quantitative assessment though KAOS provides richer formal semantics using Linear Temporal Logic. Moreover, in comparison with DDP and CORAS the GR framework is more expressive in capturing stakeholders intentions. At last, the GR framework is the only framework that deals with risk and opportunity, since some risks appear because the stakeholders decide to pursue an opportunity. With this feature, one can perform trade-off analysis to decide whether one opportunity is worth to pursue or not." [Asnar et al., 2011]

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Banking

Project size: N/A

RE Practice: Using a framework

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

Lesson ID: LL067_11 [Waldmann, 2011]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2011

Lesson: When applying insights from agile development to requirements engineering, the key concepts from the agile world such as user stories or product owners must be mapped intelligently to appropriate concepts in RE, not copied.

Source: Case study

Rationale: "While Requirements Engineering textbooks state that a requirements specification must be complete, in real-life projects we are always starting too late, with too few resources, so we cant do everything. The software development community has solved a similar problem (not having enough resources to implement everything that was asked for) by introducing agile development methods, which offer ways of segmenting the overall project, and choosing which parts to allocate resources to." [Waldmann, 2011]

Impact: "Our case study confirmed that a flexible requirements engineer ing process inspired by agile development methods can de- liver results that provide business value, even with severe resource constraints.Our case study also demonstrated that agile requirements engineering activities can indeed feed into development project that follows a classical V-model approach, by making a clear distinction between incremental delivery of requirements vs. non-incremental delivery of implementation. The implementation part also included hardware development subprojects, and our case study demonstrates the feasibility of agile requirements engineering activities preceding development activities which, for technical reasons, cannot deliver in multiple releases." [Waldmann, 2011]

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Hearing solutions, including hearing instruments, hearing protection and communication systems, and implantable hearing systems

Project size: "The project duration is more than 3 years from concept to the launch of 1st products. It involves more than 100 R&D engineers, including the equivalent of 2-4 full-time requirements engineers. The system requirements specification currently consists of around 1200 system requirements items, from which component-specific technical requirements are derived." [Waldmann, 2011]

RE Practice: N/A

RE Phase: Elicitation, analysis, specification, validation

Software Process: Agile

Project Date: N/A

Recording Date: N/A

Organisation Name: Sonova Group

Repeatability: First time industrial case study results

<u>Lesson ID:</u> LL099_11 [Bahrs and Nguyen, 2011]

Journal: N/A

Conference: N/A

Workshop:RES4

Year: 2011

Lesson: Use the Smarter architecture & engineering (SmarterAE) approach for requirements management. It is a way of rethinking the requirements, architecture and systems engineering life cycles on SOA, Modernization and Transformation projects.

Source: Industrial experience

Rationale: "Many project struggle with the so called requirements hand off. This is the situation where requirements are documented in text and handed to a team for implementation. In the majority of projects, the developers struggle with semantics, clarity and on time delivery. This occurs on projects of various time durations and complexity from small embedded systems to large geographically dispersed systems of systems." [Bahrs and Nguyen, 2011]

Impact: "Reduction in costs by 33%. Reduction in time by 40%.Successful partitioning of teams across time zones and organizations. Successful delivery of Claims Processing and Border Management applications in 30 days, from requirement to execution. Predicted savings of 15, 25 and 40% over three years. 60% of projects producing the correct assets. Business process assets more difficult to produced than all other business architecture assets. Asset types and standards continuing to change." [Bahrs and Nguyen, 2011]

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Business Process Management (BPM) project. SOA project. Legacy Modernization project. Transaction Processing project. Business Agility project. Model-driven Architecture (MDA) project. Model-driven Architecture (MDA) project. Real-time mission critical systems project. Product development project

Project size: "SmarterAE is typically valuable in large complex enterprise-wide projects." [Bahrs and Nguyen, 2011]

RE Practice: N/A

RE Phase: Elicitation, managing

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: IBM

Repeatability: Numerous projects and customers are using the SmarterAE approach over the last two years.

6.1.2 Lessons from Year 2012

Lesson ID: LL007_12 [Yi et al., 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Use Yi's et al. approach to mine cross-tree binary constraints in the construction of feature models.

Source: Evaluating experiment

Rationale: "Identifying features and then building a feature tree takes a lot of effort, and many semi-automated approaches have been proposed to help the situation. However, finding cross-tree constraints is often more challenging which still lacks the help of automation." [Yi et al., 2012]

Impact: "The approach successfully finds binary constraints at a high recall (near 100% in most cases). The precision is unstable and dependent on the test feature models. In most cases the requires constraints are better mined than the excludes constraints; a possible reason is that the rationale behind excludes is often beyond feature descriptions. Continuous feedback from human analysts benefits the mining process, especially for mining excludes constraints. Therefore in practice, our classifier should be used in an interactive way, that is, human analysts check only a few constraint candidates after each turn of mining, and then the classifier repeats the train-optimize-test process again." [Yi et al., 2012]

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Healthcare in the US

Project size: The weather station feature model: 22 features, 196 feature pairs, 6 requires, and 5 excludes. Graph Product Line feature model: 15 features, 91 feature pairs, 8 requires, and 5 excludes

RE Practice: N/A

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: S.P.L.O.T. repository

Repeatability: First time experimental results

Lesson ID: LL053_12 [Braun et al., 2012]

Journal: N/A

Conference: N/A

Workshop: RELAW

Year: 2012

Lesson: Natural Language before Performance Model does not impact the quality of regulations or only very late in the regulatory process.

Source: Industrial experience

Rationale: "The status quo is that performance modeling is not routinely included in the regulatory process, which may lead to lack of clarity, inconsistencies, and difficulties measuring and hence assessing compliance." [Braun et al., 2012]

Impact: "There is no or little improvement in terms of inconsistencies in the regulations, the understandability of the regulations, and the measurability of desired regulation outcomes. It is likely that some measures will be difficult to quantify, leading to problems for effectively enforcing the regulations." [Braun et al., 2012]

Target Object: Technique/method

Type: Negative

Expression: Implicit

Application Domain: Aviation Security

Project size: "Transport Canada is engaged in a multi-year modernization process to review and renew its Aviation Security regulations." [Braun et al., 2012]

RE Practice: Modeling

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Transport Canada

Repeatability: First time industrial experience

Lesson ID: LL108_12 [Svensson et al., 2012]

Journal: RE

Conference: N/A

Workshop: N/A

Year: 2012

Lesson: Use the QUality PERformance (QUPER) model with estimations of benefit and cost of quality targets in relation to market expectations as a basis for the architecting of quality requirements.

Source: Case study

Rationale: "Quality requirements play a critical role in driving architectural design and are an important issue in software development. Therefore, quality requirements need to be considered, specified, and quantified early during system analysis and not later in the development phase in an ad-hoc fashion." [Svensson et al., 2012]

Impact: "In general, QUPER does not only help in creating a more aligned view of quality requirements, but also to use one method to measure all quality requirements. All subjects confirmed that QUPER would support and coordinate the early decision-making process, e.g., release planning. The QUPER model is aimed to facilitate the elicitation, specification, quantification, and prioritization of QR."

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: "Electronic payment-processing: payment terminals, transaction processing, and development of saving- and customer-card systems."

Project size: "Company employs more than 250 employees, has more than 120,000 customers and business partners." [Svensson et al., 2012]

RE Practice: N/A

RE Phase: Elicitation, Specification, Prioritization

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: Confirm previous results from the mobile handset

6.2 RE Lessons from Practice

Recall from Section 5.1, the first research question for the survey was:

RQ 2.1 *What are the elicited lessons learnt from industry/practice?*

Results:

Lesson ID: LL109_12

Type of organization you have worked in substantially (please choose one or more): Academic institution

Key roles played in your career (please choose one or more): Requirements analyst/Engineer, Business analyst, Developer, Architect, Researcher, Consultant

Number of years of work experience: 16+

Year: 2012

Lesson: The most important thing I have learned about RE is to allow as much time as is needed

Source: Industrial experience

Rationale: Because it's pervasive

Impact: Spend enough time and the quality of the product goes up

Target Object: Process, policy, Other: project management

Type: Negative

Application Domain: Banking

Project size: 5 person-year

RE Practice: All

RE Phase: Elicitation, Analysis, Prioritization, Negotiation, Specification, Modeling, Validation

Software Process: Waterfall

Repeatability: YES.. anytime not enough time is given for RE, the resulting requirements specs are deficient, leading to a poor product

Lesson ID: LL110_12

Type of organization you have worked in substantially (please choose one or more): Academic institution

Key roles played in your career (please choose one or more): Requirements analyst/Engineer, Business analyst, Researcher, Consultant

Number of years of work experience: 5-10 years

Year: 2012

Lesson: 1. Requirements are in most cases not just ”The system shall” statements. Good requirements rather describe reference objects that are to be supported or even implemented by a system (e.g., use cases, users, business processes, processed data, screens, etc.). In particular, thinking in such reference objects allow people to know at least partially a) what to ask, b) in which order, c) by whom and d) when finished. We learned that using reference objects as a mental model is much better to organize RE processes than trying to distinguish between problem and solution space, between what and how or between business (usage) requirements and technical requirements as there are too much overlaps in practice. 2. Consider different perspectives. Requirements Engineering is no end in itself. Requirements Engineering must satisfy the consumers of the requirements (e.g., the developers) and the producers of the requirements (e.g., the customers and users). Requirements are stated because people feel problems to be solved. Thus, requirements should not limit themselves on software properties and must range from a very high and software-independent level (e.g., business analysis) down to software implementation. 3. The requirements on requirements must be better considered. Many software requirements are useless because the requirements of requirements consumers on requirements, e.g., regarding content, structure, notation, point of delivery etc. are not clear.

An RE Process must elaborate the requirements of the customers and users, of course, but the process itself must fulfill the requirements of the developers! 4. Consider power and politics. Without officially involving important decision makers at the very beginning, much RE effort may be wasted, because these decision makers can skip every requirement at a very later point in time, otherwise.

Source: Industrial experience

Rationale: Because many organizations (of different size and different domain) are challenged by them

Impact: When following these lessons we experienced better requirements, higher employee satisfaction in the project team and higher efficiency

Target Object: Technique/method, process, policy

Type: Negative

Application Domain: different ones (see above)

Project size: different ones (normally medium-size projects)

RE Practice: N/A

RE Phase: Elicitation, Analysis, Prioritization, Specification

Software Process: Waterfall, Iterative, Spiral, Code-and-Fix

Repeatability: Lesson 1 experienced in almost every RE teaching and transfer project independent of the domain. People in practice have huge problems to know what to describe, by whom in which degree of detail. By thinking in reference objects, these problems could always be eliminated. Lesson 2 experienced especially in projects where other parties (e.g. business consultants or implementation companies) were involved. Without considering RE as a holistic design process, redundancies, overlaps and conflicts occur in such settings, because every party is doing its own RE in an isolated scope without having clear interfaces. Lesson 3 experienced in several consulting projects in which we helped organization to improve their RE processes. Understanding customer needs is sometimes less problematic than producing specifications with which developers can work without additional explanations. So many misconceptions about RE are often caused by the fact that requirements specifications are not experienced as a help for practitioners. Lessons 4 experienced in almost every project with the German government. Many people are involved to discuss requirements and solutions and at the end millions euro of tax are spent but some ”leader” decides not to proceed.

Lesson ID: LL111_12

Type of organization you have worked in substantially (please choose one or more): Academic institution

Key roles played in your career (please choose one or more): Researcher, Teacher, Other: I have participated in some projects in industry as RE consultant and I have directed a lot of student projects where RE was present

Number of years of work experience: 16+ years

Year: 2012

Lesson: - It is important the organization and writing of the requirements in order there are not misunderstandings among the customer and developer. - It is important that they are enough complete and detailed, in order the developer can not skip to implement some aspect of the system. - In systems with a lot of different user roles and IT previous knowledge it is impossible to satisfy the needs of all future users. It is the customer organization who has to finally decide when there are contradictory requirements, and do some activities to teach the users of the new system to get used to it and to the changes in processes that it implies.

Source: Other

Rationale: N/A

Impact: N/A

Target Object: Technique/method, process, policy

Type: Negative

Application Domain: Curriculum Management, other diverse domains...

Project size: 3000 users, other diverse sizes.

RE Practice: N/A

RE Phase: Elicitation, Specification

Software Process: Agile, Waterfall, Iterative, Spiral, Code-and-Fix, Rapid prototyping

Repeatability: I think they are both repeatable in all contexts.

Lesson ID: LL112_12

Type of organization you have worked in substantially (please choose one or more): Academic institution

Key roles played in your career (please choose one or more): Requirements Analyst/Engineer, Researcher, Consultant

Number of years of work experience: 1-4 years

Year: 2012

Lesson: Developers are not aware (or they do not want to be aware) of big requirements documents and specifications. They usually do not read them, or just overlook at them, without taking care of the details.

Source: Industrial experience

Rationale: It is important in distributed systems, in the way you communicate with your team, to take into account what documentation you produce while specifying the system.

Impact: The problem of a not good understanding of requirements specification document is that developers do something similar to what it is specified, but not what it is specified, affecting process efficiency and project costs.

Target Object: Technique/method

Type: Negative

Application Domain: Social websites

Project size: 5 person-month (during 1 year)

RE Practice: N/A

RE Phase: Elicitation, Analysis, Specification, Documentation

Software Process: Agile, Spiral, Code-and-Fix,

Repeatability: Yes, it is. I think it actually applies in any project, but specially in those ones that work in a distributed way (project manager and business analyst with customers in one side, development and testing team in another side, and both sides cannot meet in the same place).

Lesson ID: LL113_12

Type of organization you have worked in substantially (please choose one or more): Academic institution

Key roles played in your career (please choose one or more): Researcher, Consultant, Teacher

Number of years of work experience: 16+ years

Year: 2012

Lesson: Commitment of the managers of the company is mandatory for the success of requirement elicitation.

Source: Industrial experience

Rationale: Very important

Impact: Failure of the project

Target Object: Process, other: communication

Type: Negative

Application Domain: no-profit, elearning

Project size: 5 person-month (during 1 year)

RE Practice: N/A

RE Phase: Elicitation, Negotiation

Software Process: Other: all

Repeatability: Yes, Any context

Lesson ID: LL114_12

Type of organization you have worked in substantially (please choose one or more): Industry, Academic institution

Key roles played in your career (please choose one or more): Manager, Requirements Analyst/Engineer, Researcher, Consultant

Number of years of work experience: 16+ years

Year: 2012

Lesson: Early prototyping is essential. A screen shot sais more than 1000 words and is much less ambiguous. A lot of misunderstandings can be avoided by early prototyping. Prototyping can mean to use powerpoint prototypes of the user interface, or to implement / customize a solution. With 20% of the effort, you can visualize 80% of the functionality.

Source: Industrial experience

Rationale: The discussion of the prototype always helped to discover previous midunderstandings which might have caused unnecessary cost for the implementing team. And the prototype gave the customers a better feeling: the feeling to understand, the feeling to be understood, the feeling to be able to influence the way the software will look.

Impact: Failure of the project

Target Object: Technique/method, Tool

Type: Positive

Application Domain: Development of CRM systems and CASE tools

Project size: several projects, of 2 - 10 persons

RE Practice: N/A

RE Phase: Elicitation, Analysis, Prioritisation, Specification, Documentation

Software Process: Waterfall, Iterative

Repeatability: Product quality is higher in terms of user satisfaction. The development process will be more efficient and project cost lower. There is only on problem with software prototypes against graphical prototypes: As you can implement a first running prototype within few days, the customer does not understand why the rest of the project (setting up databases, developing technical interfaces, reports, etc, and quality assurance) will take several months. For customers/ users the user interface is what they pay for. They told me that the need for database and technical interfaces is not their need but ours (the development teams). So, they even discussed whether they must pay for the database or we must. I could convince them that in fact THEY need the database. The prototype clearly showed what happened without a database behind: Data simply get lost.

Lesson ID: LL115_12

Type of organization you have worked in substantially (please choose one or more): Industry, Academic institution

Key roles played in your career (please choose one or more): Requirements Analyst/Engineer, Developer, Architect, Researcher, Consultant, Teacher

Number of years of work experience: 16+ years

Year: 2012

Lesson: Even if you do not do requirements engineering, merely by writing code, you make requirement decisions. So there is no avoiding determining requirements.

Source: Industrial experience

Rationale: It shows how ridiculous skipping or abbreviating RE is.

Impact: If you don't do RE, then requirement decisions are made by the programmers, each acting on his or her own.

Target Object: Process

Type: Negative

Application Domain: Development of CRM systems and CASE tools

Project size: 20 person years

RE Practice: The whole RE process

RE Phase: Elicitation, Analysis, Prioritisation, Negotiation, Modeling, Validation

Software Process: Agile, Waterfall, Iterative, Spiral, Code-and-Fix, Rapid prototyping

Repeatability: It happens in EVERY development

6.3 Summary and Discussion

The previous two sections discussed the results of the empirical study, which answered two of the research questions (RQ 1.1 and RQ 2.1) from section 5.1. In this section, we present a summary of the results and discuss them with regard to the remaining research questions (RQ 1.2, RQ.1.3, RQ.1.4, RQ 1.5, RQ 2.2, RQ 2.3, and RQ 2.4).

RQ 1.2 *How many lessons learnt are in the RE literature?*
As we've seen in Section 6.1, we elicited a total of ***209*** lessons from the RE literature in years 2011 and 2012; 101 lessons from year 2011 and 108 lessons from year 2012. For year 2011, out of 162 selected papers (see table 6.1), 108 lessons were elicited, meaning that 67% of the papers contained lessons. For year 2012, 101 lessons were elicited from 199 selected papers, meaning that 51% of the papers contained lessons. The difference in numbers may be due to a number of factors. First of all, some papers contained several lessons while others contained none. Thus, there may have been papers in 2012 that were rich in lessons as opposed to the papers in 2011, which skewed the numbers a bit. Another factor may be due to the fact that the literature was reviewed by one researcher. While several measures have been taken to mitigate this risk (see Section 5.3), there is a likelihood that some lessons have been overlooked in some papers.

RQ 1.3 *How are the lessons learnt expressed in the RE literature (explicitly or implicitly)?*
Table 6.2 shows the numbers of explicit and implicit lessons in the literature. Out of the 101 elicited lessons of year 2011, only 25 were explicitly mentioned as lessons learnt and 76 were implicit in the literature (i.e. there was a need to analyze the surrounding text and context to elicit the lesson). That is, 25% are explicit lessons and 75% are implicit. For 2012, 22 out 108 lessons are explicit (20%) and 86 are implicit (80%). In general, only 47 (22%) out of the total of 209 lessons, are explicit and 162 (78%) are implicit. We can see that the numbers for both years do not show a significant difference. Although it is too early to draw concrete

conclusions, the current results show that our initial observation that the lessons are scattered, mainly implicitly in the literature, holds true. This may indicate that the RE community is not explicitly documenting the lessons they are learning.

Table 6.2: Numbers for the attribute 'Expression' for the lessons of years 2011 and 2012

Expression	**2011**	**Percentage**	**2012**	**Percentage**
Explicit	25	24.75%	22	20.37%
Implicit	76	75.25%	86	79.63%

RQ 1.4 *Where (in which area of RE) are the lessons from the RE literature concentrated?*
Table 6.3 shows the number of lessons for each RE phase for years 2011 and 2012. We can see that the highest number of lessons in year 2011 are for the analysis (34%), elicitation (30%) and specification (30%) phases. In 2012, while the same phases held the three top spots, the order was a bit different. Elicitation had the highest number of lessons (33%), then analysis (31%) and specification (23%). The difference between the number of lessons targeting documentation between 2011 and 2012 is large (1% to 19%). This can be partly explained by the fact that one paper contained seven lessons related to documentation and another contained five, which skewed the numbers significantly. The lessons related to the remaining RE phases (prioritisation, negotiation, validation, verification, and managing) are similarly small for both years. We can clearly see that there is a dearth of lessons in these areas of RE and that the lessons are mainly targeted at the elicitation, analysis, and specification phases. It is important to note, however, that the percentages do not add up to 100% because some lessons are concerned with more than one phase.

Table 6.3: Numbers for the attribute ‘RE Phase’ for the lesson of years 2011 and 2012

RE Phase	**2011**	**Percentage**	**2012**	**Percentage**
Unidentified	12	11.88%	15	13.89%
Elicitation	30	29.70%	36	33.33%
Analysis	34	33.66%	34	31.48%
Prioritisation	5	4.95%	5	4.63%
Negotiation	4	3.96%	6	5.56%
Specification	28	27.72%	25	23.15%
Documentation	1	0.99%	20	18.52%
Validation	7	6.93%	4	3.70%
Verification	1	0.99%	1	0.93%
Managing	3	2.97%	1	0.93%

As for the target object of the lessons, most of the lessons’ target object is ‘technique/method’; 77% and 75% for years 2011 and 2012, respectively (see Table 6.4). Lessons on ‘tools’ take the second place; 20% and 22% for years 2011 and 2012 respectively. The lessons on the remaining target objects are very few. This suggests that the RE community concentrates on techniques and tools more than on any other target objects. It would be interesting to explore the effects of this on RE practices in projects and organisations.

Table 6.4: Numbers for the attribute ‘Target Object’ for the lessons of years 2011 and 2012

Target Object	**2011**	**Percentage**	**2012**	**Percentage**
Unidentified	1	0.99%	0	0.0%
Technique/method	78	77.23%	81	75.00%
Tool	20	19.80%	24	22.22%
Policy	2	1.98%	2	1.85%
People	2	1.98%	0	0.0%
Language	5	4.95%	0	0.0%
Artifact: requirements	0	0.0%	5	4.63%
RE analyst	0	0.0%	2	1.85%

Table 6.5 shows the numbers and percentages of lessons related to each RE practice in the table. Around half of the lessons were not clearly related to a specific RE practice (52% and 50% for years 2011 and 2012 respectively). Modeling, however, received a lot of attention in year 2012 as opposed to year 2011(16% versus 30%). The remaining practices had only a small number (5 maximum) of related lessons.

Table 6.5: Numbers for the attribute ‘RE Practice’ for years 2011 and 2012

RE Practice	**2011**	**Percentage**	**2012**	**Percentage**
Unidentified	52	51.49%	54	50%
Tracing	4	3.96%	4	3.70%
Modeling	3	2.97%	17	15.74%
Prototyping	2	1.98%	2	1.85%
Reuse	2	1.98%	3	2.78%
Using a prioritisation framework	1	0.99%	3	2.78%
Applying ‘Pictionades’	0	0.0%	5	4.63%
Communication	3	2.97%	5	4.63%
Interviews	1	0.99%	1	0.93%
Brainstorming	1	0.99%	1	0.93%
Annotation	1	0.99%	2	1.85%
Use of use cases	1	0.99%	1	0.93%
Use of specification pattern system (SPS)	0	0.0%	1	0.93%
Use of text-based synchronous communication	0	0.0%	5	4.63%
Use of automatic checks	1	0.99%	1	0.93%
Using patterns	1	0.99%	5	4.63%
Use of PRISM model checker	1	0.99%	1	0.93%
Use of task descriptions	1	0.99%	2	1.85%
Using feature trees	0	0.0%	1	0.93%
Using War stories approach	1	0.99%	0	0.0%
Using a social network	1	0.99%	0	0.0%
Using a framework	3	2.97%	0	0.0%
Using model-driven development	1	0.99%	0	0.0%
Using modeling languages	1	0.99%	0	0.0%
Automatic tracing	1	0.99%	0	0.0%
Using a requirements model	1	0.99%	0	0.0%
Using a specification language	1	0.99%	0	0.0%
Visualising traceability information	1	0.99%	0	0.0%
Using observation techniques	1	0.99%	0	0.0%
Using quantitative models	1	0.99%	0	0.0%
Using software quality models	1	0.99%	0	0.0%
Assisted tracing	2	1.98%	0	0.0%
Using templates	2	1.98%	0	0.0%
Using storytelling	1	0.99%	0	0.0%
Using visual narratives	1	0.99%	0	0.0%
Using CPR analysis	1	0.99%	0	0.0%
Using scenarios	1	0.99%	0	0.0%
Using storyboards	1	0.99%	0	0.0%
Protocol analysis	1	0.99%	0	0.0%

Table 6.6 shows the number of lessons for the different application domains from which they emerged. Most of the lessons emerged from the 'university' domain (12% and 8% for years 2011 and 2012 respectively) as most of the experiments take place in an academic setting. We must mention here that the lessons emerging from the 'hearing solutions' application domain come from one paper.

Because the values of this attribute are highly variable, we resorted to listing the numbers of the application domains that have 3 or more lessons only. For the complete list of application domains, please see Chapter 4.

Table 6.6: Numbers for the attribute 'Application Domain' for the lessons of years 2011 and 2012

Application Domain	**2011**	**Percentage**	**2012**	**Percentage**
Unidentified	9	8.91%	7	6.48%
Automative	3	2.97%	3	2.77%
Healthcare (HIPAA)	5	4.95%	4	3.70%
University	12	11.88%	9	8.33%
Aviation	1	0.99%	6	5.55%
Solar	0	0.0%	7	6.48%
Drives	0	0.0%	6	5.55%
Hearing solutions	12	11.88%	0	0.0%
Event management	4	3.96%	0	0.0%
Software platform providing core assets to developers of other organizations	9	8.91%	0	0.0%

For the attribute 'software process', about 89% of the lessons for both years 2011 and 2012 did not identify a related software process. However, the Agile SW process has the most lessons in 2011. In 2012, Agile and Iterative processes had an equal number of lessons.

Table 6.7: Numbers for the attribute 'Software Process' for the lessons of years 2011 and 2012

Software Process	**2011**	**Percentage**	**2012**	**Percentage**
Unidentified	89	88.11%	96	88.89%
Agile	10	9.90%	7	6.48%
Iterative	2	1.98%	7	6.48%
Waterfall	0	0.0%	4	3.70%

Table 6.8 shows that the lessons come mainly from case studies (52% and 43%), experiments (17% and 8%), and industrial experience (11% and 26%) for both years 2011 and 2012, respectively. The lessons from industrial experience in 2012 are considerably more than their counterpart in 2011. While the reason behind this is not clear, one plausible explanation may be that the conferences in 2012 accepted more industrial papers than in 2011. We can see that the numbers are in favour of the lessons from case studies. It would be interesting to explore the reasons and implications of this observation. In addition, this may say something about the reliability of the lessons because a lesson from an experiment for example, may be more reliable than one from a case study as experiments are conducted under certain conditions that can be replicated.

Table 6.8: Numbers for the attribute 'source' for the lessons of years 2011 and 2012

Source	**2011**	**Percentage**	**2012**	**Percentage**
Unidentified	0	0.0%	0	0.0%
Qualitative study	1	0.99%	0	0.0%
Systematic review	3	2.97%	0	0.0%
End user study	1	0.99%	0	0.0%
Case study	52	51.49%	46	42.59%
Exploratory experiment	1	0.99%	0	0.0%
Controlled experiment	6	5.94%	10	9.26%
Exploratory study	4	3.96%	2	1.85%
Confirmatory case study	0	0.0%	1	0.93%
Experiment	17	16.83%	9	8.33%
Pilot case study	1	0.99%	2	1.85%
Quasi-experiment	2	1.98%	0	0.0%
Explanatory case study	1	0.99%	0	0.0%
Retrospective case study analysis	0	0.0%	1	0.93%
Survey	1	0.99%	3	2.78%
Questionnaire	1	0.99%	5	4.63%
Field assessment	1	0.99%	0	0.0%
Evaluating example	1	0.99%	0	0.0%
Illustrative example	0	0.0%	2	1.85%
Simulation	0	0.0%	1	0.93%
Workshop	0	0.0%	1	0.93%
Document analysis study	1	0.99%	0	0.0%
Industrial experience	11	10.89%	28	25.93%

Table 6.9 shows that most lessons are 'positive'. We came across 3 cases where the lesson was both negative and positive because the experience from which it was derived had both successful and failing aspects. This did not occur to us in the beginning of the study, but after eliciting the lessons, we realised that it was difficult to identify whether a lesson was 'positive' or 'negative' in a clean-cut manner. A possible explanation for the large number of positive lessons is that researchers tend to publish positive results rather than negative ones.

Table 6.9: Numbers for the attribute 'Type' for the lessons of years 2011 and 2012

Type	**2011**	**Percentage**	**2012**	**Percentage**
Unidentified	0	0.0%	0	0.0%
Positive	83	82.17%	64	59.26%
Negative	13	12.87%	29	26.85%
Both	2	1.98%	1	0.92%
Neutral	3	2.97%	14	12.96%

Table 6.10 shows how many of the following attributes were identified and unidentified in the elicited lessons from years 2011 and 2012: project size, project date, recording date, organization name, rationale and impact. We do not list the values of the attributes as they are not of significant importance to our research goal; they are meant to aid users during the reuse of a lesson. It can be noticed that almost none of the lessons indicated the project and recording dates. This was expected as these dates may be relevant only when lessons learnt are utilised in practice. In our case, the publication year is a good indication of the date of the lesson.

RQ 1.5 *What is the quality of the lessons learnt elicited from the RE literature?*

To assess the quality of a specific lesson, we compare the completeness of the values of its attributes to the complete set of attributes. Thus, fewer the 'unidentified' (i.e. N/A) values, higher the quality of a lesson. Although there are many unidentified values in the lessons we elicited, there are lessons with more identified values than others. We demonstrate our approach here with a few examples of high and low quality lessons.

Table 6.11 contains an example of a low-quality lesson. Out of the 19 attributes (we considered the ‘journal’, ‘conference’, and ‘workshop’ attributes as one because it is an OR situation; filling one of them is considered sufficient information), only 9 attributes had values. The absence of the attribute values for application domain, project size, impact, RE practice, and RE phase, makes it difficult to assess the applicability of the lesson to other contexts. For reasons of anonymity, we have decided “not” to include citation and identification of the lessons learnt.

Table 6.11: An example of a low quality lesson from literature

Attribute	**Value**
ID	LL050_12
Journal	N/A
Conference	N/A
Workshop	MoDRE
Year	2012
Lesson	“When requirements are captured with models, for a variety of reasons it is necessary to maintain them in hierarchical databases.”
Source	Industrial experience
Rationale	“Because of the impedance mismatch between model structure (directed graph) and requirements database (tree structure).”
Impact	N/A
Target Object	Technique/method
Type	Negative
Expression	Explicit
Application Domain	N/A
Project Size	N/A
RE Practice	N/A
RE Phase	N/A
Software Process	N/A
Project Date	N/A
Recording Date	N/A
Organisation Name	N/A
Repeatability	N/A

Table 6.12 contains an example of a relatively high-quality lesson. Out of the 19 attributes, 16 of the values are given. The missing attributes (project date, recording date, and organization name) do not significantly affect the quality, and thus, applicability of the lesson. Project size and impact are given in detail, therefore, giving an idea of where to apply the lesson and what

to expect from applying it.

Table 6.12: An example of a high quality lesson from literature

Attribute	**Value**
ID	LL077_12
Journal	N/A
Conference	REFSQ
Workshop	N/A
Year	2012
Lesson	Use variability modeling that allows abstracting from requirements with AND, OR, and REQUIRES relationships to structure the release planning inputs.
Source	Case study.
Rationale	"Requirements catalogues for software release planning are often not complete and homogeneous. Current release planning approaches, however, assume such commitment to detail at least implicitly."
Impact	"The feature tree, in comparison with a flat backlog of requirements, reduced complexity of release planning. The abstraction from requirements to features reduced the total number of elements to be considered by a factor 10.3. The feature tree provided a basis to discuss the scope of pilot projects with the stakeholders identified in the stakeholder tree. Stakeholder needs that could not directly be addressed led to discovering new potential features. In comparison to a flat list of requirements, the feature tree allowed building a mental model of the solution. The reduced number of features allowed building a shared vocabulary with stakeholders, the color coding visualizing growth of the solution, and AND-OR feature dependencies understanding design options. This focused discussions and communication with stakeholders on aspects that were essential for planning. Decisions could be taken together with these stakeholders, which led to trust in the plans and in the product organization."
Target Object	Technique/method
Type	Positive
Expression	Implicit
Application Domain	Software as a service for managing media such as text, sound, pictures, and movies
Project Size	"Responsible for the development was a product manager, a project manager, and a team of up to five developers. The requirements catalogue was managed in a word processor document and used as a basis for release planning. It contained 108 requirements. The requirements were grouped into 12 sections and 19 subsections or themes. In average, a group contained 3.6 requirements and was allocated to 1.93 releases."
RE Practice	Modeling, using feature trees
RE Phase	Analysis
Software Process	Agile
Project Date	N/A
Recording Date	N/A
Organisation Name	N/A
Repeatability	First time case study results

Applying this method to asseses the quality of the elicited lessons will result in lessons of varying degrees of quality. None of the elicited lessons had all the values. However, the quality of the lessons varied between the two examples we gave above. This assessment dealt with the 'completeness' issue of a lesson, which is only one of aspect of quality we are planning to assess. As part of our future work, we are considering other quality aspects such as giving the attributes 'weights', as the presence of one attribute may be more important than another. For example, the attribute 'application domain' is more important than the attribute 'recording date'. Because the associated risk with not knowing which application domain the lessons emerged from is higher than that of not knowing the recording date of a lesson, which is used usually for technicalities within an organization. In addition, we intend to evaluate the sources of the lessons with relation to quality. A lesson from a controlled experiment, for example, may be considered of higher quality than one from a case study if a "causal" relationship is an integral part of the lesson learnt.

RQ 2.2 *Where (in which area of RE) are the lessons from the practice concentrated?*
The answers we have received so far from the survey (seven responses) are not sufficient for a complete analysis and interpretation. We will, however, discuss the available responses.

Although the number of lessons is small, the same trend we noticed in the results from the systematic literature review appears to hold true here. Most of the lessons are for the elicitation phase (all seven lessons) with an equal number of lessons for the analysis and specification phases (five lessons). Four of the lessons are for the prioritisation phase, three for negotiation and two for the validation and documentation phases.

Most of the lessons' (five lessons) target objects are 'process', then 'technique/method' (four lessons), 'policy' (three lessons), 'tool' (one lesson), 'project management' (one lesson), and 'communication' (one lesson).

The application domains were different for each lesson: banking, curriculum management, social websites, non-profit, e-learning, CRM system and CASE tool development (two lessons).

With regard to the RE practice related to the lessons, five lessons did not identify an RE practice and two indicated that the lesson is applicable to the 'whole RE process'.

The 'waterfall' software process had the most lessons (five lessons) iterative, spiral, code-and-fix have 4 lessons each, Agile has 3 lessons, and rapid prototyping has two lessons. The selection, however, may be affected by the fact that we listed all the different software processes for the respondent to choose from. If that hadn't been the case, these software processes may have not occurred to them.

The source of the all the lessons was 'industrial experience'. This, however, was expected as it is the aim of the survey to gather lessons from industry.

It is interesting to note that 6 out of the 7 lessons were 'negative'. If you recall from our discussion of the lesson types from the systematic literature review, the majority of the lessons were 'positive'. If this pattern continued with an increased number of responses, it would be interesting to explore the reasons behind this difference of lesson types between industry and research.

RQ 2.3 *What is the quality of the lessons learnt elicited from RE practice?*
When applying the same quality assessment method discussed above to the lessons elicited from practice, we will get the same lessons with varying degrees of quality. Most of the provided lessons, however, had all of the required values except for 'RE practice'. The least complete lesson is shown in table 6.13, which is missing 4 values out of a total of 14 attributes.

Table 6.13: An example of a low quality lesson from practice

Attribute	**Value**
ID	LL111_12
Year	2012
Lesson	-It is important the organization and writing of the requirements in order there are not misunderstandings among the customer and developer. - It is important that they are enough complete and detailed, in order the developer can not skip to implement some aspect of the system. - In systems with a lot of different user roles and IT previous knowledge it is impossible to satisfy the needs of all future users. It is the customer organization who has to finally decide when there are contradictory requirements, and do some activities to teach the users of the new system to get used to it and to the changes in processes that it implies.
Source	Other
Rationale	N/A
Impact	N/A
Target Object	Technique/method, process, policy
Type	Negative
Application Domain	Curriculum Management, other diverse domains...
Project Size	3000 users, other diverse sizes.
RE Practice	N/A
RE Phase	Elicitation, Specification
Software Process	Agile, Waterfall, Iterative, Spiral, Code-and-Fix, Rapid prototyping
Repeatability	I think they are both repeatable in all contexts.

RQ 2.4 *How do the lessons learnt from industry compare with the lessons learnt from literature?*

Although it is quite early to make a comparison, when comparing the available lessons from industry with the lessons from practice, one striking difference is that the lessons we collected from the survey tend to be more of a general nature, while the lessons elicited from the literature are more specific. A reasonable explanation would be that the respondents are sharing lessons ‘from the top of their heads’ in a survey and not specific lessons that they may encounter during a project. A different method for collecting lessons may yield different results.

Chapter 7: Populated RE Lesson Maps

In chapter 6 we presented the results of the empirical study (i.e., lessons elicited from literature and practice). In this chapter, we use the elicited lessons to populate the RE Lesson Maps we discussed in Chapter 4. As we've discussed previously, the Lesson Maps allow for different permutations of one or more attributes. We choose here the following five different permutations to demonstrate the distribution of lessons across selected attributes:

- **RE Phase** (section 7.1): to depict the distribution of lessons across the different RE phases. Although the discussion of the results in Chapter 6 identified where the lessons are concentrated, it would be enlightening to see it depicted visually in a RE Lesson Map.

- **RE Phase X Expression** (section 7.2): to depict the distribution of explicit and implicit lessons across the different RE phases. This will help in showcasing, for example, where the explicit lessons are concentrated (e.g., elicitation, validation, etc.)

- **Type X Source** (section 7.3): to depict the distribution of positive and negative lessons across the sources of lessons. This would help in identifying, for example, where the negative lessons mainly come from (e.g., industrial experience, case studies, etc.)

- **Target Object X RE Phase** (section 7.4): to depict the distribution of the lessons' target object across the RE phases. This would help in identifying, for example, where the lessons on tools are concentrated (e.g., elicitation, analysis, etc.).

- **Type X Expression X RE Phase** (section 7.5): to depict the distribution of negative and positive across the explicit and implicit lessons across the different RE phases. We can see then, for example, all the positive, explicit lessons under elicitation.

It is important to note that there are far more different permutations of RE Lesson Maps. We have selected the aforementioned permutations because the results are anticipated to yield interesting emerging hypotheses. In addition, it would be difficult to manually depict maps consisting of more than three attributes or attributes that have many values for the large number of lessons we have. This is one of the current limitations of our approach, which we discuss in Chapter 9. Another note here is that the maps were populated with the lessons elicited from literature only. Due to the small number of lessons we have received thus far from practice, we did not include them in the Lesson Maps.

7.1 Map 1: RE Phase

This map (Figure 7.1) depicts the distribution of all the elicited lessons across the different RE phases (elicitation, analysis, prioritisation, negotiation, specification, documentation, validation, verification, and managing). We can see that the lessons are concentrated in the elicitation, analysis phases, then the specification phase. The lessons in the managing and verification phases are scarce, which may indicate a lack of studies in those areas.

7.2 Map 2: RE Phase X Expression

Figure 7.2 depicts the the distribution of implicit and explicit lessons across the different RE phases. According to the results of the empirical study (Chapter 6), it is not a surprise that most of the lessons (in varying degrees) are implicit, with most of the implicit lessons falling under the elicitation and specification phases. The explicit lessons, however, are mainly concentrated under the analysis and specification phases, which indicates a conscious effort from the RE community to learn and share their lessons in these two phases as opposed to the other phase (e.g., prioritisation, negotiation, validation, verification and managing). The dearth of explicit lessons in the remaining phases may indicate that the RE community is not giving them adequate attention, thus, not learning enough.

7.3 Map 3: Type X Source

This map (Figure 7.3) depicts the distribution of negative, positive, and neutral lessons across the following sources: case study, controlled experiment, experiment, and industrial experience (we left out the sources with a very small number of lessons). Given the fact that most of the elicited lessons are positive and come from case studies (see Chapter 6), the concentration of lessons in the intersection of 'positive' and 'case study' comes as no surprise. However, an interesting observation worth noting is that, despite the relatively small number of negative lessons, most of the negative lessons come from industrial experience. This may suggest that researchers tend to share the positive experiences from their studies, while practitioners are more willing to share their negative experiences with the community. Although we did not include the lessons from practice (i.e., survey) in the Lesson Maps, if you recall from Chapter 6, 6 out of the lessons from industry were negative, which further supports this observation.

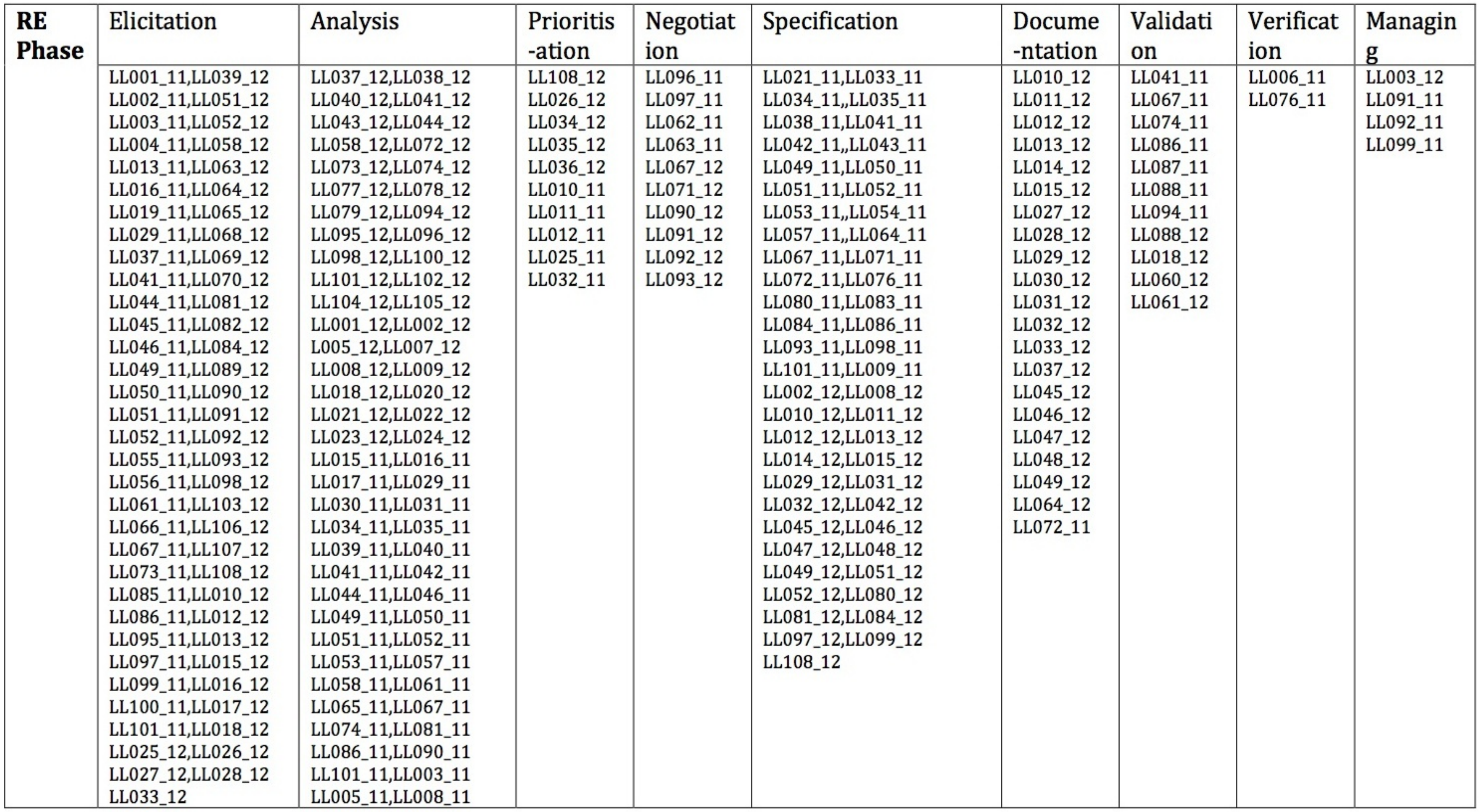
7.4 Map 4: Target Object X RE Phase

In this map (Figure 7.4), we were concerned with learning about the distribution of the lessons' target objects across the different RE phases. The map shows that most of the lessons are concerned with techniques/methods for the elicitation and analysis phases. While there is a relatively good number of lessons on techniques and methods for the specification, prioritisation, negotiation, and validation phases, we cannot say the same for the verification and managing phases. The lessons on tool are mainly for the analysis phase, then the elicitation and specification phases. The lessons on tools for the remaining phases are scarce. This is an interesting observation as the there is a good amount of RE tools for documenting and managing requirements. However, it seems that researchers and practitioners are not learning or sharing lessons in those areas. The lessons on the remaining target objects (policy, people, language, requirements, RE analysts) for all the RE phases are meager. Although 'people' (i.e. stakeholders) play an important role in the RE process, there doesn't seem to be any relevant lessons in the literature.

7.5 Map 5: Type X Expression X RE Phase

Figure 7.5 depicts the distribution of lessons across three dimensions: type, expression and elicitation. We mentioned in Chapters 3 and 4 that there is no restriction on the number of dimensions that a map may represent. However, due to the difficulty of depicting more than three dimensions in a tabular form with no tool support, we will show an example of a 3-dimensional map only. It can be observed that there are many more negative implicit lessons than explicit lessons. There are four phases (elicitation, negotiation, verification, and validation) that do not have any explicit, negative lessons. This again supports our previous claim that individuals and organisations, mainly researchers, may not be so keen on sharing negative experiences explicitly. The only case where there are more explicit lessons than implicit ones are under the 'documentation' phase. The positive explicit lessons are more than than the negative ones.

We are aware that lessons from two-years worth of literature may not be representative of the state of lessons learnt in RE; however, these maps provide a good starting point to build upon for future studies. Further studies to support or disprove these studies will be helpful. In addition, empirical studies that study the reasons and effects of the presented observations may aid in improving RE processes. Moreover, these depictions illustrate the potential benefits of the proposed maps and aids us in achieving our overall research objective (i.e, learning the state of lessons learnt in RE).

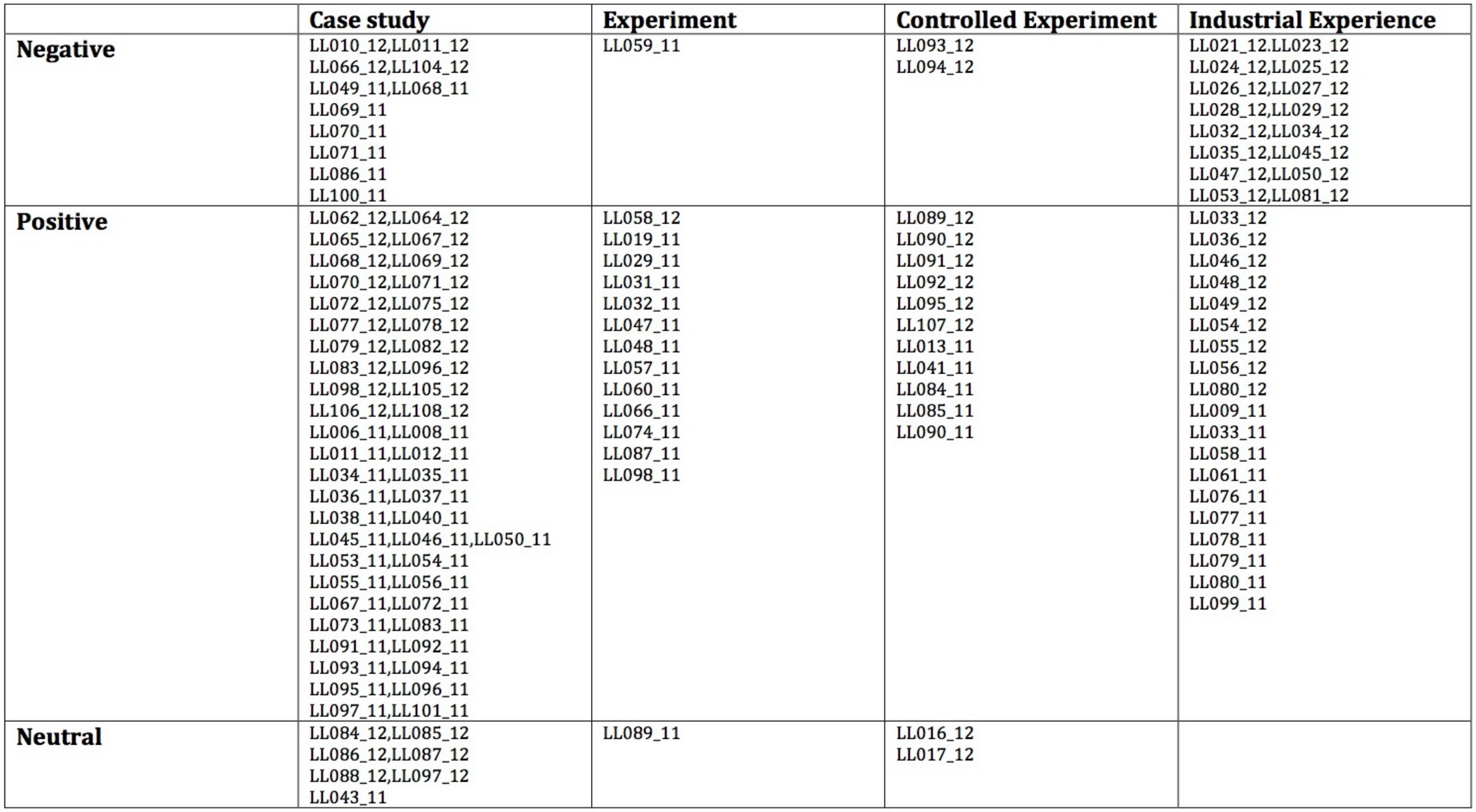


RE Phase	Elicitation	Analysis	Prioritis-ation	Negotiat ion	Specification	Docume-ntation	Validati on	Verificat ion	Managin g
	LL001_11,LL039_12	LL037_12,LL038_12	LL108_12	LL096_11	LL021_11,LL033_11	LL010_12	LL041_11	LL006_11	LL003_12
	LL002_11,LL051_12	LL040_12,LL041_12	LL026_12	LL097_11	LL034_11,,LL035_11	LL011_12	LL067_11	LL076_11	LL091_11
	LL003_11,LL052_12	LL043_12,LL044_12	LL034_12	LL062_11	LL038_11,LL041_11	LL012_12	LL074_11		LL092_11
	LL004_11,LL058_12	LL058_12,LL072_12	LL035_12	LL063_11	LL042_11,,LL043_11	LL013_12	LL086_11		LL099_11
	LL013_11,LL063_12	LL073_12,LL074_12	LL036_12	LL067_12	LL049_11,LL050_11	LL014_12	LL087_11		
	LL016_11,LL064_12	LL077_12,LL078_12	LL010_11	LL071_12	LL051_11,LL052_11	LL015_12	LL088_11		
	LL019_11,LL065_12	LL079_12,LL094_12	LL011_11	LL090_12	LL053_11,,LL054_11	LL027_12	LL094_11		
	LL029_11,LL068_12	LL095_12,LL096_12	LL012_11	LL091_12	LL057_11,,LL064_11	LL028_12	LL088_12		
	LL037_11,LL069_12	LL098_12,LL100_12	LL025_11	LL092_12	LL067_11,LL071_11	LL029_12	LL018_12		
	LL041_11,LL070_12	LL101_12,LL102_12	LL032_11	LL093_12	LL072_11,LL076_11	LL030_12	LL060_12		
	LL044_11,LL081_12	LL104_12,LL105_12			LL080_11,LL083_11	LL031_12	LL061_12		
	LL045_11,LL082_12	LL001_12,LL002_12			LL084_11,LL086_11	LL032_12			
	LL046_11,LL084_12	L005_12,LL007_12			LL093_11,LL098_11	LL033_12			
	LL049_11,LL089_12	LL008_12,LL009_12			LL101_11,LL009_11	LL037_12			
	LL050_11,LL090_12	LL018_12,LL020_12			LL002_12,LL008_12	LL045_12			
	LL051_11,LL091_12	LL021_12,LL022_12			LL010_12,LL011_12	LL046_12			
	LL052_11,LL092_12	LL023_12,LL024_12			LL012_12,LL013_12	LL047_12			
	LL055_11,LL093_12	LL015_11,LL016_11			LL014_12,LL015_12	LL048_12			
	LL056_11,LL098_12	LL017_11,LL029_11			LL029_12,LL031_12	LL049_12			
	LL061_11,LL103_12	LL030_11,LL031_11			LL032_12,LL042_12	LL064_12			
	LL066_11,LL106_12	LL034_11,LL035_11			LL045_12,LL046_12	LL072_11			
	LL067_11,LL107_12	LL039_11,LL040_11			LL047_12,LL048_12				
	LL073_11,LL108_12	LL041_11,LL042_11			LL049_12,LL051_12				
	LL085_11,LL010_12	LL044_11,LL046_11			LL052_12,LL080_12				
	LL086_11,LL012_12	LL049_11,LL050_11			LL081_12,LL084_12				
	LL095_11,LL013_12	LL051_11,LL052_11			LL097_12,LL099_12				
	LL097_11,LL015_12	LL053_11,LL057_11			LL108_12				
	LL099_11,LL016_12	LL058_11,LL061_11							
	LL100_11,LL017_12	LL065_11,LL067_11							
	LL101_11,LL018_12	LL074_11,LL081_11							
	LL025_12,LL026_12	LL086_11,LL090_11							
	LL027_12,LL028_12	LL101_11,LL003_11							
	LL033_12	LL005_11,LL008_11							

Figure 7.1: ‘RE Phase’ Lesson Map

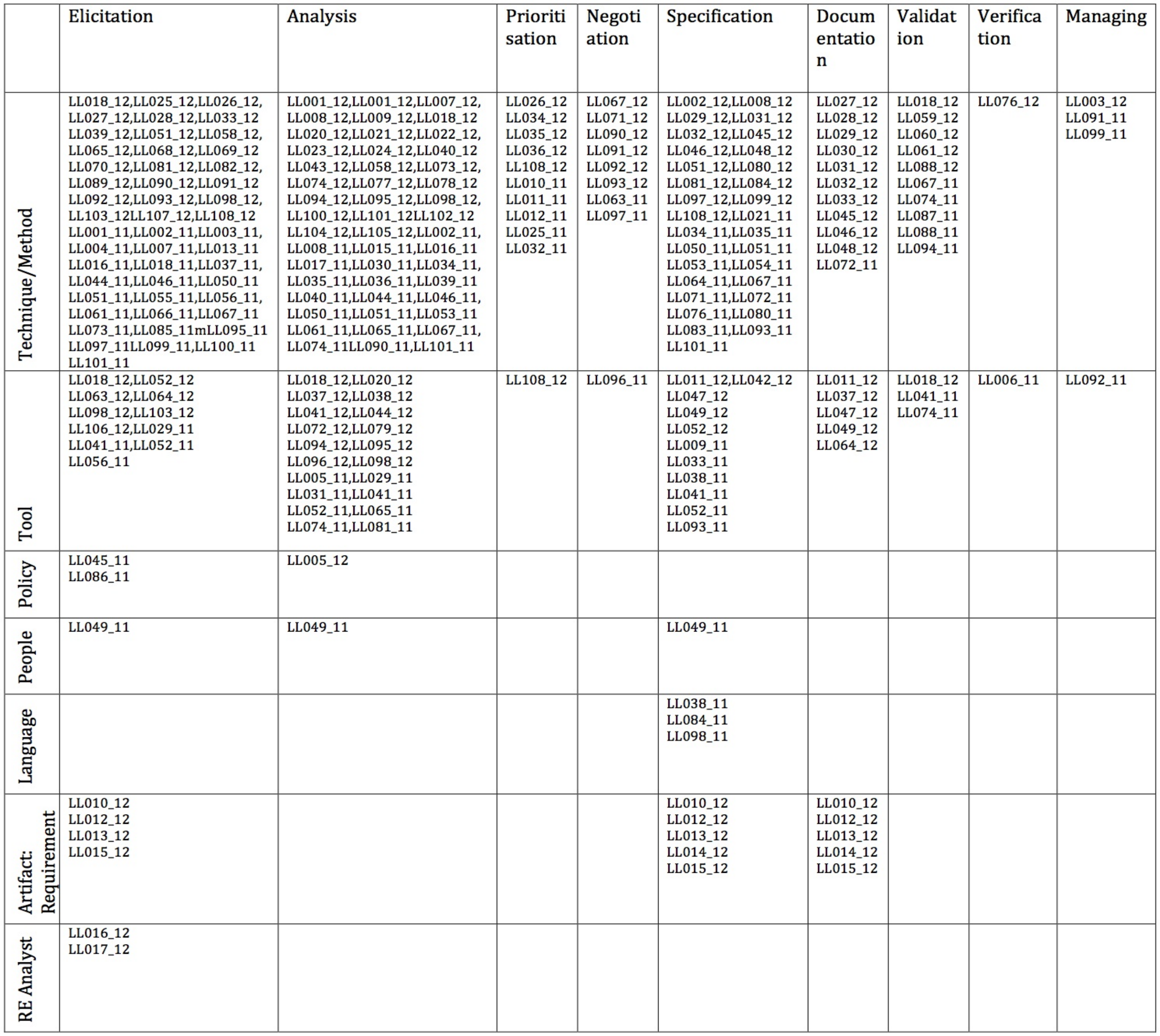
	Elicitation	Analysis	Prioriti sation	Negoti ation	Specification	Documentation	Validat ion	Verificat ion	Managi ng
Explicit	LL033_12,LL084_12 LL002_11mLL016_11 LL045_11,LL046_11 LL073_11 LL095_11	LL022_12,LL023_12, LL024_12,LL043_12, LL015_11,LL016_11 LL017_11,LL034_11, LL035_11,LL036_11, LL046_11,LL058_11	LL035_12 LL036_12		LL029_12,LL031_12 LL045_12,LL046_12 LL047_12,LL048_12 LL049_12,LL033_11 LL034_11,LL035_11 LL071_11,LL072_11 LL080_11	LL029_12,LL030_12 LL031_12,LL033_12 LL045_12,LL046_12 LL047_12,LL048_12 LL049_12,LL072_11	LL088_12		LL091_11
Implicit	LL010_12,LL012_12 LL013_12,LL015_12 LL016_12,LL017_12 LL018_12,LL025_12 LL026_12,LL027_12 LL029_12,LL039_12 LL051_12,LL052_12 LL058_12,LL063_12 LL064_12,LL066_12 LL068_12,LL069_12 LL070_12,LL081_12 LL082_12,LL089_12 LL090_12,LL091_12 LL092_12,LL093_12 LL098_12,LL103_12 LL106_12,LL107_12 LL108_12,LL001_11 LL003_11,LL004_11 LL007_11,LL013_11 LL018_11,LL029_11 LL037_11,LL041_11 LL044_11,LL049_11 LL050_11,LL051_11 LL052_11,LL055_11 LL056_11,LL061_11 LL066_11,LL067_11 LL085_11,LL086_11 LL097_11,LL099_11 LL100_11,LL101_11	LL001_12,LL002_12 LL005_12,LL007_12 LL008_12,LL009_12 LL018_12,LL020_12 LL021_12,LL037_12 LL038_12,LL040_12 LL041_12,LL044_12 LL058_12,LL072_12 LL073_12,LL074_12 LL077_12,LL078_12 LL079_12,LL094_12 LL095_12,LL096_12 LL098_12,LL100_12 LL101_12,LL102_12 LL104_12,LL105_12 LL005_11,LL008_11 LL029_11,LL030_11 LL031_11,LL039_11 LL040_11,LL041_11 LL042_1LL044_11 LL049_11,LL050_11 LL051_11,LL052_11 LL053_11,LL057_11 LL061_11,LL065_11 LL067_11,LL074_11 LL081_11,LL086_11 LL090_11 LL101_11	LL026_12 LL034_12 LL108_12 LL010_11 LL011_11 LL012_11 LL025_11 LL032_11	LL067_12 LL071_12 LL091_12 LL092_12 LL093_12 LL062_11 LL063_11 LL096_11 LL097_11	LL002_12,LL008_12 LL010_12,LL011_12 LL012_12,LL013_12 LL014_12,LL015_12 LL032_12,LL042_12 LL051_12,LL052_12 LL080_12,LL081_12 LL097_12,LL099_12 LL108_12,LL009_11 LL021_11,LL038_11 LL041_11,LL042_11 LL043_11,LL049_11 LL050_11,LL051_11 LL052_11,LL053_11 LL057_11,LL064_11 LL067_11,LL076_11 LL083_11,LL084_11 LL086_11 LL093_11 LL098_11 LL101_11	LL010_12 LL011_12 LL012_12 LL013_12 LL014_12 LL015_12 LL027_12 LL028_12 LL032_12 LL037_12 LL064_12	LL018_12 LL059_12 LL060_12 LL061_12 LL041_11 LL067_11 LL074_11 LL086_11 LL087_11 LL088_11 LL095_11	LL076_12 LL006_11	LL003_12 LL092_11 LL099_11

Figure 7.2: ‘RE Phase X Expression’ Lesson Map



	Case study	Experiment	Controlled Experiment	Industrial Experience
Negative	LL010_12,LL011_12 LL066_12,LL104_12 LL049_11,LL068_11 LL069_11 LL070_11 LL071_11 LL086_11 LL100_11	LL059_11	LL093_12 LL094_12	LL021_12.LL023_12 LL024_12,LL025_12 LL026_12,LL027_12 LL028_12,LL029_12 LL032_12,LL034_12 LL035_12,LL045_12 LL047_12,LL050_12 LL053_12,LL081_12
Positive	LL062_12,LL064_12 LL065_12,LL067_12 LL068_12,LL069_12 LL070_12,LL071_12 LL072_12,LL075_12 LL077_12,LL078_12 LL079_12,LL082_12 LL083_12,LL096_12 LL098_12,LL105_12 LL106_12,LL108_12 LL006_11,LL008_11 LL011_11,LL012_11 LL034_11,LL035_11 LL036_11,LL037_11 LL038_11,LL040_11 LL045_11,LL046_11,LL050_11 LL053_11,LL054_11 LL055_11,LL056_11 LL067_11,LL072_11 LL073_11,LL083_11 LL091_11,LL092_11 LL093_11,LL094_11 LL095_11,LL096_11 LL097_11,LL101_11	LL058_12 LL019_11 LL029_11 LL031_11 LL032_11 LL047_11 LL048_11 LL057_11 LL060_11 LL066_11 LL074_11 LL087_11 LL098_11	LL089_12 LL090_12 LL091_12 LL092_12 LL095_12 LL107_12 LL013_11 LL041_11 LL084_11 LL085_11 LL090_11	LL033_12 LL036_12 LL046_12 LL048_12 LL049_12 LL054_12 LL055_12 LL056_12 LL080_12 LL009_11 LL033_11 LL058_11 LL061_11 LL076_11 LL077_11 LL078_11 LL079_11 LL080_11 LL099_11
Neutral	LL084_12,LL085_12 LL086_12,LL087_12 LL088_12,LL097_12 LL043_11	LL089_11	LL016_12 LL017_12	

Figure 7.3: ‘Type X Source’ Lesson Map



	Elicitation	Analysis	Prioritisation	Negotiation	Specification	Documentation	Validation	Verification	Managing
Technique/Method	LL018_12,LL025_12,LL026_12, LL027_12,LL028_12,LL033_12 LL039_12,LL051_12,LL058_12, LL065_12,LL068_12,LL069_12 LL070_12,LL081_12,LL082_12, LL089_12,LL090_12,LL091_12 LL092_12,LL093_12,LL098_12, LL103_12LL107_12,LL108_12 LL001_11,LL002_11,LL003_11, LL004_11,LL007_11,LL013_11 LL016_11,LL018_11,LL037_11, LL044_11,LL046_11,LL050_11 LL051_11,LL055_11,LL056_11, LL061_11,LL066_11,LL067_11 LL073_11,LL085_11mLL095_11 LL097_11LL099_11,LL100_11 LL101_11	LL001_12,LL001_12,LL007_12, LL008_12,LL009_12,LL018_12 LL020_12,LL021_12,LL022_12, LL023_12,LL024_12,LL040_12 LL043_12,LL058_12,LL073_12, LL074_12,LL077_12,LL078_12 LL094_12,LL095_12,LL098_12, LL100_12,LL101_12LL102_12 LL104_12,LL105_12,LL002_11, LL008_11,LL015_11,LL016_11 LL017_11,LL030_11,LL034_11, LL035_11,LL036_11,LL039_11 LL040_11,LL044_11,LL046_11, LL050_11,LL051_11,LL053_11 LL061_11,LL065_11,LL067_11, LL074_11LL090_11,LL101_11	LL026_12 LL034_12 LL035_12 LL036_12 LL108_12 LL010_11 LL011_11 LL012_11 LL025_11 LL032_11	LL067_12 LL071_12 LL090_12 LL091_12 LL092_12 LL093_12 LL063_11 LL097_11	LL002_12,LL008_12 LL029_12,LL031_12 LL032_12,LL045_12 LL046_12,LL048_12 LL051_12,LL080_12 LL081_12,LL084_12 LL097_12,LL099_12 LL108_12,LL021_11 LL034_11,LL035_11 LL050_11,LL051_11 LL053_11,LL054_11 LL064_11,LL067_11 LL071_11,LL072_11 LL076_11,LL080_11 LL083_11,LL093_11 LL101_11	LL027_12 LL028_12 LL029_12 LL030_12 LL031_12 LL032_12 LL033_12 LL045_12 LL046_12 LL048_12 LL072_11	LL018_12 LL059_12 LL060_12 LL061_12 LL088_12 LL067_11 LL074_11 LL087_11 LL088_11 LL094_11	LL076_12	LL003_12 LL091_11 LL099_11
Tool	LL018_12,LL052_12 LL063_12,LL064_12 LL098_12,LL103_12 LL106_12,LL029_11 LL041_11,LL052_11 LL056_11	LL018_12,LL020_12 LL037_12,LL038_12 LL041_12,LL044_12 LL072_12,LL079_12 LL094_12,LL095_12 LL096_12,LL098_12 LL005_11,LL029_11 LL031_11,LL041_11 LL052_11,LL065_11 LL074_11,LL081_11	LL108_12	LL096_11	LL011_12,LL042_12 LL047_12 LL049_12 LL052_12 LL009_11 LL033_11 LL038_11 LL041_11 LL052_11 LL093_11	LL011_12 LL037_12 LL047_12 LL049_12 LL064_12	LL018_12 LL041_11 LL074_11	LL006_11	LL092_11
Policy	LL045_11 LL086_11	LL005_12							
People	LL049_11	LL049_11			LL049_11				
Language					LL038_11 LL084_11 LL098_11				
Artifact: Requirement	LL010_12 LL012_12 LL013_12 LL015_12				LL010_12 LL012_12 LL013_12 LL014_12 LL015_12	LL010_12 LL012_12 LL013_12 LL014_12 LL015_12			
RE Analyst	LL016_12 LL017_12								

Figure 7.4: ‘Target Object X RE Phase’ Lesson Map

	Elicitation		Analysis		Prioritisation		Negotiation		Specification		Documentation		Verification		Validation	
	Explicit	Implicit	Explicit	Implicit	Explicit	Implicit	Explicit	Implicit	Explicit	Implicit	Explicit	Implicit	Explicit	Implicit	Explicit	Implicit
Negative		LL010_12,LL012_12 LL013_12,LL015_12 LL025_12,LL026_12 LL027_12,LL028_12 LL081_12,LL093_12 LL004_11,LL049_11 LL086_11,LL100_11	LL023_12 LL024_12	LL021_12 LL038_12 LL094_12 LL104_12 LL039_11 LL049_11 LL086_11	LL035_12	LL026_12 LL034_12		LL093_12 LL062_11	LL029_12 LL045_12 LL047_12 LL071_11	LL010_12 LL011_12 LL012_11 LL013_12 LL014_12 LL015_12 LL032_12 LL081_12 LL049_11 LL086_11	LL029_12 LL045_12 LL047_12	LL010_12 LL011_12 LL012_12 LL013_12 LL014_12 LL015_12 LL027_12 LL028_12 LL032_12				LL060_12 LL061_12 LL086_11
Positive	LL033_12 LL002_11 LL016_11 LL045_11 LL046_11 LL073_11 LL095_11	LL018_12,LL039_12 LL051_12,LL052_12 LL058_12,LL063_12 LL064_12,LL065_12 LL067_12,LL069_12 LL070_12,LL082_12 LL089_12,LL090_12 LL091_12,LL092_12 LL098_12,LL103_12 LL106_12,LL107_12 LL108_12,LL001_11 LL003_11,LL007_11 LL013_11,LL018_11 LL029_11,LL037_11 LL041_11,LL050_11 LL051_11,LL052_11 LL055_11,LL056_11 LL061_11,LL066_11 LL067_11,LL085_11 LL097_11,LL099_11 LL101_11	LL002_11 LL014_11 LL015_11 LL016_11 LL017_11 LL034_11 LL035_11 LL036_11 LL046_11 LL058_11	LL001_12 LL002_12 LL005_12 LL007_12 LL009_12 LL018_12 LL020_12 LL040_12 LL041_12 LL044_12 LL058_12 LL072_12 LL073_12 LL074_12 LL077_12 LL078_12 LL079_12 LL095_12 LL096_12 LL098_12 LL100_12 LL101_12 LL102_12 LL105_12 LL005_11 LL008_11 LL029_11 LL030_11 LL031_11 LL040_11 LL041_11 LL050_11 LL051_11 LL052_11 LL053_11 LL057_11 LL061_11 LL067_11 LL074_11 LL081_11	LL036_12	LL108_12		LL067_12 LL071_12 LL090_12 LL091_12 LL092_12 LL063_11 LL096_11 LL097_11	LL046_12 LL048_12 LL049_12 LL033_11 LL034_11 LL035_11 LL072_11 LL080_11	LL042_12 LL051_12 LL052_12 LL080_12 LL108_12 LL009_11 LL021_11 LL038_11 LL041_11 LL043_11 LL050_11 LL051_11 LL052_11 LL053_11 LL054_11 LL057_11 LL067_11 LL076_11 LL083_11 LL084_11 LL093_11 LL098_11 LL101_11	LL033_12 LL046_12 LL048_12 LL049_12 LL072_11	LL064_12		LL076_12 LL006_11		LL018_12 LL059_12 LL041_11 LL067_11 LL074_11 LL087_11 LL088_11 LL094_11
Neutral	LL084_12	LL016_12,LL044_11 LL017_12	LL022_12 LL043_12	LL008_12 LL037_12 LL042_11 LL044_11					LL031_12 LL084_12	LL008_12 LL097_12	LL030_12 LL031_12	LL037_12			LL088_12	

Figure 7.5: ‘Type X Expression X RE Phase’ Lesson Map

Chapter 8: Implications

This chapter describes the implications of the results of our study. Sections 8.1 and 8.2 discuss the implication on research and practice respectively.

8.1 Implications on Research

This study has implications on research in several dimensions. Firstly, the observations made from the populated maps are anticipated to promulgate further research in order to either support or refute the observed trends. Further empirical studies are needed to understand the drivers and effects of these trends on practice, project time, cost and quality, which may lead to the creation of new RE theories through lesson-driven feedback from practice.

Secondly, to our knowledge, such a comprehensive list of lessons learnt does not exist. Therefore, our list of lessons adds significantly to the current RE body of knowledge.

Thirdly, during our discussion of the related work (see Chapter 2), we demonstrated how the topic of lessons learnt has been given relatively significant attention in non-software engineering disciplines at large and in software engineering in general. The same, however, cannot be said about requirements engineering. Therefore, we hope that this work and our future work on the topic raise awareness among individuals in the RE community about the importance of lessons learnt in RE encouraging them to take action to bring the concept of lessons and lesson maps to life in the RE field.

Finally, as we've seen in Chapters 6 and 7, the large number of lessons and the complexity of the resultant maps call for research on tools to support the storage, operationalisation, and management of the lessons and lesson maps. Although researchers have proposed tools for lessons learnt in other domains (e.g., Lessons Learnt System for Software Testing [Andrade et al., 2013]), to our knowledge, no RE specific lessons learnt tools are available, especially those that support our lessons representation and lesson maps (see Appendix D for some initial ideas for a RE lesson learnt tool).

8.2 Implications on Practice

The proposed maps and the elicited lessons are anticipated to change the survey statistics favourably (see Appendix A): increased use and creation of RE lessons in projects; increased sharing of lessons; simplified access to lessons; etc. This would promote a grass-roots discipline of RE lessons across people, projects, applications, domains, etc. The utilisation of the lessons and lesson maps in practice will aid RE practitioners during the RE process and consequently, improve it. The effects of this improvement is anticipated to be felt upon overall project costs, quality, and time. It is important to note that the presented concepts are generic enough for use outside RE thus promoting this philosophy elsewhere in a project.

Chapter 9: Limitations, Future Work and Conclusions

Section 9.1 of this chapter discusses the limitations of our study and the ongoing future work we intend to carry to address these limitations. Finally, Section 9.2 concludes the thesis.

9.1 Limitations and Future Work

In Chapter 2, our analysis of the related work showed that lessons learnt have not received significant attention in the field of RE. To the best of our knowledge, this study is the first of its kind that extensively studied the topic of lessons learnt specifically within the RE domain. The novel solution (i.e., RE lesson maps) and knowledge (i.e., elicited lessons) contribute to scientific body of knowledge in RE. However, it is important to note that our study has its limitations and researchers and practitioners are encouraged to take caution when generalising the results of our study in research and practice.

In this study, we presented our concept of a RE lesson and RE lesson map, which were based on scientific groundings, and which were validated by several experts (see Section 4.3). Despite these efforts, further research and feedback from both researchers and practitioners is needed to develop a well-rounded and mature concept of lessons and lesson maps that can be utilised in practice. To address this issue, we intend to get further feedback from research and industry as part of our going future work by publishing further peer-reviewed papers and conducting empirical studies in industrial settings to validate our concepts for consistency, completeness and usefulness.

Another limitation of this study is concerned with the results of the empirical studies (i.e. elicited lessons learnt). First of all, the lessons from the systematic literature review is not representative of all the RE literature. We, therefore, intend to expand on the resources used for the SLR by including more journals, conference and workshop proceedings, from more years. Second of all, due to the fact that the systematic literature review was conducted by one researcher, the results of the study may be prone to researcher bias (i.e., overlooking some lessons and including others that may not be considered a lesson). To deal with this limitation, we plan on including more researchers in the study to validate the elicited lessons. Thirdly, the lessons we have received to date from industry are very small in number. Distributing the survey to more participants of diverse backgrounds in RE and using other methods to gather RE lessons from practitioners (e.g., interviews, workshops, brainstorming sessions) are part of our ongoing work.

Finally, we saw in Chapter 6 that the size of the results from only two-year literature is rather large. Therefore, the increased number of elicited lessons complexity of maps will increase the difficulty with which we manage, store and operate on the lessons and lesson maps. In addition, more advanced features and operations will facilitate the use of lessons and lesson maps in practice. Hence, adequate tool support is needed to deal with these concerns (see Appendix D for a working concept of a LL tool).

However, the discussed limitations do not diminish the importance of our results as they are considered a first step towards laying the groundwork for lessons learnt in RE.

9.2 Conclusions

Lessons learnt have been explored in many non-software engineering disciplines such as education [Bodycott and Walker, 2001], management [Lee, 2008], medicine [Rogers et al., 2001] and others. In software engineering lessons learnt also, lessons learnt have received significant attention. Researchers and practitioners in software engineering explicitly share their lessons (e.g., [Basili et al., 2002, Boehm, 2006, Dick and Woods, 1997]) and propose methods, processes, and tools (e.g., [Andrade et al., 2013, Weber et al., 2001, Vandeville and Shaikh, 1999]) for lessons learnt. On the other hand, lessons learnt in RE, to our knowledge, have not yet been systematically explored. Our research objective was to understand and determine the state of lessons learnt in RE and promote their use by creating a scientific basis for the structuring and organization of lessons embodied in the concept of "Lesson Maps" and populating them with lessons elicited from the literature and practice. To achieve this objective, we presented, in this thesis, our solution-building (i.e., RE Lesson Maps) and knowledge-seeking (i.e., empirical study) work on lessons learnt, which is the first of its kind in the field.

Our presented structured representation of a RE lesson provides a means to encapsulate a lesson along with its context information to allow for use in similar situations. The Lesson Maps provide a means to organise lessons in a way that shows the distribution of RE lessons across different RE subareas, application domains, RE practices, etc. These maps, once populated, will help us in better understanding the state of lessons learnt in RE. The concepts of a RE lesson and lesson map have been peer-reviewed and validated by senior researchers who gave positive feedback with regard to its potential usefulness and benefits.

To achieve our research objective of understanding the state of lessons learnt in RE, we populated the RE Lesson Maps with lessons elicited from literature. We found 209 lessons from two-year literature (2011 and 2012). Out of the 209 lessons, only 47 (22%) lessons were explicitly mentioned in the literature. The remaining 78% of the lesson were implicit in the literature (see Table 6.2, Chapter 6). This may indicate that the RE community is not explicitly documenting their lessons learnt.

The lessons were mainly concentrated in the elicitation (32%), analysis (35%), and specification (25%) RE phases (see Table 6.3, Chapter 6). The lessons for the remaining RE phases were very small. The results also showed that 76% and 21% of the lessons targeted techniques/methods and tools for the RE process, respectively (see Table 6.4, Chapter 6). 47% of the lessons came from case studies, 22% from some form of experiment, and 19% from industrial experience (see Table 6.8, Chapter 6). 70% of the lessons were positive, 20% negative, and 10% neutral (see Table 6.9, Chapter 6). One interesting finding from populating the lesson maps was that the majority of negative lessons came from industrial experience (see Figure 7.3, Chapter 7).

Our concepts of a lesson and lesson map and the results from the empirical study have implications on both practice and industry. The elicited lessons and proposed lesson maps are anticipated to aid practitioners in their RE processes, and consequently, improve them. The results from the empirical study add to the scientific body of knowledge in the RE field. This calls for further empirical studies to validate and better understand the consequences of our results.

Our ongoing future work is intended to address some of the limitations of our study, such as the two-year literature that has been reviewed, which will be expanded by adding resources from more years. More lessons will also be elicited from practice as the current number of lessons is very small. Finally, further feedback from researchers and practitioners will be obtained to develop and mature our concepts of RE lessons and lesson maps.

From the above, we conclude that:

- Lessons maps (the solution-seeking part of this work) are a promising way to obtain an impression of the light and dark areas of RE.
- The populated maps indicate that approximately only 20% of the elicited lessons are explicitly described in the literature and 80% are implicit (See Table 6.2, Chapter 6). There is thus a lot of room to reverse the trend.
- Due to the potentially increasing number of lessons learnt and thus, the complexity of the lesson maps, there is a need for tools for lessons learnt in RE as they are non-existent

–even as prototypes. This is much needed if we are to promote lessons learnt in daily practice of RE.

- Lesson quality is an important aspect if we are to ensure usability and reuse of the lessons learnt.

Appendix A: A Survey of Lessons Learnt

The following survey was used to get a better understanding of the state of lessons learnt in practice.

The Survey:

Demographic Information

Type of organization you work in (Please choose all that apply in your case):

- Industry
- Governmental organization
- Academic institution
- Other (please specify)

Number of years of experience in software/system development or IT:

- 1-4 years
- 5-10 years
- 11-15 years
- 16 + years

Number of years of experience as a Requirements Analyst or Business Analyst (or related area):

- None
- 1-4 years

- 5-10 years
- 11-15 years
- 16 + years

Key roles played in the organization (Please choose all that apply in your case):

- Manager
- Requirements Analyst/Engineer
- Business Analyst
- Developer
- Architect
- Other (please specify)

Survey Questions - There are nine questions

1. What do you understand by the term ”lesson”? Please choose all that you think that apply.
    - Positive lesson: A successful experience that encourages the same/similar behavior in a similar situation in the future to achieve the same/similar observed results.
    - Negative lesson: An unpleasant experience that requires different behavior in a similar situation in the future to avoid the observed results.
    - Other (please write your definition of a ”lesson”)
2. How important do you think are lessons learned in Requirements Engineering (RE) for your organization’s ”RE” processes? (Note that lessons can come from sources such as: past projects, people, websites, literature, etc.)

- Very important
- Somewhat important
- Unimportant
- Other (please specify):

3. How often are RE lessons learned actually used in your organization's RE processes?

    - Frequently
    - Occasionally
    - Hardly ever
    - Other (please specify):

4. If the practice of using RE lessons learned in projects is ingrained in your organisation, please indicate the sources from which these lessons were (or are being) obtained:

    - Development projects, including methods, techniques, tools, processes, artifacts, products, etc.
    - People
    - Websites
    - Peer-reviewed scientific literature
    - Books
    - Technical reports
    - Other sources (please indicate which):

5. If your organization uses RE lessons in projects, how are these lessons shared within the organization? Please select all that apply:

    - Database, knowledge base, web server, or other form of formal storage of lessons learned.

- Formally communicated in workshops, training or tutorial sessions and the like.
- Informal sharing (e.g., verbally communicated by individuals).
- Other (please specify):

6. How difficult is it for your organisation to find, gather, elicit or get access to RE lessons learned from whichever sources? Please indicate this in the table below.

	Very difficult	**Difficult**	**Manageable**	**Easy**	**Very Easy**
Development projects					
People					
Websites					
Peer-reviewed scientific literature					
Negotiation					
Books					
Technical reports					
Other sources					

7. If your organization does NOT use (or seldom uses) RE lessons learned in projects, would it actually use them if they were made readily available? Please choose one of the following:

   - Yes
   - Maybe (please explain)
   - No (please indicate why)

8. What do you think is the impact of not (or seldom) utilising lessons learned in projects, in terms of productivity loss, project delays, cost overruns, etc.? Please choose the level of impact for each attribute in the table below.

	Very High	High	Medium	Low	Very Low
Productivity loss					
Project delays					
Cost overruns					
Product quality problems					
Repeating mistakes					
Opportunities lost					
Project failure					
Customer dissatisfaction					

9. Any other comment you would like to make?

Results:

Type of organization you work in (Please choose all that apply in your case):

Value	Count	Percentage
Industry	36	80.0%
Governmental organization	3	6.7%
Academic institution	13	28.9%
Other (please specify)	8	17.8%

Number of years of experience in software/system development or IT:

Value	Count	Percentage
1-4 years	6	13.3%
5-10	16	35.6%
11-15	10	22.2%
16+ years	13	28.9%

Number of years of experience as a Requirements Analyst or Business Analyst (or related area):

Value	Count	Percentage
None	5	11.1%
1-4 years	15	33.3%
5-10	13	28.9%
11-15	7	15.6%
16+ years	5	11.1%

Key roles played in the organization (Please choose all that apply in your case):

Value	**Count**	**Percentage**
Manager	23	51.1%
Requirements Analyst/Engineer	30	66.7%
Business Analyst	13	28.9%
Developer	20	44.4%
Other (please specify)	13	28.9%

1. What do you understand by the term ”lesson”? Please choose all that you think that apply.

Value	**Count**	**Percentage**
Positive	36	81.8%
37	30	84.1%
Other (please write your definition of a “lesson”)	10	22.7%

2. How important do you think are lessons learned in Requirements Engineering (RE) for your organization’s ”RE” processes? (Note that lessons can come from sources such as: past projects, people, websites, literature, etc.)

Value	**Count**	**Percentage**
Very important	26	59.1%
Somewhat important	14	31.8%
Unimportant	1	2.3%
Other (please specify)	3	6.8%

3. How often are RE lessons learned actually used in your organization’s RE processes?

Value	**Count**	**Percentage**
Frequently	13	29.6%
Occasionally	21	47.7%
Hardly ever	10	22.7%
Other (please specify)	0	0.0%

4. If the practice of using RE lessons learned in projects is ingrained in your organisation, please indicate the sources from which these lessons were (or are being) obtained:

Value	Count	Percentage
Development projects	32	82.1%
People	34	87.2%
Websites	3	7.7%
Peer-reviewed scientific literature	5	12.8%
Books	9	23.1%
Technical reports	5	12.8%
Other sources (please indicate which)	6	15.4%

5. If your organization uses RE lessons in projects, how are these lessons shared within the organization? Please select all that apply:

Value	Count	Percentage
Database, knowledge base, web server, or other form of formal storage	14	33.3%
Formally communicated in workshops, training or tutorial sessions	16	38.1%
Informal sharing	38	90.5%
Other (please specify)	3	7.1%

6. How difficult is it for your organisation to find, gather, elicit or get access to RE lessons learned from whichever sources? Please indicate this in the table below.

	Very difficult	Difficult	Manageable	Easy	Very Easy
Development projects	3 (7.1%)	7 (16.7%)	18 (42.9%)	10 (23.8%)	4 (9.5%)
People	1 (2.5%)	6 (15.0%)	15 (37.5%)	8 (20.0%)	10 (25.0%)
Websites	4 (10.3%)	13 (33.3%)	14 (35.9%)	7 (17.9%)	1 (2.6%)
Peer-reviewed scientific literature	6 (15.0%)	16 (40.0%)	8 (20.0%)	10 (25.0%)	0 (0.0%)
Books	3 (7.9%)	9 (23.7%)	14 (36.8%)	12 (31.6%)	0 (0.0%)
Technical reports	7 (18.4%)	10 (26.3%)	15 (39.5%)	6 (15.8%)	0 (0.0%)
Other sources	3 (11.5%)	8 (30.8%)	14 (53.8%)	1(3.8%)	0 (0.0%)

7. If your organization does NOT use (or seldom uses) RE lessons learned in projects,

would it actually use them if they were made readily available? Please choose one of the following:

Value	Count	Percentage
Yes	12	46.2%
Maybe (please explain)	10	38.5%
No (please indicate why)	4	15.4%

8. What do you think is the impact of not (or seldom) utilising lessons learned in projects, in terms of productivity loss, project delays, cost overruns, etc.? Please choose the level of impact for each attribute in the table below.

	Very High	High	Medium	Low	Very Low
Productivity loss	7 (16.7%)	21 (50.0%)	8 (19.0%)	3 (7.1%)	3 (7.1%)
Project delays	5 (11.9%)	22 (52.4%)	11 (26.2%)	2 (4.8%)	2 (4.8%)
Cost overruns	7 (16.7%)	20 (47.6%)	12 (28.6%)	1 (2.4%)	2 (4.8%)
Product quality problems	13 (31.0%)	17 (40.5%)	10 (23.8%)	2 (4.8%)	0 (0.0%)
Repeating mistakes	20 (47.6%)	19 (45.2%)	2 (4.8%)	1 (2.4%)	0 (0.0%)
Opportunities lost	6 (14.6%)	8 (19.5%)	11 (26.8%)	13 (32.6%)	3 (7.3%)
Project failure	3 (7.0%)	12 (27.9%)	12 (27.9%)	14 (32.6%)	2 (4.7%)
Customer dissatisfaction	10 (24.4%)	13 (31.7%)	11 (26.8%)	5 (12.2%)	2 (4.9%)

Appendix B: Lessons Learnt in Requirements Engineering: A Survey

This appendix includes the survey we used at REFSQ'13 to elicit lessons from practice.

Demographic Information

Type of organization you have worked in substantially (please choose one or more):

- Industry
- Governmental organization
- Academic institution
- Other (please specify)

Key roles played in your career (please choose one or more):

- Manager
- Requirements Analyst/Engineer
- Business Analyst
- Developer
- Architect
- Researcher
- Consultant
- Teacher
- Other (please specify)

Number of years of work experience:

- None
- 1-4 years
- 5-10 years
- 11-15 years
- 16 + years

Survey Questions

1. Please type your lesson in the following box:
2. Lesson repeatability:

   Is this lesson ”repeatable”? If yes, please justify this, giving the properties of this lesson which makes it repeatable and the context in which it is repeatable (including any contexts in which you think it is *not* repeatable).
3. Rationale:

   Why do you think this lesson is important?
4. Source of the lesson:
   - Industrial experience
   - Case study
   - Survey
   - Controlled experiment
   - Other
5. Target object of the lesson:

   The ’target object’ of a lesson is the ”thing” the lesson is about.
   - Technique/method

- Tool
- Process
- Product
- Policy
- Other

6. Negative lesson: A lesson from an unsuccessful experience that requires different behaviour in a similar situation in the future to avoid the observed results.

   Positive lesson: A lesson from a successful experience that encourages the same/similar behaviour in a similar situation in the future to achieve the same/similar observed results.

   Lesson type:

   - Positive
   - Negative

7. Application domain from which the lesson emerged:

   E.g.: banking, healthcare, electronics, etc.

8. Project size (if applicable):

   Any information indicating the size of the project from which the lesson was derived (e.g., number of people, number of LOC or function-points, person-years, etc.).

9. RE Activity(s) (choose one or more):

   One or more RE activities to which the lesson applies.

   - Elicitation
   - Analysis
   - Prioritisation

- Negotiation
- Specification
- Documentation
- Modeling
- Validation
- Other:

10. RE Practice(s) (if applicable) to which the lesson is related:

    An RE practice is a specific way for achieving a particular software development goal and it may be applied to one or more subprocesses (e.g., prototyping, using checklists, use case modelling, prioritisation via voting, tracing using a requirements tool, win-win model of negotiation, elicitation via interviews, using a prioritisation framework, etc.)

11. Software Process (choose one or more if applicable):

    The software process to which the lesson may apply.

    - Agile
    - Waterfall
    - Iterative
    - Spiral
    - Code-and-Fixe
    - Rapid prototyping
    - Other:

12. Impact:

    What is the anticipated impact of the lesson on such items as: product quality, process efficiency, project costs, effort needed in carrying out tasks, etc. Be as specific as you can.

13. E-mail address (optional):

    If you would like to receive the survey report, please enter your e-mail address.

14. If you would not like us to contact you for clarification purposes, please uncheck this box.

    - Yes, contact me for clarification purposes

15. If you would like to add another lesson, please check the following box:

    - Yes, I have another lesson to share!

Appendix C: Results of the Empirical Study: Elicited Lessons

This appendix includes the result of the systematic literature review we conducted to elicit lessons from the RE literature. In Chapter 6 we listed 3 from the literature of year 2011 and 3 from 2012. The complete list of lessons (209 lessons) are listed below.

It is important to note that *most* of the values in the lesson (mainly explicit lessons), rationale, and impact attributes have been included word-by-word from their sources, which are cited at the beginning of each lesson. Some values for the application domain and project size attributes have also been included exactly as mentioned in their sources.

Elicited Lessons:

Lesson ID: LL01_11 [Gonzales and Leroy, 2011]
Journal: EMSE
Conference: N/A
Workshop: N/A
Year: 2011
Lesson: Augment existing elicitation processes with the Appreciative Inquiry to elicit requirements.
Source: Case study, controlled experiment, quasi-experiment
Rationale: For decades and still today, software development projects have failed because they do not meet the needs of users, are over-budget, and abandoned. To help address this problem, the user requirements elicitation process was modified based on principles of Appreciative Inquiry.
Impact: Appreciative Inquiry demonstrated benefits to the requirements gathered by increasing the number of unique requirements as well as identifying more quality-based (non-functional)

and forward-looking requirements. It worked best when there was time for participants to reflect on the thought-provoking questions and when the facilitator was knowledgeable of the subject-matter and had extra time to extract and translate the requirements. The participants (end-users and developers) expressed improved project understanding. End- users participated consistently with immediate buy-in and enthusiasm, especially those users who were technically-inhibited. Appreciative Inquiry can augment existing methods by presenting a positive and future aspect for a proposed system resulting in improved user requirements.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Teacher online community, retail websites, campus website

Project size: Controlled experiment: men (10) and women (15)

RE Practice: N/A

RE Phase: Elicitation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: California State Polytechnic

Repeatability: 4 studies

Lesson ID: LL02_11 [Sim and Alspaugh, 2011]

Journal: EMSE

Conference: N/A

Workshop: N/A

Year: 2011

Lesson: Use humanities analysis techniques to analyse data elicited using the War Stories approach.

Source: Qualitative study

Rationale: When analyzing data elicited using the war stories technique, previously introduced by Lutters and Seaman, we encountered unexpected challenges in applying standard qualitative analysis techniques. After reviewing the literature on stories and storytelling, we realized that a richer analysis would be possible if we accorded more respect to the datas structure and nature as stories, rather than treating our participants utterances simply as textual data.

Impact: Allowed us to preserve more of the story structure.

Target Object: Technique/method

Type: Positive

Expression: Explicit

Application Domain: Practitioners from industry and academia

Project size: A total of 34 requirements engineers agreed to participate in the study. Participants came from the following three groups: 14 attendees at the 2006 International Requirements Engineering Conference (RE06), 15 practitioners at Intuit, Inc. in San Diego, and 5 practitioners from elsewhere in Southern California.

RE Practice: Using War Stories approach

RE Phase: Elicitation, analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time qualitative study results

Lesson ID: LL03_11 [Dieste and Juristo, 2011]

Journal: TSE

Conference: N/A

Workshop: N/A

Year: 2011

Lesson: Use unstructured interviews to elicit requirements.

Source: Systematic review

Rationale: Some experiences suggest that elicitation techniques are more or less equivalent for simple and well-defined problems. For real problems though, a number of studies suggest that elicitation techniques are not interchangeable, and there are far-reaching differences with regard to what type of knowledge each technique can uncover. Other aspects, like quantity of information or elicitation efficiency, are features that might distinguish one elicitation technique from another.

Impact: Unstructured interviews (although it is reasonable to assume that the same applies to structured interviews), are equally as or more effective than introspective techniques (such as protocol analysis) and sorting techniques. Unstructured interviews output more complete information than retrospective techniques (such as protocol analysis) sorting techniques and laddering. Unstructured interviews (although it is reasonable to assume that the same applies to structured interviews), are less efficient than sorting techniques and Laddering, but as efficient as introspective techniques (such as protocol analysis).

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: General

Project size: Selected and extracted data from 26 of those publications. The selected publications contain 30 empirical studies. These studies were designed to test 43 elicitation techniques and 50 different response variables. We got 100 separate results from the experiments. The aggregation generated 17 pieces of knowledge about the interviewing, laddering, sorting, and protocol analysis elicitation techniques.

RE Practice: Interviews

RE Phase: Elicitation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: N/A

<u>***Lesson ID:***</u> LL04_11 [Dieste and Juristo, 2011]

Journal: TSE

Conference: N/A

Workshop: N/A

Year: 2011

Lesson: Do not use introspective techniques (such as protocol analysis) for eliciting requirements.

Source: Systematic review

Rationale: Some experiences suggest that elicitation techniques are more or less equivalent for simple and well-defined problems. For real problems though, a number of studies suggest that elicitation techniques are not interchangeable, and there are far-reaching differences with regard to what type of knowledge each technique can uncover. Other aspects, like quantity of information or elicitation efficiency, are features that might distinguish one elicitation technique from another.

Impact: They are the worst of all tested techniques in all dimension (effectiveness, efficiancy, completeness) and are outperformed by unstructured interviews (although it is reasonable to assume that the same applies to structured interviews) and some sorting techniques and laddering.

Target Object: Technique/method

Type: Negative

Expression: Implicit

Application Domain: General

Project size: Selected and extracted data from 26 of those publications. The selected publications contain 30 empirical studies. These studies were designed to test 43 elicitation techniques and 50 different response variables. We got 100 separate results from the experiments. The aggregation generated 17 pieces of knowledge about the interviewing, laddering, sorting, and protocol analysis elicitation techniques.

RE Practice: Protocol analysis

RE Phase: Elicitation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: N/A

<u>Lesson ID:</u> LL05_11 [Kamalrudin et al., 2011]

Journal: N/A

Conference: ICSE

Workshop: N/A

Year: 2011

Lesson: Use Kamalrudin's et al. tool to check for requirements consistency, correctness and completeness between requirements representations.

Source: End user study

Rationale: Requirements specifications need to be checked against the 3Cs - Consistency, Completeness and Correctness in order to achieve high quality. This is especially difficult when working with both natural language requirements and associated semi-formal modelling representations.

Impact: Low-level inconsistency problems can be identified such as natural language phrases without matching semi-formal model elements and meta-model constraint violations of the extracted model. Higher-level problems, including inconsistency, incompleteness and incorrectness can be identified by comparing the semi-formal model to the Essential interaction pattern and to the best practice examples of EUC interaction pattern templates. A visual differencing technique highlights differences between the pattern template and EUC model. Modifications to EUC, abstract interaction and natural language requirements representations are also supported with consistency management support between the different representations. most participants finding our tool to be useful for improving quality and managing consistency of requirements.

Target Object: Tool

Type: Positive

Expression: Implicit

Application Domain: University

Project size: Participants in the study were 11 software engineering post- graduate students, several of whom had previously worked in industry as developers and/or requirements engineers.

RE Practice: Semi-automated checking

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time end user study results ***Lesson ID:*** LL06_11 [Borges et al., 2011]

Journal: N/A

Conference: ICSE

Workshop: N/A

Year: 2011

Lesson: Use Borges' et al. tool for adapting and evolving software requirements models that uses model checking and machine learning techniques for verifying properties and evolving model descriptions.

Source: Case study

Rationale: The specification process still involves considerable human-centered efforts, which are notably error and inconsistency-prone. Errors may occur in different phases of the process, from the requirements specification through to the design and actual specification and implementation of the software models. When specification errors or inconsistencies are found, current tools provide limited information about what should be rectified in the system being developed.

Impact: Unifying, in a robust and sound process, the verification of properties of a given model, the evolution of this model according to such properties, and the possibility of adapting this knowledge model from the observation of an actual system. First, the framework is capable of coping with errors in the specification process so that performance degrades gracefully. Second, the framework can also be used to re-engineer a model from examples only, when an initial model is not available.

Target Object: Tool

Type: Positive

Expression: Implicit

Application Domain: Pump system

Project size: N/A

RE Practice: N/A

RE Phase: Verification

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time evaluating case study results

<u>Lesson ID:</u> LL07_11 [Li et al., 2011]

Journal: N/A

Conference: ICSE

Workshop: N/A

Year: 2011

Lesson: Use Li's et al. domain specific requirements model to elicit requirements in scientific computing projects.

Source: Exploratory experiment

Rationale: There is a need for methodologies, which capture requirements for scientific computing projects, because traditional requirements engineering methodologies are difficult to apply in this domain.

Impact: They found this approach of modeling requirements very suitable for their domain as they can easily understand the terminology and it is very easy to identify the relationship between a requirement and how it evolves from the theoretical model. This domain specific requirements model allows them to capture the rationale behind a given requirement and helps them to discover open issues. Firstly, a requirements model is at a higher level of abstraction than a textual requirements specification. Models can support traceability, reusability and extensibility. Therefore, it can better deal with complexity and change. Secondly, a domain specific model provides abstractions and notations targeted at the specific domain. This makes modeling less complicated and reduces the learning effort for scientists. The model facilitates the communication across the domain boundary between the scientific computing domain and the software engineering domain. It promotes software engineering in scientific computing projects.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: University

Project size: Five developers from four different research projects participated in the experiment. Three of them are Ph.D students, while two have already received their Ph.D and are continuing their work as scientific researchers.

RE Practice: Modeling

RE Phase: Elicitation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time evaluating experiment results

<u>Lesson ID:</u> LL08_11 [Kof and Penzenstadler, 2011]

Journal: N/A

Conference: N/A

Workshop: REEW

Year: 2011

Lesson: Use a NLP approach with a CASE tool to generate a requirement model from a natural language requirements document.

Source: Case study

Rationale: Despite the fact that documents are mostly written in natural language, natural language processing (NLP) is barely used in industrial requirements engineering. Trade-offs of the existing NLP approaches hamper their broad usage: existing approaches either introduce a restricted language and, correspondingly, are able to process solely this restricted language, or, in the case of a non-restricted language, they cannot adapt to different writing styles, often co-existent in a single requirements document.

Impact: Efficient analysis and early construction: Given a textual requirements document, it helps to explore the document and to construct a system model. Automated tracing: When constructing the model, it maintains explicit traces between the model and the textual document. Early verification of requirements documents: If the model description in the document is incomplete, it makes this incompleteness apparent, by creating an incomplete system model.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Steam boiler, auto pilot, bay area rapid transit and instrument cluster

Project size: N/A

RE Practice: Modeling

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time evaluating case study results

Lesson ID: LL09_11 [Boutkova, 2011]

Journal: N/A

Conference: N/A

Workshop: REEW

Year: 2011

Lesson: Use the Feature Based Variability Management approach with DOORS to document and manage variability.

Source: Industrial experience

Rationale: A problem for the application of the feature modelling approach in requirements specification is a lack of tool support. An analysis of the existing tools shows some deficiencies that render these tools unsuitable for the needs of Daimler passenger car development (Daimler PCD).

Impact: The new approach allows creating of a new specification for a product variant within 15-30 minutes. Therefore the novel approach radically reduces the time effort for creating variant specifications. (1) The reduction of the time effort for creating the variant specifications.(2) The know-how about reasons for the variability.(3) Independence from a specific requirements management tool.

Target Object: Tool

Type: Positive

Expression: Implicit

Application Domain: Passenger car development

Project size: There are about 400 technical requirements. In the initial specification all requirements are mapped to the next three car models.

RE Practice: Modeling

RE Phase: Specification

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Daimler PCD

Repeatability: First time industrial experience

Lesson ID: LL010_11 [van Tuijl et al., 2011]

Journal: N/A

Conference: N/A

Workshop: REEW

Year: 2011

Lesson: Use the Bubblesort prioritization technique instead of the Analytical Hierarchy Processing (AHP) technique.

Source: Experiment, questionnaire

Rationale: Software vendors often face the difficult task to deal with large amounts of requirements that enter the company every day. When dealing with this vast amount of requirements, the notion of deciding which requirements will be addressed first, and which will be addressed later, is an important decision. To support software development teams in decision-making, different prioritization techniques are discussed in previous literature.

Impact: Bubblesort outpaced AHP on all aspects in terms of usability, time consumption and perceived reliability.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: University

Project size: Twelve students. We created a list of twenty requirements for a widely used web-based software package, called Google Maps.

RE Practice: N/A

RE Phase: Prioritisation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Utrecht University

Repeatability: First time experiment results

Lesson ID: LL011_11 [Helferich and Mautsch, 2011]

Journal: N/A

Conference: N/A

Workshop: RePriCo

Year: 2011

Lesson: Use the Quality Function Deployment Product Portfolio Planning (QFD-PPP) method to prioritize requirements and and customer segments.

Source: Case study

Rationale: Where different customers requirements are relatively similar but diverse when analyzed in further detail, offering a number of distinct variants of the software package can be beneficial. A large number of software development techniques have been developed to cope with this complexity. Still, every variant developed results in additional effort.

Impact: Enables software companies to base decisions on core and variable assets on actual customer needs and therefore assists in controlling product (and especially architectural) complexity. A segment formed using traditional marketing methods was too broad and that a value-based analysis helped defining optional modules that can be offered to the customers.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Datacenter operations

Project size: 71 customer requirements, which were divided into 12 requirement categories.

RE Practice: N/A

RE Phase: Prioritisation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

Lesson ID: LL012_11 [Loconsole et al., 2011]

Journal: N/A

Conference: N/A

Workshop: RePriCo

Year: 2011

Lesson: Use the MAAD (Method for Agile, Automated and Distributed prioritization) an algorithmic prioritization method for requirements prioritisation.

Source: Case study

Rationale: Prioritisation can be very time consuming, especially when dealing with large amounts of information. During the prioritisation process both short-term and long-term effects of information items such as requirements, tasks and requests must be evaluated. This is particularly complicated if the items affect each other. Another issue when performing the prioritization is that the knowledge required to prioritize usually is distributed among several persons, for instance employees and customers.

Impact: MAAD prioritisation was easier- to-use, less time-consuming, more accurate, more scalable and the prioritization method most suitable for Flygprestanda compared to Wiegers method and the prioritization method used at the company.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Aviation

Project size: Currently about 7000 to 10000 requirements and change requests are handled each year, distributed over a few hundred software and service projects. The participants were chosen in order to have different project roles represented: two customers, one manager and three developers.

RE Practice: N/A

RE Phase: Prioritisation

Software Process: Agile in the sense of iterative work, relatively short release cycles and a clear focus on producing code rather than extensive documentation

Project Date: N/A

Recording Date: N/A

Organisation Name: Flygprestanda AB

Repeatability: First time evaluating case study results

Lesson ID: LL013_11 [El-Sharkawy and Schmid, 2011]

Journal: N/A

Conference: REFSQ

Workshop: N/A

Year: 2011

Lesson: Use Sharkawy's et al. heuristic approach to support creativity in requirements engineering.

Source: Controlled experiment

Rationale: The core problem in product innovation is creativity. It provides the basis for innovation and all other steps in innovation build on this. Thus, in order to improve product innovation, we must support people in being more creative.

Impact: We did show effectively that not only our approach does work, it is also more effective than an approach relying on random triggers.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: University

Project size: 11 people were assigned to group 1 and 9 people to group 2.

RE Practice: N/A

RE Phase: Elicitation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time evaluating experiment results

<u>Lesson ID:</u> LL014_11 [Mahaux et al., 2011]

Journal: N/A

Conference: REFSQ

Workshop: N/A

Year: 2011

Lesson: For stakeholder analysis, one could augment the checklists with environment related roles.

Source: Exploratory study

Rationale: Sustainability has become one of the grand challenges of our civilisation. Because of their pervasiveness, the way we design, and consequently use, software-intensive systems has a significant impact on sustainability. This gives software requirements engineering an important role to play in society.

Impact: This enabled us to imagine desired and undesired customer-oriented scenarios focusing on the sustainability aspect.

Target Object: Technique/method

Type: Positive

Expression: Explicit

Application Domain: Event management

Project size: Seven employees Belgian company

RE Practice: N/A

RE Phase: Stakeholder analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Yellow Events

Repeatability: First time exploratory results

Lesson ID: LL015_11 [Mahaux et al., 2011]

Journal: N/A

Conference: REFSQ

Workshop: N/A

Year: 2011

Lesson: For context modeling, one could chose a model that will enable to think about the system as a whole in its environment-at-large, and show the physical flows, not only the information flows, coming to or from the product.

Source: Exploratory study

Rationale: Sustainability has become one of the grand challenges of our civilisation. Because of their pervasiveness, the way we design, and consequently use, software-intensive systems has a significant impact on sustainability. This gives software requirements engineering an important role to play in society.

Impact: It helped a lot in understanding the problem, including the important environmental dimension. This model shifted the focus from the product to its wider environment and helped the authors consider where the impacts on sustainability might be.

Target Object: Technique/method

Type: Positive

Expression: Explicit

Application Domain: Event management

Project size: Seven employees Belgian company

RE Practice: Modeling

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Yellow Events

Repeatability: First time exploratory results

Lesson ID: LL016_11 [Mahaux et al., 2011]

Journal: N/A

Conference: REFSQ

Workshop: N/A

Year: 2011

Lesson: For goal modeling, one could prepare and instantiate a generic sustainability goal model.

Source: Exploratory study

Rationale: Sustainability has become one of the grand challenges of our civilisation. Because of their pervasiveness, the way we design, and consequently use, software-intensive systems has a significant impact on sustainability. This gives software requirements engineering an important role to play in society.

Impact: This helped defining more precisely what sustainability meant for Yellow, to complete the list of impacts and possible means of action. Some new insights were found, like the possible contribution to compensation programs, leading to new requirements.

Target Object: Technique/method

Type: Positive

Expression: Explicit

Application Domain: Event management

Project size: Seven employees Belgian company

RE Practice: Modeling

RE Phase: Elicitation, analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Yellow Events

Repeatability: First time exploratory results

Lesson ID: LL017_11 [Mahaux et al., 2011]

Journal: N/A

Conference: REFSQ

Workshop: N/A

Year: 2011

Lesson: For use case analysis, one could investigate risks and mitigations through misuse cases directed at sustainability threats.

Source: Exploratory study

Rationale: Sustainability has become one of the grand challenges of our civilisation. Because of their pervasiveness, the way we design, and consequently use, software-intensive systems has a significant impact on sustainability. This gives software requirements engineering an important role to play in society.

Impact: Consists of various functional modules and allowed to phase the project. the question of the possible ways to mitigate the identified harm (typically a new detailed use case or a modification to an existing one) brought the project one step further towards the design of a solution.

Target Object: Technique/method

Type: Positive

Expression: Explicit

Application Domain: Event management

Project size: Seven employees Belgian company

RE Practice: Using use cases

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Yellow Events

Repeatability: First time exploratory results

Lesson ID: LL018_11 [Adam, 2011]

Journal: N/A

Conference: REFSQ

Workshop: N/A

Year: 2011

Lesson: Use the 'match early approach' only in contexts in which people stating requirements and people mapping requirements to service infrastructures are distributed spatially and temporally.

Source: Industrial case study

Rationale: Achieving a tight fit between requirements and reusable assets is not the usual case in practice, even if especially SOA makes such promises. The very early consideration of existing services and their alignment with requirements have therefore been recommended by several references, as otherwise the fit will rather depend on luck.

Impact: The match early approach was an efficient means for performing elicitation workshops. We expect this observation to be the main advantage of match early, as requirements elicitation is typically not integrated with reuse infrastructure matching in todays practice (unfortunately, this setting could not be constructed in a controlled experiment).

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Public safety project

Project size: The platform functionality included around 170 functions and was encapsulated in 17 CSD.

RE Practice: N/A

RE Phase: Elicitation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: German Police and Fire Station

Repeatability: First time case study results. The results are not supported by results of the controlled experiment.

Lesson ID: LL019_11 [Gervasi and Zowghi, 2011]

Journal: N/A

Conference: REFSQ

Workshop: N/A

Year: 2011

Lesson: Use machine learning techniques to mine requirements traces.

Source: Experiment

Rationale: While most research is concerned with ways to reduce the effort needed to establish and maintain traceability links, a different question can also be asked: how is it possible to harness the vast amount of implicit (and tacit) knowledge embedded in already-established links? Is there something to be learned about a specific problem or domain, or about the humans who establish traces, by studying such traces?

Impact: The additional knowledge gained could be used to help familiarize with an unknown domain, to shed some light on refinement decisions, to understand linking policies, or in the end to obtain a more accurate semi-automatic linking of new or changed requirements based on previous history.

Target Object: Tool

Type: Positive

Expression: Implicit

Application Domain: Publicly-available dataset of requirements with traceability information, originally based on the CM-1 project by the NASA Metrics Data Program

Project size: 235 high-level SRS (software requirements specification) which are refined to 220 low-level SDS (software design specification) for the same DPU (data processing unit); 361 manually-verified links relate the two sets and, so to say, tell the story of the refinement

RE Practice: Tracing

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: NASA

Repeatability: First time case study results.

Lesson ID: LL020_11 [Markov et al., 2011]

Journal: N/A

Conference: REFSQ

Workshop: N/A

Year: 2011

Lesson: Trace features to product drivers and business goals.

Source: Industrial case study

Rationale: Misbalance between technology-driven and market-driven requirements.

Impact: Ensure effectiveness of requirements management.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Development, maintenance, and enhancement of a software platform providing core assets to developers of other organizations.

Project size: From project initiation to project completion, the improvement effort took almost two years and was supported by a team of requirements engineering experts.

RE Practice: Tracing

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Siemens

Repeatability: First time industrial case study results

Lesson ID: LL021_11 [Markov et al., 2011]

Journal: N/A

Conference: REFSQ

Workshop: N/A

Year: 2011

Lesson: Apply Workflow driven Feature Model with sellable units and sufficient granularity.

Source: Industrial case study

Rationale: Missing requirements prioritisation process and constant requirements overload.

Impact: Efficient requirements elicitation and negotiation, ensure common product view.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Development, maintenance, and enhancement of a software platform providing core assets to developers of other organizations.

Project size: From project initiation to project completion, the improvement effort took almost two years and was supported by a team of requirements engineering experts.

RE Practice: N/A

RE Phase: Specification

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Siemens

Repeatability: First time industrial case study results

Lesson ID: LL022_11 [Markov et al., 2011]

Journal: N/A

Conference: REFSQ

Workshop: N/A

Year: 2011

Lesson: Simplify tracing by specifying hierarchical relationships between requirement artifacts.

Source: Industrial case study

Rationale: Insufficient traceability.

Impact: Less effort for requirement authors by simplified structure, avoidance of tracing errors.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Development, maintenance, and enhancement of a software platform providing core assets to developers of other organizations.

Project size: From project initiation to project completion, the improvement effort took almost two years and was supported by a team of requirements engineering experts.

RE Practice: Tracing

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Siemens

Repeatability: First time industrial case study results

Lesson ID: LL023_11 [Markov et al., 2011]

Journal: N/A

Conference: REFSQ

Workshop: N/A

Year: 2011

Lesson: Use Feature Model to Family Model mapping. Ensure clear hierarchy of requirement objects.

Source: Industrial case study

Rationale: Intransparent mapping between problem and solution space.

Impact: Early impact analysis and estimations, support of project management.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Development, maintenance, and enhancement of a software platform providing core assets to developers of other organizations.

Project size: From project initiation to project completion, the improvement effort took almost two years and was supported by a team of requirements engineering experts.

RE Practice: N/A

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Siemens

Repeatability: First time industrial case study results

Lesson ID: LL024_11 [Markov et al., 2011]

Journal: N/A

Conference: REFSQ

Workshop: N/A

Year: 2011

Lesson: Use optimized review concept, (review of solution specs at feature completion).

Source: Industrial case study

Rationale: Inefficient review of detailed specs upfront.

Impact: Saving of review effort by consideration of iterative changes to specifications.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Development, maintenance, and enhancement of a software platform providing core assets to developers of other organizations.

Project size: From project initiation to project completion, the improvement effort took almost two years and was supported by a team of requirements engineering experts.

RE Practice: N/A

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Siemens

Repeatability: First time industrial case study results

Lesson ID: LL025_11 [Markov et al., 2011]

Journal: N/A

Conference: REFSQ

Workshop: N/A

Year: 2011

Lesson: Use MVP, priority based iterative, just-in-time preparation of feature specs.

Source: Industrial case study

Rationale: Waterfall approach, many handoffs, work in progress, missing feedback loops.

Impact: Avoidance of waste, flexibility to react on changes, risk reduction.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Development, maintenance, and enhancement of a software platform providing core assets to developers of other organizations.

Project size: From project initiation to project completion, the improvement effort took almost two years and was supported by a team of requirements engineering experts.

RE Practice: N/A

RE Phase: Prioritisation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Siemens

Repeatability: First time industrial case study results

<u>Lesson ID:</u> LL026_11 [Markov et al., 2011]

Journal: N/A

Conference: REFSQ

Workshop: N/A

Year: 2011

Lesson: Administer single requirement artifacts (CM for RE objects).

Source: Industrial case study

Rationale: Specification based RE, possibility of inconsistencies across product versions.

Impact: Version and change management for requirement artifacts, enabling shorter release cycles.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Development, maintenance, and enhancement of a software platform providing core assets to developers of other organizations.

Project size: From project initiation to project completion, the improvement effort took almost two years and was supported by a team of requirements engineering experts.

RE Practice: N/A

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Siemens

Repeatability: First time industrial case study results

<u>Lesson ID:</u> LL027_11 [Markov et al., 2011]

Journal: N/A

Conference: REFSQ

Workshop: N/A

Year: 2011

Lesson: Model feature variability upfront.

Source: Industrial case study

Rationale: No proactive variant management and no support for reuse.

Impact: Easy generation of variants, support of R&D platform strategy.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Development, maintenance, and enhancement of a software platform providing core assets to developers of other organizations.

Project size: From project initiation to project completion, the improvement effort took almost two years and was supported by a team of requirements engineering experts.

RE Practice: Modeling

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Siemens

Repeatability: First time industrial case study results

Lesson ID: LL028_11 [Markov et al., 2011]

Journal: N/A

Conference: REFSQ

Workshop: N/A

Year: 2011

Lesson: Minimize separation of phases and work streams.

Source: Industrial case study

Rationale: N/A

Impact: Intermediate results may not be accepted between teams.

Target Object: Technique/method

Type: Negative

Expression: Implicit

Application Domain: Development, maintenance, and enhancement of a software platform providing core assets to developers of other organizations.

Project size: From project initiation to project completion, the improvement effort took almost two years and was supported by a team of requirements engineering experts.

RE Practice: Modeling

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Siemens

Repeatability: First time industrial case study results

Lesson ID: LL029_11 [Knauss et al., 2011]

Journal: N/A

Conference: REFSQ

Workshop: N/A

Year: 2011

Lesson: Use Bayesian classifier to elicit and analyse security requirements.

Source: Experiment

Rationale: Ignoring security issues early in a project is a major source of recurring security problems in practice. Identifying security-relevant requirements is labour- intensive and error-prone. Security may be neglected in order to finish on time and in budget.

Impact: This can increase security awareness within the software development process. The approach succeeds in assisting requirements engineers in their task of identifying security-relevant requirements, in that it reliably identifies the majority of the security-relevant requirements (recall ¿ 0.9) with only few false positives (precision ¿ 0.8) in software evolution scenarios. Our evaluation of different training strategies shows that the classifier can quickly be adopted to a new domain when no previous versions of requirements specifications are available for training. This could be done by a security expert during a first interview. we achieved very good results in cases where the classifier is applied to the requirements from the same source as it was trained with. We also observed poor results in cases where the classifier was applied to a different requirements specification than the one it was trained with.

Target Object: Tool

Type: Positive

Expression: Implicit

Application Domain: The Common Electronic Purse Specification (ePurse), the Customer Premises Network specification (CPN), and the Global Platform Specification (GP)

Project size: Three industrial requirements documents for evaluation; Common Electronic Purse (ePurse): 124 requirements and 83 security relevant requirements. Customer Premises Network (CPN): 210 requirements and 41 security relevant requirements. Global Platform Spec. (GP): 176 requirements and 63 security relevant requirements.

RE Practice: N/A

RE Phase: Elicitation, analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time experiment results

Lesson ID: LL030_11 [Veerappa and Letier, 2011]

Journal: N/A

Conference: REFSQ

Workshop: N/A

Year: 2011

Lesson: Use cluster analysis techniques used in the area of market segmentation for identifying relevant groups of stakeholders to be used for requirements decision making.

Source: Pilot case study

Rationale: Novel web-based requirements elicitation tools offer the possibility to collect requirements preferences from large number of stakeholders. Such tools have the potential to provide useful data for requirements prioritisation and selection. However, existing requirements prioritisation and selection techniques do not work in this context because they assume requirements ratings from a small number of stakeholders groups, rather than from a large number of individuals. They also assume that the relevant groups of stakeholders have been identified a priori, and that all stakeholders within a group have the same preferences.

Impact: There is an improvement in overall closeness of the ratings used to make decisions when using cluster analysis.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: University

Project size: Carried out a survey at UCL asking 50 potential stakeholders to rate 5 requirements R1, R2, R3, R4 and R5 for an online calendar on a 10 point scale. We obtained responses from 47 stakeholders

RE Practice: N/A

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: UCL

Repeatability: First time pilot case study results

Lesson ID: LL031_11 [Gacitua et al., 2011]

Journal: RE

Conference: N/A

Workshop: N/A

Year: 2011

Lesson: Use relevance-based abstraction identification (RAI) technique(RAI-1 tool) to automatically identify abstraction.

Source: Experiment

Rationale: When first approaching an unfamiliar domain or requirements document, it is often useful to get a quick grasp of what the essential concepts and entities in the domain are. This process is called abstraction identification, where the word abstraction refers to an entity or concept that has a particular significance in the domain. Abstraction identification has been proposed and evaluated as a useful technique in requirements engineering (RE). Identifying abstractions from such (often large) documents imposes a high cognitive load on the requirements engineer. Attention lapses, for example, can result in important abstractions being overlooked.

To support RE, therefore, . . . the desire is for a clerical tool that helps with the tedious, error-prone steps of what a human elicitor does . . .

Impact: In the context of the RFID book evaluation, there is a clear performance advantage to RAI-1 over AbstFinder and C-value. Moreover, the effect of the modifications made to RAI that led to RAI-1 have been dramatic. RAI-1 performs much better than RAI-0 in all our metrics; absolute recall and precision, the rate of increase in recall and rate of decline in precision, and in lag.

Target Object: Tool

Type: Positive

Expression: Implicit

Application Domain: Textbook on RFID and its applications

Project size: The book is large enough to present a real problem of information overload. It is 595 pages long and contains 156,028 words so its size serves to simulate the volume of text that a requirements engineer might encounter in a range of domain materials such as standards, manuals or indeed text books. The books analytical index holds 911 entries.

RE Practice: N/A

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time evaluating experiment results

Lesson ID: LL032_11 [Liaskos et al., 2011]

Journal: RE

Conference: N/A

Workshop: N/A

Year: 2011

Lesson: Use Liaskos' et al. extended state-of-the-art goal modeling notation to support the representation of preference (nice- to-have) requirements.

Source: Experiment

Rationale: Current goal-oriented modeling frameworks treat goals as mandatory requirements that must be fulfilled by any proposed solution. In this respect, such frameworks cannot accommodate preference (nice-to-have) requirements that might be posed by stakeholders.

Impact: Our experiments indicate that the task of reasoning about preferences and alternatives allows better under- standing of the connection between the stakeholder attitudes and alternative designs. This makes it particularly useful for exploring alternative designs during early requirements stages, supporting priority elicitation activities by directly showing the implication of certain prioritisations, improving domain understanding and model accuracy, or, potentially, supporting the customization of software systems by connecting the high-level design descriptions obtained through the tool into configurations of variation points in the software itself.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: The health care domain, involving nursing processes, the ATM domain, exploring different behavioral designs for an automated teller machine, and the classic meeting scheduler domain, investigating different ways to schedule meetings

Project size: The nursing domain (24 mandatory elements 11 of which tasks plus 7 quality preferences), an extended meeting scheduling example (53 mandatory elements, 32 tasks, 13 quality preferences) as well as the ATM example (28 mandatory elements, 21 tasks, 4 quality preferences)

RE Practice: Modeling

RE Phase: Prioritisation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time evaluating experiment results

Lesson ID: LL033_11 [Mashkoor and Jacquot, 2011]

Journal: RE

Conference: N/A

Workshop: N/A

Year: 2011

Lesson: Use Event-B for domain engineering.

Source: Industrial experience

Rationale: Well-specified requirements are crucial for good software design; they depend on the understanding of the domain. Thus, domain engineering becomes an essential activity. The possibility to have a formal model of a domain, consistent with the use of formal methods for developing critical software working within it, is an important issue. Safety-critical domains, like transportation, exhibit interesting features, such as high levels of nondeterminism, complex interactions, stringent safety properties, and multifaceted timing attributes. The formal representation of these features is a challenging task. Most customers express their requirements either in natural language or in terms of scenarios. Most of the requirements engineering methodologies are therefore nonformal or semi formal. One of the problems with less formal techniques is that they may be ambiguous, which makes the requirements engineering phase error prone.

Impact: Event-B is an adequate language for domain engineering

Target Object: Tool

Type: Positive

Expression: Explicit

Application Domain: Transportation systems

Project size: Our current domain model contains one abstract machine and seven refinements. In parallel with the machines, two contexts are being refined. The first is the context Net, which models the static properties of the network (its topology, quantities associated to its elements, etc.). The second is the context StartState, which helps to set and prove the INITIALISATION event of the machines.

RE Practice: Modeling

RE Phase: Specification

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time industrial experience

<u>Lesson ID:</u> LL034_11 [Sutcliffe et al., 2011]

Journal: RE

Conference: N/A

Workshop: N/A

Year: 2011

Lesson: Use a mix of designer-led initiative using HCI patterns and user-centred design that responded to user requirements for requirements engineering.

Source: Case study

Rationale: User interfaces (UI) and HCI are direct concerns of requirements engineering rather than a veneer of interactive components that adorn the software system. For example, many requirements for decision-support systems can only be considered in terms of user interaction, while functionalities of the user interface are first-order requirements that serve user goals, e.g., functional requirements to interactively explore information spaces, virtual worlds and social networks.

Impact: The mix of designer-led initiative using HCI patterns and user-centred design that responded to user requirements worked well. HCI influenced the requirements specification process as well as the consequent requirements for the ADVISES project in three ways. First, the scenario-based process facilitated exploration of users requirements and, more importantly, their design realisation. This enabled users to contribute to developing a software specification which would change their work practices. Experience with a similar user-centred development approach was reported by Maiden and Robertson. The second influence was application of functional allocation as a means of refining the functional requirements and user interface architecture. The functional allocation heuristics which proved to be useful in ADVISES could be applied in RE approaches when different system implementations are being considered; for example, when strategic rationale models are created to explore alternative implementation boundaries in strategic dependency i* models. The third influence was application of HCI knowledge in the form of principles and patterns, in particular as solutions to visualisation problems. HCI knowledge was supplied by the first author, supplemented by HCI design patterns literature. HCI requirements based on the gaps problems stimulated the visualisation design as well as pointing towards the patterns solution, e.g., multiple displays enable different users to scan the maps and graphs according to their needs. Linking research questions to display templates supports the users workflow more directly, by providing the necessary information related to the users tasks. Although concurrent multipanel displays may appear to increase complexity, none of our users complained about the displays being too complex.

Target Object: Technique/method

Type: Positive

Expression: Explicit

Application Domain: Visualisation tools to support epidemiological research and public health decision-making

Project size: N/A

RE Practice: N/A

RE Phase: Analysis, specification

Software Process: Iterative

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

<u>Lesson ID:</u> LL035_11 [Sutcliffe et al., 2011]

Journal: RE

Conference: N/A

Workshop: N/A

Year: 2011

Lesson: Use the the combination of storyboards, scenarios and prototypes integrated in a user-centred design cycle for requirements engineering.

Source: Case study

Rationale: N/A

Impact: The key to user engagement. Visualisation of realistic designs enabled the users to critique and contribute ideas in their own terms without understanding software engineering notations. Our experience has been that even simple notations such as use cases present a barrier to understanding; furthermore, abstract models are less meaningful for users.

Target Object: Technique/method

Type: Positive

Expression: Explicit

Application Domain: Visualisation tools to support epidemiological research and public health decision-making

Project size: N/A

RE Practice: Using storyboards, scenarios, prototypes

RE Phase: Analysis, specification

Software Process: Iterative

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

Lesson ID: LL036_11 [Ballejos and Montagna, 2011]

Journal: RE

Conference: N/A

Workshop: N/A

Year: 2011

Lesson: Use Ballehos' et al. integrated model for representing and managing stakeholder-related concepts in the development of an information system.

Source: Case study

Rationale: In the software engineering area, stakeholders play a significant role in requirements elicitation and validation. Moreover, all the project management is integrally affected by stakeholders perspectives and their participation. This effect is strengthened when projects involve several organizations. Thus, a clear and explicit representation of the stakeholders and their attributes is required in order to achieve their effective management. The integration of this representation with other models capturing the knowledge of engineering design processes can be of great utility in software development projects.

Impact: The model helped shaping the planning and implementation of the design process, in whose progress stakeholders were interested and involved and also contributed to the effective project team leadership, organizing stakeholder-related information. Through the stakeholders modeling, a range of significant issues were revealed that would not otherwise have surfaced until implementation of the system and therefore too late to resolve without major wastage of resources. For example, information regarding the most interested and influencing stakeholders can be checked during the process. These stakeholders are the ones that will be really conducting the process, influencing decisions or, perhaps, making them. The use of the model significantly increased the chances to guarantee an appropriate and consistent level of stakeholder information representation in the project, since the particular roles and power types of the specific project were instantiated.

Target Object: Technique/method

Type: Positive

Expression: Explicit

Application Domain: Public health area of an Argentinean province

Project size: N/A

RE Practice: Modeling

RE Phase: Stakeholder analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

Lesson ID: LL037_11 [Pitula and Radhakrishna, 2011]

Journal: RE

Conference: N/A

Workshop: N/A

Year: 2011

Lesson: Elicit requirements by applying Structured Digital Story- telling (SDS) to elicit needs directly from the end-users and apply a conceptual model of experiential culture to interpret these needs and additional constraints arising from the broader social context.

Source: Case study

Rationale: In many developing countries, a growing effort is under- way to provide disadvantaged people in rural areas with access to digital content and services using Information and Communication Technologies (ICT). Such efforts are referred to by the term ICT for Development or ICT4D. Although numerous pilot projects have been attempted over the past decades, few have managed to bring long- term sustained benefits to the people that they target.

Impact: All the participants were able to tell their story and were enthusiastic about doing so. Villagers participated readily and quickly picked up the operation of the application. Once they began talking, they became engaged in telling their story and were not distracted by the mechanics of recording. While in almost all cases they participated in groups, their stories were highly personal and did not show any signs of groupthink. At the same time, the group provided an audience for the teller, making the narration a natural communicative exchange. Our analysis of the stories indicates that they are highly useful in identifying the participants concerns and reveal an abundance of contextual information.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain:Information and Communication Technologies (ICT) such as mobile phones, to assist in reducing disparities in socioeconomic conditions throughout the world. Such efforts have come to be known as ICT for Development or ICT4D

Project size: Altogether 30 stories were collected, 17 on farming and 13 on higher education. These were told by both male and female participants represent- ing a broad age range, from children to the elderly and a cross section of financial situations, from the very poor to those considered well off by local standards.

RE Practice: N/A

RE Phase: Elicitation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

Lesson ID: LL038_11 [Luna et al., 2011]

Journal: RE

Conference: N/A

Workshop: N/A

Year: 2011

Lesson: Use WebSpec, a domain-specific language for specifying the most relevant and characteristic requirements of Web applications.

Source: Case study

Rationale: Web application development is a complex and time-consuming process that involves different stakeholders (ranging from customers to developers); these applications have some unique characteristics like navigational access to information, sophisticated interaction features, etc. However, there have been few proposals to represent those requirements that are specific to Web applications.

Impact: The customer liked the use of mockups and the simulation features of WebSpec as they gave him a clear picture of the understanding of the analyst regarding the requirements. On the other hand, some diagrams were rather complex (specially the list of actions) and thus hard to understand by the customer. He suggested providing a simplified view of the diagram in those cases. In the development team, the most appreciated feature was the test suite derived directly from the diagrams. The test suite was used to asses whether the requirements were correctly implemented during the development cycle and to check that new code did not break existing functionality. The test suite grew quickly, and therefore, the time consumed to run the tests also grew. Finally, in the coding side, mockups and WebSpec diagrams help to implement the requirement using the code derivation features (GWT effect handler) and were appreciated by developers as it automates UI changes.

Target Object: Language, tool

Type: Positive

Expression: Implicit

Application Domain: Web development for the postgraduate area

Project size: The development team is composed of two developers, one analyst and a project manager. The requirements were obtained from one person (the head of the college), thus avoiding any conflict resulting between different stakeholders. The project was divided in sprints (as in most agile approaches) in which we tackle a set of requirements delivering a running application to the customer. In our case, we had six sprints to implement several user stories though here we only show the first three sprints. Each sprint was delivered within 2 weeks, thus gathering quick feedback from the customer.

RE Practice: N/A

RE Phase: Specification

Software Process: Agile

Project Date: N/A

Recording Date: N/A

Organisation Name: College of Medicine in the University of La Plata

Repeatability: First time evaluating case study results

Lesson ID: LL039_11 [Yue et al., 2011]

Journal: RE

Conference: N/A

Workshop: N/A

Year: 2011

Lesson: An ideal approach for transforming requirements into analysis models would have the following characteristics: (1) requirements should be easy to document using the format required by the approach, (2) generated analysis models should be complete (i.e., contain structural and behavioral aspects of a system), (3) the approach should contain the least number of transformation steps as possible (high efficiency), (4) the approach should be automated, and (5) the approach should support traceability management.

Source: Systematic review

Rationale: Model transformation is one of the basic principles of Model Driven Architecture. To build a software system, a sequence of transformations is performed, starting from requirements and ending with implementation. However, requirements are mostly in the form of text, but not a model that can be easily understood by computers; therefore, automated transformations from requirements to analysis models are not easy to achieve.

Impact: N/A

Target Object: Technique/method

Type: Negative

Expression: Implicit

Application Domain: N/A

Project size: The systematic review led to the analysis of 20 primary studies (16 approaches) obtained after a carefully designed pro- cedure for selecting papers published in journals and con- ferences from 1996 to 2008 and Software Engineering textbooks.

RE Practice: Modeling

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: Systematic literature review results

Lesson ID: LL040_11 [Asnar et al., 2011]

Journal: RE

Conference: N/A

Workshop: N/A

Year: 2011

Lesson: Use the Goal-Risk framework for modeling and reasoning about risk during requirements analysis extending the Tropos goal modeling framework.

Source: Case study

Rationale: Risk analysis is traditionally considered a critical activity for the whole software systems lifecycle. Risks are identified by considering technical aspects (e.g., failures of the system, unavailability of services, etc.) and handled by suitable countermeasures through a refined design. This, however, introduces the problem of reconsidering system requirements.

Impact: Positive experiences in communicating GR models to analysts and domain experts. This is an important strength for any requirements analysis technique because it empowers domain experts to under- stand and critique proposed models. Moreover, the learning process for experts to understand and use a GR model takes relatively short period (approximately 23 months). The GR framework supports risk analysis during the very early phases of software development. Consequently, it reduces the risk of requirements revision, and consequently the cost of development. In comparison with KAOS, this framework allows analysts to perform qualitative and quantitative assessment though KAOS provides richer formal semantics using Linear Temporal Logic. Moreover, in comparison with DDP and CORAS the GR framework is more expressive in capturing stakeholders intentions. At last, the GR framework is the only framework that deals with risk and opportunity, since some risks appear because the stakeholders decide to pursue an opportunity. With this feature, one can perform trade-off analysis to decide whether one opportunity is worth to pursue or not.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Banking

Project size: N/A

RE Practice: Using a framework

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

Lesson ID: LL041_11 [Pires et al., 2011]

Journal: RE

Conference: N/A

Workshop: N/A

Year: 2011

Lesson: Use Pires' et al. multi-viewed RE process and ReTool for requirements elicitation, analysis, specification and validation.

Source: Controlled experiment

Rationale: RE activities aim at managing all the requirements-related knowledge. It is common that such knowledge is concretized in a set of artifacts such as use cases, story boards, natural language documents, and business process specifications. These artifacts comprise the so-called Requirements Document. The production of such document is often regarded as one of the most difficult activities in the software development process. The resources applied in building a solid RE process have been shown to pay off. However, studies conducted by renowned IT consulting groups as the Standish, the Gartner, and the Forrester groups have pointed out that a large number of projects still fail to achieve their goals and some of them are even canceled due to requirements-related issues.

Impact: Accomplished analysis indicates the potential applicability of using RETool in the RE process provided that sufficient training is given to the users. The results draw from our experiment indicate that a clear communication channel between the teams involved in a RE process minimizes most of the scope and communication issues, since both parts can express their view about the system scope using a suitable notation. Moreover, the proposed process and tool provide a simple, but useful, traceability scheme. The experiment also shown that the proposed knowledge validation technique was effective to deal with volatility issues, once we were able to track down requirements inconsistencies by using ontologies along with reasoning mechanisms. Finally, the experiment shown that our approach correctly integrates the different views that represent the RE knowledge, once no information was lost during the transformation.

Target Object: Tool

Type: Positive

Expression: Implicit

Application Domain: Health Watcher system (HW) used in a University setting

Project size: Two (2) subjects. The subjects were Computer Science M.Sc. Students in their second year, with basic knowledge in both requirements and software engineering. Second project: 9 subjects

RE Practice: N/A

RE Phase: Elicitation, analysis, specification, validation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Federal University of Rio Grande do Norte, University of Sydney

Repeatability: First time evaluating experiment results

Lesson ID: LL042_11 [Gordon and Breaux, 2011]

Journal: N/A

Conference: N/A

Workshop: RELAW

Year: 2011

Lesson: Use Gordon's et al. heuristics (union, disjoint, minimum) to resolve potential conflicts or differences between requirements.

Source: Case study

Rationale: Increasingly, information systems are becoming distributed and pervasive, enabling organizations to deliver services remotely to individuals and to share and store personal information worldwide. However, system developers face significant challenges in identifying and managing the many laws that govern their services and products. the existing approach - paper-based laws and policies - can no longer scale with technological innovation, and that the regulations must be accessible to policy makers, business analysts, and software developers if an honest expectation of compliance can be preserved.

Impact: We believe these heuristics can be applied to potential conflicts across regulatory requirements to discover a legal landscape consisting of choices that system designers must consider in the context of their products and services, business practices, internal policies, preferences, and risk profiles.

Target Object: Language

Type: Neutral

Expression: Implicit

Application Domain: Regulatory requirements. Regulations from multiple jurisdictions

Project size: 5 laws: We selected the following laws by inviting suggestions from a legal expert with seven years of privacy and security law expertise; additionally, Wisconsin was chosen due to its unique inclusion of biometric data as personal information.

RE Practice: N/A

RE Phase: Specification, analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: AR: Personal Information Protection Act, Arkansas Chapter 4.110; 2005. MA: Security Breaches, Massachusetts Chapter 93H; 2007. MD: Personal Information Protection Act, Maryland Subtitle 14-35; 2008. NV: Security of Personal Information, Nevada Chapter 603A; 2006. WI: Notice of Unauthorized Access to Personal Information, Wisconsin Chapter 134.98; 2006.

Repeatability: First time case study results

Lesson ID: LL043_11 [Uusitalo et al., 2011]

Journal: N/A

Conference: N/A

Workshop: RELAW

Year: 2011

Lesson: Use Easy Approach to Requirements Syntax (EARS) to write/rewrite domain requirements.

Source: Quasi-experiment

Rationale: Requirements of a system are gathered from various stakeholders, but especially in safety critical application domains, such as the nuclear energy domain, public authorities also impose requirements. Major parts of requirements are often written in natural language. Despite being widely applied and a convenient means, natural language requirements have deficiencies such as impreciseness and vagueness. One approach to improve especially existing requirements is to rewrite the requirements applying structured natural language templates such as Easy Approach to Requirements Syntax (EARS).

Impact: The presentation of the EARS- format requirements revealed that although the resulted requirements were a natural interpretation of the original requirement, the results had explicated some assumptions from the original document. The assumptions made by the researcher were not correct, and this was seen to highlight the ambiguity of the sentences of YVL B.1 that required domain knowledge to understand correctly. The application of EARS to RS-RCSU was seen to somewhat clarify the original expressions in the document, but the reaction of the panel was mild, probably because the document was not important to them.The panels opinion was that there was utility in applying EARS into YVL B.1. The method appeared relatively lightweight in terms of effort and learning required as compared to the benefits of its application. The utility of applying EARS into RS-RCSU remained more uncertain. Even though the results attained in improving the content of the document were promising, the panel was unsure whether these results could be generalized to their own requirements specifications. This uncertainty was due to the panel members not being familiar with the structure and content of RS-RCSU.

Target Object: Language

Type: Neutral

Expression: Implicit

Application Domain: Nuclear power plants

Project size: YVL 2.1 Safety classification of systems: 11 processed pages out of a total of 11 pages. YVL B.1 Safety Design of Nuclear Power Plant: 2 processed pages out of a total of 48 pages. Requirements Specification for Rod Control System Upgrade: A Generic Specification for Westinghouse Pressurized Water Reactors (RS-RCSU): 74 processed pages out of a total of 138 pages.

RE Practice: Using templates

RE Phase: Specification

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Finnish public authority guidelines for nuclear safety (YVL)

Repeatability: Preliminary quasi-experiment results

Lesson ID: LL044_11 [Schmidt et al., 2011]

Journal: N/A

Conference: N/A

Workshop: RELAW

Year: 2011

Lesson: Use commitment, privilege, and right (CPR) analysis to identify legally compliant requirements from data use agreements (DUAs).

Source: Case study

Rationale: Security and privacy requirements are often not explicitly stated and are often not easy to elicit. Within the healthcare domain, regulations created pursuant to the U.S. Health Insurance Portability and Accountability Act (HIPAA) specify that a DUA must exist for certain uses and disclosures of protected health information as a limited data set. For compliance reasons, it is important for requirements engineers to ask for and evaluate DUAs, as they are legally binding on the parties.

Impact: Through this grounded theory approach we found that both policy documents and DUAs yielded CPR classifications. In contrast to policy documents, CPRs expressed in DUAs convey the exchange of information between two contractual entities. Within our analysis of four DUAs, we observed that DUAs contain mostly commitments. DUA requirements are legally binding and thus critical to ensuring compliance.

Target Object: Technique/method

Type: Neutral

Expression: Implicit

Application Domain: Healthcare (HIPAA)

Project size: Two policy documents and 4 DUAs

RE Practice: Using CPR analysis

RE Phase: Elicitation, analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

Lesson ID: LL045_11 [Schmidt et al., 2011]

Journal: N/A

Conference: N/A

Workshop: RELAW

Year: 2011

Lesson: Data use agreements (DUAs) are a good source of contractual compliance requirements for a system that manages limited data sets.

Source: Case study

Rationale: N/A

Impact: DUAs are signed by both parties and either create or extend a contractual relationship. Most of the statements express recipient commitments, relating to the requirements of the recipients system.

Target Object: Policy

Type: Positive

Expression: Explicit

Application Domain: Healthcare (HIPAA)

Project size: Two policy documents and 4 DUAs

RE Practice: N/A

RE Phase: Elicitation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

Lesson ID: LL046_11 [Schmidt et al., 2011]

Journal: N/A

Conference: N/A

Workshop: RELAW

Year: 2011

Lesson: Natural language patterns within DUAs can be used to classify statements as CPRs

Source: Case study

Rationale: Even though DUAs are mentioned in HIPAA, most requirements engineers likely would not know to ascertain whether a DUA exists when developing software requirements for healthcare systems. For compliance reasons, it is important for requirements engineers to evaluate DUAs, as they are legally binding on the parties.

Impact: N/A

Target Object: Technique/method

Type: Positive

Expression: Explicit

Application Domain: Healthcare (HIPAA)

Project size: Two policy documents and 4 DUAs

RE Practice: N/A

RE Phase: Elicitation, analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

Lesson ID: LL047_11 [Ohashi et al., 2011]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2011

Lesson: Use Ohashi's 2-step approach and tool for establishing traceability links between requirements definition artifacts and design phase artifacts during the design phase.

Source: Experiment

Rationale: There are thousands of functions and screens in real business applications. It is difficult to establish suitable links from these many elements. In addition, there is a constraint that the workload of setting traceability links should be small. At software quality assurance, artifacts of the software development process must satisfy several conditions. Customer requirements should be traced in the corresponding design, to ensure that no excess specification is designed. Furthermore, it must adapt to the changes in customers requirements.

Impact: In the case in which models were not used, the number of candidate links was 1209, and when models were used the average number of candidate links was reduced to 201.5. In the case in which categories were not used, the number of candidates was 397, and the time needed to create a link was 23 sec/link. In the case of using categories, the average number of candidates was 8.1, and time for setting was 15 sec/link.The number of candidates was reduced to 1/6 by using models, and to 1/49 by using categories. Applying our approach resulted in a sufficiently small enough number of candidate links, and sufficiently reduced time to be effective in real software development projects.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: N/A

Project size: Business Process (most detail): 72. System Function: 397. Screen:526. Slip: 145. Interface: 116. Conceptual Data: 53

RE Practice: Tracing

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time evaluating experiment results

Lesson ID: LL048_11 [Chen et al., 2011]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2011

Lesson: Use Chen's et al. requirements-driven self- tuning method for the survivability assurance of Web systems.

Source: Experiment

Rationale: Running in a highly uncertain and greatly complex environment, Web systems cannot always provide full set of services with optimal quality, especially when work loads are high or subsystem failures are frequent. Hence, it is significant to continuously maintain a high satisfaction level of survivability, hereafter survivability assurance, while relaxing or sacrificing certain quality or functional requirements that are not crucial to the survival of the entire system.

Impact: In each experiment with one of the three kinds of methods, we collected and analyzed the average earned business value and the four qualities per minute in a continuous running of about 77 minutes with different numbers of concur- rent users varying from 10 to 90. The quality and functional reasoning-2 mostly outperforms the static and quality reasoning method, and the quality and functional reasoning-0 mostly outperforms the static method and partially outperforms the quality reasoning method.Adopting the quality and functional reasoning-1 method, the response time is significantly improved compared to static method and slightly improved compared to quality reasoning method; however, the usability is a little reduced but is still acceptable since the usability for static, quality reasoning and quality and functional reasoning-1 method are 10.0, 8.99 and 9.01 respectively. Similarly, availability is improved while cost is increased.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Online shopping system

Project size: Randomly generated goal models whose sizes vary from 25 to 150.

RE Practice: N/A

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time evaluating experiment results

Lesson ID: LL049_11 [Massey et al., 2011]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2011

Lesson: The average graduate- level software engineering student is ill-prepared to write legally compliant software with any confidence and that domain experts are an absolute necessity.

Source: Case study

Rationale: Software engineers regularly build systems that are required to comply with laws and regulations. To this end, software engineers must determine which requirements have met or exceeded their legal obligations and which requirements have not. Requirements that have met or exceeded their legal obligations are legally implementation ready, whereas requirements that have not met or exceeded their legal obligations need further refinement. Research is needed to better understand how to support software engineers in making these determinations.

Impact: N/A

Target Object: People

Type: Negative

Expression: Implicit

Application Domain: Healthcare (HIPAA)

Project size: Each case in our case study received three inputs: (1) a sample legal text; (2) a requirements specification that includes a glossary of terms; and (3) a traceability mapping of the individual requirements to the legal text. The legal text chosen for this study is a HIPAA regulatory section governing technical safeguards. Instead of including all 75 iTrust system requirements, we started with the 15 requirements with legal obligations outlined in HIPAA, then iteratively applied our methodology for evaluating requirements for legal compliance to the other iTrust system requirements. After three iterations, we identified an additional 17 requirements for inclusion in our study. We selected one of our 32 requirements to be used as a part of the tutorial for providing basic training to participants about how software engineers may reason about legal compliance. As a result, we had 31 remaining requirements to use in our case study. The participants in our case study are 32 computer science graduate students who have taken or are taking the graduate-level software engineering course

RE Practice: N/A

RE Phase: Elicitation, specification, analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: North Carolina State University

Repeatability: First time case study results

Lesson ID: LL050_11 [Massey et al., 2011]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2011

Lesson: The metrics used in the legal requirements triage algorithm are useful in identifying LIR requirements.

Source: Case study

Rationale: Software engineers regularly build systems that are required to comply with laws and regulations. To this end, software engineers must determine which requirements have met or exceeded their legal obligations and which requirements have not. Requirements that have met or exceeded their legal obligations are legally implementation ready, whereas requirements that have not met or exceeded their legal obligations need further refinement. Research is needed to better understand how to support software engineers in making these determinations.

Impact: N/A

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Healthcare (HIPAA)

Project size: Each case in our case study received three inputs: (1) a sample legal text; (2) a requirements specification that includes a glossary of terms; and (3) a traceability mapping of the individual requirements to the legal text. The legal text chosen for this study is a HIPAA regulatory section governing technical safeguards. Instead of including all 75 iTrust system requirements, we started with the 15 requirements with legal obligations outlined in HIPAA, then iteratively applied our methodology for evaluating requirements for legal compliance to the other iTrust system requirements. After three iterations, we identified an additional 17 requirements for inclusion in our study. We selected one of our 32 requirements to be used as a part of the tutorial for providing basic training to participants about how software engineers may reason about legal compliance. As a result, we had 31 remaining requirements to use in our case study. The participants in our case study are 32 computer science graduate students who have taken or are taking the graduate-level software engineering course

RE Practice: Using metrics

RE Phase: Elicitation, specification, analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: North Carolina State University

Repeatability: First time case study results

Lesson ID: LL051_11 [Dietsch et al., 2011]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2011

Lesson: Requirements elicitation and formalisation can practically be outsourced to a consulting party.

Source: Industrial case study

Rationale: Natural language safety requirements in industrial standards pose risks for ambiguities which need to be resolved by the system manufacturer in concertation with the certificate authority. This is especially challenging for small and medium-sized enterprises (SME).

Impact: The approach reduces the entry costs while, at the same time, it allows for a gradual introduction of requirements engineering techniques and formal methods to development processes in SME according to the needs of SME.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: A medium-sized company (SME) specialized in developing high-frequency radio-based fire alarm systems (FAS).

Project size: The company employs about 20 people, of which three are dedicated software developers

RE Practice: Using metrics

RE Phase: Elicitation, specification, analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time industrial case study results

Lesson ID: LL052_11 [Dietsch et al., 2011]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2011

Lesson: Consulting parties employ visual narratives as a means to make the benefits of formal methods accessible to stake- holders lacking the corresponding educational background.

Source: Industrial case study

Rationale: Natural language safety requirements in industrial standards pose risks for ambiguities which need to be resolved by the system manufacturer in concertation with the certificate authority. This is especially challenging for small and medium-sized enterprises (SME).

Impact: Especially for the timing relations prevalent in the considered requirements, the drawing and the documentation proved to be very effective. We could convey differences between interpretations with it and later it was also possible to refer to the drawings to compare different scenarios among each other.

Target Object: Tool

Type: Positive

Expression: Implicit

Application Domain: A medium-sized company (SME) specialized in developing high-frequency radio-based fire alarm systems (FAS).

Project size: The company employs about 20 people, of which three are dedicated software developers

RE Practice: Using visual narratives

RE Phase: Elicitation, specification, analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time industrial case study results

<u>Lesson ID:</u> LL053_11 [Sampath et al., 2011]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2011

Lesson: Use the formalism Structured Transition Systems (STS) that facilitates the rapid evolution of specifications.

Source: Case study

Rationale: At this early stage of development, only overall goals of features are understood, and there is a need to discover all possible scenarios of operation. Developing specifications for such features is difficult because there is very little precedent from which one can draw experience. Also, the requirements may depend critically on the technology available for implementation, such as sensors or actuators available in the market, cost of implementation etc. As a result specifications are continually revised and evolve from early conception to final implementation.

Impact: The ability to use analysis results to refine and reinforce parts of the specification by importing analysis results into STS specifications. In practice, this leads to a feedback loop where requirements can be rapidly refined using analysis engines to drive the development of requirements. Our method proved to be quite effective in developing a consistent and unambiguous specification of the feature. We were also able to identify a number of safety-critical corner- case scenarios that had not been identified by the subject- matter experts. These scenarios would have been very difficult to identify by just inspection of the textual specification.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Automotive. novel in-vehicle control features

Project size: The document was around 50 pages long, and the functional specification itself had around 60 requirements, with each having on average 4-5 sub-requirements. There were around 5 bubble-diagrams showing the high- level mode behaviour of the feature.

RE Practice:N/A

RE Phase: Specification, analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time evaluating case study results

Lesson ID: LL054_11 [Ernst et al., 2011]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2011

Lesson: Use the using a "requirements engineering knowledge base (REKB), whose specification is formalized to desirable solutions as the requirements change on top of a reason maintenance system (ATMS).

Source: Case study

Rationale: While most research on design for software focuses on finding some correct solution, this ignores that such a solution is often only correct in a particular, and often short-lived, context.

Impact: Our numbers suggest that the incremental algorithm constitutes a clear improvement on starting from scratch. For example, looking for alternative solutions can be done nearly instantly, allowing stakeholders to use our tool as a workbench for solution identification. While the naive algorithm (adding new changes to the REKB and re-calculating the labels) is not terribly slow, there is a large relative difference we expect to see in larger models as well. The timing results for finding minimal new changes are also nearly-instant, allowing the REKB to support interactive decision-making.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: N/A

Project size: Large, randomly generated requirements models. We start with a 400-node model

RE Practice: Reuse

RE Phase: Specification

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time evaluating case study results

Lesson ID: LL055_11 [Boulila et al., 2011]

Journal: N/A

Conference: N/A

Workshop: MERE

Year: 2011

Lesson: Brainstorming and Storytelling can play a complementary role in requirements elicitation.

Source: Case study

Rationale: Existing requirements elicitation approaches have proven insufficient to record complete, consistent, and correct requirements. Studies conducted have shown that 40% of defects in software projects are due to incorrect recorded requirements. Therefore, some innovative approaches have been developed to deal with the lack of addressing the above- mentioned issues including video-based methods.

Impact: The Storytelling group was more dynamic, showed clearly more emotional signs than the brainstorming group, members showed more interaction with each other, gave more stories than solutions, and more anecdotes than brainstorming. This group produced by far much more content than the first group, which essentially conducted brainstorming activities. The second group produced less content (topics, questions, alternatives, various short stories, etc) and was thinking in terms of solutions described in use cases than stories and anecdotes like in the first group. Quantitatively seen, the first group produced almost three times more requirements than the first group. The first group acted as a ”control group”, the second group acted as a ”treatment group”. Storytelling fulfilled the requirements and provided more elements to be more effective than brainstorming. Indeed, the S-group produced a higher number of use cases covering all of the ticket machine requirements the group members could think of. In addition, more specific details were revealed which were not observed in the results of the brainstorming group. Moreover, the use cases were clearly stated and related issues were solved during the time-frame allocated for the case study.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Ticket machine

Project size: Twenty- five domain experts from various industrial companies academic institutions to collect requirements using a Storytelling technique

RE Practice: Using story-telling

RE Phase: Elicitation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

Lesson ID: LL056_11 [Schneider, 2011]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2011

Lesson: Use Scheider's ConTexter tool for focusing spontaneous feedback to support evolution of the running system.

Source: Case study

Rationale: Modern software systems are rarely built from scratch. They rather evolve over a long period of time while components and subsystems are developed independently. During that evolution, new and changing requirements emerge when end-users interact with the system. Users encounter situations that provoke spontaneous complaints or suggestions, which may be the seed of new requirements. However, there are two challenges: How to capture spontaneous reactions and how to focus and let them mature into valid requirements? Traditional requirements engineering techniques are hardly applicable for capturing spontaneous feedback: - End-users are on their own when unforeseen effects or failures occur. Usually, no requirements expert is present to capture feedback or requirements. - It is difficult or impossible to stimulate spontaneous responses in a lab or interview situation. Some phenomena are difficult to anticipate and to reproduce. They emerge from complex interactions and dependencies between users, devices, and services.

Impact: SE members came up with 44 entities. In response to the open question, all 28 students together listed 47 different entities they wanted to feed back to. There was an overlap of

18 entities (41% of 44) which both parties independently brought up as relevant. SE members had defined 26 entities that none of the students nominated in the open question. However, when students saw them on the list, many students declared they wanted to give feedback. 25 of those 26 entities were accepted by at least one student. On average, each student selected 7 from that list. In turn, students nominated 29 additional entities that SE members had not defined. SE members considered 5 of them (17% of 29) very interesting and wanted to define them. They were less interested in 10 others (34%), and explicitly rejected 14 (48%) entities, almost half of the 29 entities proposed by students. Rejected entities were considered outside the scope and responsibility of SE as a provider. They could distract rather than focus feedback. Many entities (15) had been defined via a context in the hallway of the SE department. The density of WiFi antennas was also high. The ratio of entity density per discriminating WiFi signals determines an upper bound for resolution in entity identification. In our case, only a small number of defined entities could not be distinguished based on available WiFi signals. Indistinguishable entities need to be presented together to end-users for final selection. End-user selection becomes tedious when too many candidates are proposed. Entity resolution can be increased by more WiFi antennas, or by using RFID or NFC tags. In our case, resolution over all defined entities was acceptable.

Target Object: Technique/method, tool

Type: Positive

Expression: Implicit

Application Domain: University

Project size: Nine members of the SE department contributed to defining 44 entities in a 30-minute informal session.

RE Practice: N/A

RE Phase: Elicitation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Leibniz UniversitŁt Hannover

Repeatability: First time evaluating case study results

<u>Lesson ID:</u> LL057_11 [Rauf et al., 2011]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2011

Lesson: Use Rauf's et al. framework for the specification of LSs as templates and the extraction of their instances from rich- text documents.

Source: Experiment

Rationale: Software requirements documents (SRDs) are often authored in general-purpose rich-text editors, such as MS Word. SRDs contain instances of logical structures, such as use case, business rule, and functional requirement. Automated recognition and extraction of these instances enables advanced requirements management features, such as automated traceability, template conformance checking, guided editing, and interoperability with requirements management tools such as RequisitePro. The variability in content and physical representation of these instances poses challenges to their accurate recognition and extraction.

Impact: The framework provides an opportunity for combining an open environment of generic and widely adopted text editors with the advanced features offered by RM tools. The recall was 100% for 33 templates and 97%, 95% and 83% for T21,19,24, respectively. PrecisionLS was 100% for all templates except one. PrecisionLC was 100% for 34 templates and 86% for T27 and T34. The extraction time for most templates was less than 2 seconds, which is good for practical purposes. The longest time, 12 seconds, was for T14, whose 115 instances were spread across 5 documents. Another factor affecting the extraction time is the number of LCs: T14 has 13; T23 has only 7. template size is approximately proportional to the no. of its LCs.

The maximum template size was 52 lines, still acceptable for human viewing. In general, the framework makes structured content of rich-text documents accessible for further automatic processing: Rich-text document import. Requirements management tools features. Semantic annotation. Structured query. Analysis of product line requirements.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: N/A

Project size: Used a total of 43 documents24 from three industrial partners, 7 from a use-case document repository, 6 student projects, and 6 downloaded from the Internet by searching with “Software Requirements Specification, “Software Requirements Documents, or SRS as keywords. Our evaluation considers all LSs that had at least four instances within the 43 documents, giving us a total of 36 LSs.

RE Practice: Using templates

RE Phase: Specification, analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time evaluating experiment results

Lesson ID: LL058_11 [Nolan et al., 2011]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2011

Lesson: Apply of risk analysis techniques for managing requirements uncertainty by using the following attributes for requirements: volatility, impact, precedence, time criticality and doing the following analyses: risk index, maturity index, proportional risk index.

Source: Industrial experience

Rationale: In the development of complex systems the requirements for the system will almost always remain uncertain late into the software development. In gas turbine engine control systems at Rolls-Royce, typically 50% of requirements will change between Critical Design Review and Entry into Service. Ignoring or not planning for requirements uncertainty will cause scrap and rework that will manifest later in the project.

Impact: The return on investment of this technique has been between 100:1 and 500:1. The analyses technique described in this paper adds around 1 minute extra effort for analysing each requirement but has been shown to reduce Scrap & Rework on a project from an average of 50% down to a level below 5%. The return on investment for uncertainty analysis and mitigation can be between 100:1 and 500:1, making it one of the most cost-effective improvements a project can apply.

Target Object: Technique/method

Type: Positive

Expression: Explicit

Application Domain: Control Systems department at Rolls-Royce is responsible for the Engine Electronic Controllers (EECs) for a range of small and large gas turbine engines, for the aerospace industry.

Project size: The development of a new engine can take up to 5 years and will be highly evolutionary. The electronics, the engine, and the airframe will evolve and mature through the life of a project causing new functionality and changes to emerge. Historical data shows that between the point of Critical Design Review (a system concept review gate) and Entry into Service, a project will spend approximately 50% of its cost on evolutionary work rather than new product development.

RE Practice: N/A

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Rolls Royce

Repeatability: First time industrial experience

Lesson ID: LL059_11 [Dekhtyar et al., 2011]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2011

Lesson: Analysts tracing experience, amount of effort applied to look for missing links, comfort level with tracing, etc. do not affect final TM accuracy.

Source: Experiment

Rationale: N/A

Impact: The analysts experience, effort applied, etc. do not matter.

Target Object: Technique/method

Type: Negative

Expression: Implicit

Application Domain: University

Project size: Two experimental sites for a total of 84 participants (students)

RE Practice: Assisted tracing, Tracing

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: California Polytechnic State University and University of Kentucky

Repeatability: Confirms the results from a previous study

<u>***Lesson ID:***</u> LL060_11 [Dekhtyar et al., 2011]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2011

Lesson: The initial (lower) TM accuracy and the amount of time an analyst spent vetting links provided by the tool are the most important factors impacting final TM accuracy.

Source: Experiment

Rationale: N/A

Impact: Alters our overall approach to assisted tracing. We can no longer rely on the automated tracing methods to produce high-accuracy results and expect these results to translate into even higher-accuracy ones in assisted tracing settings. While we still consider the quest for high-precision, high- recall automated tracing methods important, we must acknowledge that it will not provide a panacea for assisted tracing.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: University

Project size: Two experimental sites for a total of 84 participants (students)

RE Practice: Assisted tracing, Tracing

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: California Polytechnic State University and University of Kentucky

Repeatability: Confirms the results from a previous study

<u>Lesson ID:</u> LL061_11 [Vogl et al., 2011]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2011

Lesson: Use the CBSP (Component-Bus-System- Property) approach to reconciling requirements and architectures by capturing early decisions about the architecture along with GQM model.

Source: Industrial experience

Rationale: Understanding the relationship between software requirements and a software systems architecture is a big challenge in industrial practice. Part of the challenge stems from the fact that requirements and architectures use different terms and concepts to capture the elements relevant to each. It has been pointed out that crafting a systems requirements and its architecture should be done in a concurrent manner by interleaving their development.

Impact: Applying the 6 architectural dimensions in a lightweight fashion provided good guidance and led to the systematic analysis of requirements for architectural concerns. The effort for building the CBSP model (without the GQM extensions) was about two person days. The process also helped identifying missing requirements (e.g., if a certain architectural dimension was not addressed). The intermediate model facilitates the mapping of requirements to architectures. Furthermore, the intermediate CBSP model eases capturing and maintaining arbitrary complex relationships between requirements and architectural model elements, as well as among CBSP model elements. Deriving architectural styles from CBSP elements is not straightforward. Our basic observation is that one size does not fit all at least not when it comes to understanding the pros and cons of architectural choices. Our observation is that

CBSP and GQM are a good fit, in particular GQM provided a better and more meaningful structure for describing the CP, BP, and SP properties and assessing their satisfiability. A drawback is the current lack of tool support.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Mobile applications

Project size: The taskmind product is currently used by about 6,900 users in 3,000 projects for managing 85,000 tasks and appointments. The size of the code base is about 230 KLOC.

RE Practice: N/A

RE Phase: Elicitation, Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time industrial experience

Lesson ID: LL062_11 [Bjarnason et al., 2011]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2011

Lesson: Complex product & large organization, Low understanding of roles of others, Gaps between roles over time, Unclear vision of overall goal cause communication gaps.

Source: Explanatory case study

Rationale: Communication is essential for software development as its efficiency throughout the entire project life-cycle is a key factor in developing and releasing successful software products to the market.

Impact: Communication gaps have the following effects: Customer expectations not met, Low motivation to contribute to reqs work, Software unit controls what is implemented, Unclear requirements coverage, Test scope mismatch, Communication of incorrect reqs, Quality issues, Wasted effort, Problems with SRS. By closing the communication gaps the requirements may continue all the way through the project life-cycle and be more likely to result in software that meets the customers expectations.

Target Object: Technique/method

Type: Negative

Expression: Implicit

Application Domain: Embedded systems

Project size: A large market-driven software development company, where we have interviewed nine practitioners. The company has around 5000 employees. There are several consecutive releases of a platform (a common code base of the product line) where each of them is the basis for one or more products that reuse the platforms functionality. A major platform release has a lead time of approximately two years and is focused on functionality growth and quality enhancements for a product portfolio. For such projects, typically around 60-80 new features are added, for which approximately 700-1000 system requirements are produced. These are then implemented by 20-25 development teams with around 40- 80 developers per team, assigned to different projects. The requirements legacy database amounts to a very complex and large set of requirements at various abstraction levels in the order of magnitude of 20,000 entities, making it an example of the Very-Large Scale Requirements Engineering context

RE Practice: Communication

RE Phase: Negotiation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time explanatory case study results

Lesson ID: LL063_11 [Carvallo and Franch, 2011]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2011

Lesson: Conduct the followings method: Phase 1: Identification of the underliying platform; Phase 2: Construction of QMs; Phase 3: Evaluation of alternatives using the QMs; Phase 4: Analysis and issue of recommendations to negotiate both initial and emerging requirements and the reconciliation of stakeholders concerns.

Source: Industrial case study

Rationale: Although several methods have been proposed to support OTS component selection, the truth is that in many cases the process is driven by political and other non-technical aspects, considering components as independent and isolated. Because of this, relevant stakeholders requirements and concerns, as well as the implications that the selection of a particular component may bring to the system architecture, are simply ignored. In the worst case this may lead to the selection of unsuited or inappropriate components and eventually to miscarried projects, but more often to situations in which projects froze due to lack of stakeholders agreement in relation to the newly created architectural scenario and some of its emerging requirements.

Impact: Several benefits emerged from the experience: Fosters involvement of parties, Makes lightweight approach to requirements elicitation, Proposes a structured framework for requirements negotiation at all layers, Makes evident hidden requirements and potential risk, Helps to establish real cost of property, Makes clear the scope of suppliers services and legal concerns, Easies the identification of mismatches and the issue of recommendations.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Banking

Project size: Medium size company

RE Practice: Using software quality models

RE Phase: Negotiation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Mutualista Azuay

Repeatability: First time industrial case study results

Lesson ID: LL064_11 [Boutkova and Houdek, 2011]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2011

Lesson: Use semi-automatic identification of features in natural language specifications based on lexical analysis (MIA).

Source: Industrial experience

Rationale: Reuse of requirements leads to reduction in time spent for specification of new products. Variant management of requirement documents is an essential prerequisite in terms of a successful reuse of requirements. It supports the decisions if available requirements can be reused or not. One possibility to document the variability is feature modelling. One main challenge while introducing feature modelling in a grown environment is to extract product features from large natural language specifications. The current practice is a manual review of specifications conducted by domain experts. This procedure is very costly in terms of time.

Impact: The first experience with MIA shows that the generated feature candidate lists do help to reduce the time effort for feature identification, but not significantly. The variability experts use the feature candidate lists willingly as a starting point for the feature identification process. In our experience, it is very complicated for the variability expert to find the rationales for the variability in specifications created by other people. The feature candidate list shows which characteristic of the component or system the variability expert must regard. Based on this feature candidate list a variability expert could review the specification and prepare the questions for the specification author. The number of feature candidates is depending on number of requirements but not in the linear coherence. The typical size of a feature candidate list is about 150 words. The feedback of variability experts is that it is not problematic to remove additionally stop words and to make clusters for the words with same stem. this approach is usable. At present, it is not possible to identify 100% of all variable features and there are many false candidates.

Target Object: Technique/method

Type: Positive, negative

Expression: Implicit

Application Domain: Automotive: passenger car development

Project size: N/A

RE Practice: Reuse

RE Phase: Specification

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Daimler

Repeatability: First time industrial experience

Lesson ID: LL065_11 [Heaven and Letier, 2011]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2011

Lesson: Using Heaven's et al. automated techniques for (i) simulating quantitative goal models so as to estimate the levels of goal satisfaction contributed by alternative system designs and (ii) optimising the system design by applying a multi-objective optimisation algorithm to search through the design space.

Source: Experiment

Rationale: Making decisions among a set of alternative system designs is an essential activity of requirements engineering. It involves evaluating how well each alternative satisfies the stakeholders goals and selecting one alternative that achieves some optimal tradeoffs between possibly conflicting goals.

Impact: Relatively short searches can be performed using the genetic algorithm (e.g. a search with 50 generations for a population of 20 runs in 3.5 minutes) but they return solutions that are still far from the Pareto- optimal. Increasing the population size and number of generations improves the quality of the solutions but it did not produce better solutions than exhaustive search when allowed to run for the same amount time. For example, Figure 6 plots the cost and 14 minute response rate of the solutions explored through an exhaustive search (depicted as squares) and of the solutions returns by the genetic algorithm running (depicted as circles) when allowed to run for the same amount of time. A genetic algorithm approach would still be useful however when working over much larger models where an exhaustive search is not feasible.

Target Object: Technique/method, tool

Type: Positive, negative

Expression: Implicit

Application Domain: Ambulance service system

Project size: A global design space for the LAS system of 4?3?3?3?6 = 648 alternative systems.

RE Practice: Using Quantitative goal models

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: London Ambulance Service

Repeatability: First time evaluating experiment results

Lesson ID: LL066_11 [Brill and Knauss, 2011]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2011

Lesson: Observe anonymous users for requirements elicitation.

Source: Experiment

Rationale: Today, people find themselves surrounded by IT systems in their everyday life. Often they are not even aware that they are interacting with an IT system. More and more of these systems are context adaptive. Requirements to such systems may change for various reasons: The context may fundamentally change when other systems are introduced. New trends and fashions may evolve. Operators need to react quickly to such changes if they want to keep their systems competitive. Traditional approaches to requirements elicitation start to fail in this situation: context adaptive systems serve many users with different profiles. In addition, users may be reluctant to participate in improving it. Thus, it is hard to establish a representative model of requirements. Furthermore, it is hard to capture the context of requirements by subsequent interviews.

Impact: It allows efficient observation of many stakeholders and the derivation of new requirements. In two evaluation scenarios we show that analysts are able to i) prepare the observations based on our concepts, ii) capture the required data in the field, and iii) are able to derive requirements from the contextual observations. Observers are able to do many observations, even in complex situations and when multiple behavioral and context attributes are involved. If too many context attributes have to be considered, technical solutions can help. It is also possible to partition the context attributes over different observers. Data analysis is straight forward and can efficiently be done as long as the data can be loaded into a typical spreadsheet. It took the analyst only one hour to create a rudimentary scatter plot that allowed to assess, whether the choice of a parking lot is correlated to the relation of free parking lots and free parking lots next to pillars. Thus, we assume that the approach scales to real world problems.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Basement garage and a bus stop both situated in the center of Hanover on a workday in the afternoon

Project size: N/A

RE Practice: Using observation techniques

RE Phase: Elicitation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time evaluating experiment results

Lesson ID: LL067_11 [Waldmann, 2011]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2011

Lesson: When applying insights from agile development to requirements engineering, the key concepts from the agile world such as user stories or product owners must be mapped intelligently to appropriate concepts in RE, not copied.

Source: Case study

Rationale: While Requirements Engineering textbooks state that a requirements specification must be complete, in real-life projects we are always starting too late, with too few resources, so we cant do everything. The software development community has solved a similar problem (not having enough resources to implement everything that was asked for) by introducing agile development methods, which offer ways of segmenting the overall project, and choosing which parts to allocate re- sources to.

Impact: Our case study confirmed that a flexible requirements engineer ing process inspired by agile development methods can de- liver results that provide business value, even with severe resource constraints.Our case study also demonstrated that agile requirements engineering activities can indeed feed into development project that follows a classical V-model approach, by making a clear distinction between incremental delivery of requirements vs. non-incremental delivery of implementation. The implementation part also included hardware development subprojects, and our case study demonstrates the feasibility of agile requirements engineering activities preceding development activities which, for technical reasons, cannot deliver in multiple releases.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Hearing solutions, including hearing instruments, hearing protection and communication systems, and implantable hearing systems

Project size: The project duration is more than 3 years from concept to the launch of 1st products. It involves more than 100 R&D engineers, including the equivalent of 2-4 full-time requirements engineers. The system requirements specification currently consists of around 1200 system requirements items, from which component-specific technical requirements are derived.

RE Practice: N/A

RE Phase: Elicitation, analysis, specification, validation

Software Process: Agile

Project Date: N/A

Recording Date: N/A

Organisation Name: Sonova Group

Repeatability: First time industrial case study results

Lesson ID: LL068_11 [Waldmann, 2011]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2011

Lesson: When applying agile methods to RE, Inform stakeholders about the process by: making our release schedule more transparent, and we need to communicate the objectives and benefits of the multi-staged approach more clearly.

Source: Case study

Rationale: Our stakeholders (product managers, developers, testers) were used to a more waterfall-like approach from previous projects, and we made the mistake to not inform them well enough about our changed approach. This resulted in numerous questions about why the requirements were not complete, how much would change in the next release, etc.

Impact: N/A

Target Object: Technique/method

Type: Negative

Expression: Explicit

Application Domain: Hearing solutions, including hearing instruments, hearing protection and communication systems, and implantable hearing systems

Project size: The project duration is more than 3 years from concept to the launch of 1st products. It involves more than 100 R&D engineers, including the equivalent of 2-4 full-time requirements engineers. The system requirements specification currently consists of around 1200 system requirements items, from which component-specific technical requirements are derived.

RE Practice: Communication

RE Phase: N/A

Software Process: Agile

Project Date: N/A

Recording Date: N/A

Organisation Name: Sonova Group

Repeatability: First time industrial experience

Lesson ID: LL069_11 [Waldmann, 2011]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2011

Lesson: When applying agile methods to RE, Inform stakeholders about triage strategy by: explaining the triage strategy in more detail, and we need to clarify the difference between requirements triage and implementation triage.

Source: Case study

Rationale: There were numerous questions from stakeholders about why a particular feature was not described in detail in a release of the requirements specification, and whether that meant that the feature would not be implemented.

Impact: N/A

Target Object: Technique/method

Type: Negative

Expression: Explicit

Application Domain: Hearing solutions, including hearing instruments, hearing protection and communication systems, and implantable hearing systems

Project size: The project duration is more than 3 years from concept to the launch of 1st products. It involves more than 100 R&D engineers, including the equivalent of 2-4 full-time requirements engineers. The system requirements specification currently consists of around 1200 system requirements items, from which component-specific technical requirements are derived.

RE Practice: Communication

RE Phase: N/A

Software Process: Agile

Project Date: N/A

Recording Date: N/A

Organisation Name: Sonova Group

Repeatability: First time industrial experience

Lesson ID: LL070_11 [Waldmann, 2011]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2011

Lesson: When applying agile methods to RE, Deliver even more frequently by: increasing the number of releases, and reduce the number of new topic areas (user stories) added in each release. This will allow us to react faster and better to changing demands from our readers, in terms of style, level of details, and kinds of information included in the deliverables.

Source: Case study

Rationale: Segmenting delivery into two releases several months apart did solve our problem with insufficient resources, but it did not provide the requirements authors with enough learning opportunities with regard to the needs and expectations of our customers, the developers.

Impact: N/A

Target Object: Technique/method

Type: Negative

Expression: Explicit

Application Domain: Hearing solutions, including hearing instruments, hearing protection and communication systems, and implantable hearing systems

Project size: The project duration is more than 3 years from concept to the launch of 1st products. It involves more than 100 R&D engineers, including the equivalent of 2-4 full-time requirements engineers. The system requirements specification currently consists of around 1200 system requirements items, from which component-specific technical requirements are derived.

RE Practice: N/A

RE Phase: N/A

Software Process: Agile

Project Date: N/A

Recording Date: N/A

Organisation Name: Sonova Group

Repeatability: First time industrial experience

<u>Lesson ID:</u> LL071_11 [Waldmann, 2011]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2011

Lesson: When applying agile methods to RE, Segment deliverables horizontally as well as vertically.

Source: Case study

Rationale: We mainly segmented the requirements specification vertically, i.e. into multiple features or topic areas, each of which contains requirements-related items at multiple levels of detail, such as user needs, use cases, user requirements, and finally technical requirements. While this worked well for our developer customers, other types of customers such as product managers would have benefited from a more layered approach, where we would create agreement and document e.g. user needs at a broader scope, and then progress to analyze and document use cases at the same or even a smaller scope.

Impact: N/A

Target Object: Technique/method

Type: Negative

Expression: Explicit

Application Domain: Hearing solutions, including hearing instruments, hearing protection and communication systems, and implantable hearing systems

Project size: The project duration is more than 3 years from concept to the launch of 1st products. It involves more than 100 R&D engineers, including the equivalent of 2-4 full-time requirements engineers. The system requirements specification currently consists of around 1200 system requirements items, from which component-specific technical requirements are derived.

RE Practice: N/A

RE Phase: Specification

Software Process: Agile

Project Date: N/A

Recording Date: N/A

Organisation Name: Sonova Group

Repeatability: First time industrial experience

Lesson ID: LL072_11 [Waldmann, 2011]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2011

Lesson: When applying agile methods to RE, dont try to create one deliverable for all customers.

Source: Case study

Rationale: Much like different customers of the business product may need different product configurations, the customers of the RE product need different configurations, too. The requirements model comprises user needs, constraints, background information, use cases and line-item requirements at multiple levels of detail. Product managers are interested in user needs, use cases, background information, and high-level requirements. Developers are interested in use cases and line-item requirements that can be converted to tasks. Testers may not want to see user needs and background information, but only testable line-item requirements. In other words, the RE process serves multiple“customers, which have different needs and expect different products.

Impact: Our requirements management tool allowed us to create targeted reports for different customer groups, which contained just the information they were interested in. Also, different report formats can match the working style of different customer groups, e.g. a narrative Word document, an Excel table with line-item requirements, or an HTML page with anchors that can be linked to.

Target Object: Technique/method

Type: Positive

Expression: Explicit

Application Domain: Hearing solutions, including hearing instruments, hearing protection and communication systems, and implantable hearing systems

Project size: The project duration is more than 3 years from concept to the launch of 1st products. It involves more than 100 R&D engineers, including the equivalent of 2-4 full-time requirements engineers. The system requirements specification currently consists of around 1200 system requirements items, from which component-specific technical requirements are derived.

RE Practice: N/A

RE Phase: Specification, documentation

Software Process: Agile

Project Date: N/A

Recording Date: N/A

Organisation Name: Sonova Group

Repeatability: First time industrial experience

<u>Lesson ID:</u> LL073_11 [Waldmann, 2011]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2011

Lesson: When applying agile methods to RE, be prepared to elicit & document multiple levels of detail simultaneously.

Source: Case study

Rationale: In the beginning, we wanted to proceed sequentially from high-level product concepts and statements of user needs, via detailed requirements that are understandable by product managers, and finally to detailed technical requirements. During the requirements elicitation, information at all levels of detail was offered by the stakeholders at the same time, and we chose to accept and document all that information whenever it became available.

Impact: Filtered reports still allowed us to publish and create agreement between stakeholders at successively refined levels of detail.

Target Object: Technique/method

Type: Positive

Expression: Explicit

Application Domain: Hearing solutions, including hearing instruments, hearing protection and communication systems, and implantable hearing systems

Project size: The project duration is more than 3 years from concept to the launch of 1st products. It involves more than 100 R&D engineers, including the equivalent of 2-4 full-time requirements engineers. The system requirements specification currently consists of around 1200 system requirements items, from which component-specific technical requirements are derived.

RE Practice: N/A

RE Phase: Elicitation

Software Process: Agile

Project Date: N/A

Recording Date: N/A

Organisation Name: Sonova Group

Repeatability: First time industrial experience

Lesson ID: LL074_11 [Post et al., 2011]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2011

Lesson: Use Post's et al. algorithm to automatically check for requirements vacuity.

Source: Experiment

Rationale: A requirement is vacuous in a set of requirements if it is equivalent to a simpler requirement in the context of the other requirements. For example, the requirement if A then B is vacuous together with the requirement not A. The existence of a vacuous requirement is likely to indicate an error.

Impact: An automatic check that proves the absence of vacuity is beneficial. In fact with the help of our vacuity check we discovered one error that was not discovered in the manual review.

Target Object: Technique/method, tool

Type: Positive

Expression: Implicit

Application Domain: Automotive: car multimedia, driving assistance, engine controlling, and powertrain development

Project size: Eight specifications

RE Practice: Automatic checking

RE Phase: Analysis, validation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Bosch

Repeatability: First time evaluating experiment results

<u>***Lesson ID:***</u> LL075_11 [Wever and Maiden, 2011]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2011

Lesson: Mismatch between titles and actual responsibility, mismatch of business analysis experience level and size of project, lack of resources, Training is ad hoc without following it through, no leadership and management support are factors that inhibit effective requirements work in business-oriented projects.

Source: Survey

Rationale: Despite trends to standardize business analysis practices through accreditation bodies such as the International Institute of Business Analysis (IIBA), imparting more common requirements knowledge through training such the International Requirements Engineering Board (IREB) and the emergence of different organizational standards and regulators, there is increasing anecdotal evidence that business analysts are not able to use the skills taught, and concerns that, as a result, their work is becoming more fragmented and less rewarding. Indeed, some proponents of agile development methods even argue that the traditional role of business analysts is changing fundamentally.

Impact: An unclear scope makes it difficult to manage stakeholder expectations and results in vague estimation with scope creep and lack of scope management often presenting major obstacles on projects. The business analysts lack of scope involvement and management may also contribute to a poor stakeholder engagement strategy. The absence of requirements validation within a presumably time-stricken project environment that also suffers from a lack of process and guidance would inevitably invite failure.

Target Object: People

Type: Negative

Expression: Implicit

Application Domain: N/A

Project size: 596 past attendees of the fundamental business analysis course ran by Software Education. A total of 127 subjects completed the survey

RE Practice: N/A

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time survey results

Lesson ID: LL076_11 [Jones, 2011]

Journal: N/A

Conference: N/A

Workshop: International Workshop on Managing Requirements Knowledge

Year: 2011

Lesson: Creating word based requirements if no tool is used.

Source: Industrial experience

Rationale: N/A

Impact: Enabled us to develop our requirements processes such that we were able to describe our needs in some detail. This proved to be a great asset when procuring a tool.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Hearing instruments

Project size: Employing close to 7000 people we operate globally under a number of brands. Our 2009/10 turnover exceeded CHF 1.5 bn (an increase of 20.1% on the previous year).

RE Practice: N/A

RE Phase: Specification

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Phonak

Repeatability: First time industrial experience

Lesson ID: LL077_11 [Jones, 2011]

Journal: N/A

Conference: N/A

Workshop: International Workshop on Managing Requirements Knowledge

Year: 2011

Lesson: To reduce a (very) long list of RE tools candidates to a short list in less than a day is by asking yourself “Do the vendors on my long list understand where I am right now?, “Do they understand what it feels like to be on this side of the fence?, “Do they understand my User Needs?, and “Do they know what I need from them right now? In short, “Do they DO Requirements Engineering? Simply, to give each vendor web site five minutes to tell us what the product can do. If they can describe their tool in such a short period of time they may have done some real requirements engineering.

Source: Industrial experience

Rationale: N/A

Impact: This very simple approach reduced our list of 71 vendors to a short list of only 6!

Target Object: Technique/method

Type: Positive

Expression: Explicit

Application Domain: Hearing instruments

Project size: Employing close to 7000 people we operate globally under a number of brands. Our 2009/10 turnover exceeded CHF 1.5 bn (an increase of 20.1% on the previous year).

RE Practice: N/A

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Phonak

Repeatability: First time industrial experience

Lesson ID: LL078_11 [Jones, 2011]

Journal: N/A

Conference: N/A

Workshop: International Workshop on Managing Requirements Knowledge

Year: 2011

Lesson: Create a description of your business, a sample fictitious requirements document (so NDAs werent needed) and multiple scenarios for the tool vendors to demonstrate. This includes both sample requirements documents and specific requirements for your tool. In addition we also created a sample scenario for them to follow. A step by step list of actions that we would like them to demonstrate with the document we delivered. We invited the six short listed vendors to visit us and show us how their product would satisfy our needs. In addition two internal staff were nominated to represent our current tools (Word and Excel) and set the bar for the others to beat.

Source: Industrial experience

Rationale: N/A

Impact: Aside from taking the vendors away from their tried and tested sales presentations this was also a useful exercise for us in that it forced us to think about what we really did on a day-to-day basis. The results were as follows: Some vendors agreed to do demonstrations; some did not bother to respond. Some visited and demonstrated the scenarios as requested; others offered “out of the box sales presentations. Some sent team members with both technical expertise and customer facing (requirements elicitation) skills; others had a “techie on the end of a phone. One vendor replied: “Im very happy to put a demo together, but if ¡xx¿ is a requirement, we might not be the right choice for you. If that puts us out of the running, thats regrettable. I am absolutely convinced that you would be very successful with ¡tool¿ in your organization, but I cant let your time be wasted.

Target Object: Technique/method

Type: Positive

Expression: Explicit

Application Domain: Hearing instruments

Project size: Employing close to 7000 people we operate globally under a number of brands. Our 2009/10 turnover exceeded CHF 1.5 bn (an increase of 20.1% on the previous year).

RE Practice: N/A

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Phonak

Repeatability: First time industrial experience

Lesson ID: LL079_11 [Jones, 2011]

Journal: N/A

Conference: N/A

Workshop: International Workshop on Managing Requirements Knowledge

Year: 2011

Lesson: In order to select the right tool for you, you must take control of the selection process and drive it to completion. It is vitally important that you know your own tool requirements. This must include things like resource availability for tool use and admin. With this you can match the tool to the resources available. In our example the upkeep of a high end tool would have drained a significant proportion of the resource available in the team.

Source: Industrial experience

Rationale: N/A

Impact: Enabled us to compare like with like, confirmed that the vendors could deliver a tool to meet our needs, took the vendors away from their polished marketing routine and towards our reality as users of the tool. What is more, the selection process provided us with a great opportunity to review our own ways of working. The selected tool has been in place for close to two years and it has proven to have been a good choice, is easy to use (without consultancy services), has scaled to meet our needs and remains easy to use and support.

Target Object: Technique/method

Type: Positive

Expression: Explicit

Application Domain: Hearing instruments

Project size: Employing close to 7000 people we operate globally under a number of brands. Our 2009/10 turnover exceeded CHF 1.5 bn (an increase of 20.1% on the previous year).

RE Practice: N/A

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Phonak

Repeatability: First time industrial experience

<u>Lesson ID:</u> LL080_11 [Jones, 2011]

Journal: N/A

Conference: N/A

Workshop: International Workshop on Managing Requirements Knowledge

Year: 2011

Lesson: Retain the formatting your stakeholders understood.

Source: Industrial experience

Rationale: N/A

Impact: It meant that we did not need to re-educate our stakeholders or try to reinvent our quality system. A measure of how successful this was, is that it took many months for stakeholders to realise we had migrated from manual document product to generating reports from a tool.

Target Object: Technique/method

Type: Positive

Expression: Explicit

Application Domain: Hearing instruments

Project size: Employing close to 7000 people we operate globally under a number of brands. Our 2009/10 turnover exceeded CHF 1.5 bn (an increase of 20.1% on the previous year).

RE Practice: N/A

RE Phase: Specification

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Phonak

Repeatability: First time industrial experience

Lesson ID: LL081_11 [Daramola et al., 2011]

Journal: N/A

Conference: N/A

Workshop: International Workshop on Managing Requirements Knowledge

Year: 2011

Lesson: Use Daramola's et al. framework and tool prototype that facilitates the early identification of potential system hazards from requirements and the reuse of previous experience for conducting HAZOP.

Source: Experiment and field assessment by experts

Rationale: The capability to identify potential system hazards and operability problems, and to recommend appropriate mitigation mechanisms is vital to the development of safety critical embedded systems. Hazard and Operability (HAZOP) analysis which is mostly used to achieve these objectives is a complex and largely human-centred process, and increased tool support could reduce costs and improve quality.

Impact: Each of the three industry experts that took part in the assessment returned an evaluation report from which we computed a mean weighted score of 3.27 (out of 5) for the KROSA tool in relation to the evaluation objectives of the field assessment. The tool obtained its highest mean score ratings (out of 5) in the aspects of support for reuse (4.08), sensitivity (3.67), confidence (3.25), and accuracy of result (3.25), while the lowest mean score ratings were in the aspects of: limitations (3.0) and correctness of result (2.7). The experts were unanimous in confirming that the tool will be a valuable support for the conduct of HAZOP, with the potential to alleviate the complexity of the HAZOP process by enabling reuse of experience.

Target Object: Tool

Type: Positive

Expression: Implicit

Application Domain: (1) Rail Lock System (2) Steam Boiler Control system; (3) Adaptive Cruise Control (ACC) System.

Project size: N/A

RE Practice: N/A

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time field assessment results

Lesson ID: LL082_11 [Merten et al., 2011]

Journal: N/A

Conference: N/A

Workshop: International Workshop on Managing Requirements Knowledge

Year: 2011

Lesson: Use the Sunburst and Netmap visualization techniques for representing traceability links be- tween requirements knowledge.

Source: Evaluating example

Rationale: The representation of traceability links in requirements knowledge is vital to improve the general understanding of requirements as well as the relevance and consequences of relations between requirements artifacts and other artifacts in software engineering. Various visualization techniques have been developed to support the representation of traceability information, e.g. traceability matrices, graphs and tree structures. However, these techniques do not scale well on large amounts of artifacts and often do not provide additional functionality to present supplementary data.

Impact: Visualizing traceability links will directly support three main activities related to these links: recovering, browsing and maintenance of traceability links. Browsing, in turn, supports the understanding of requirements knowledge. Generally, traceability links substantiate the meaning of artifacts as they are shown in their specific context. Three exemplary questions have been answered by applying Sunbust and Netmap visualizations on realistic specification data. The questions could be answered using the combination of both visualization techniques. The questions exemplified in this paper are very basic and may also be answered differently, e.g. using complex database queries

Target Object: Technique/method, tool

Type: Positive

Expression: Implicit

Application Domain: N/A

Project size: N/A

RE Practice: Visualizing traceability information, Tracing

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time evaluating example results

Lesson ID: LL083_11 [Sharma and Biswas, 2011]

Journal: N/A

Conference: N/A

Workshop: International Workshop on Managing Requirements Knowledge

Year: 2011

Lesson: Use courteous logic based representations for specifying requirements.

Source: Case study

Rationale: N/A

Impact: Courteous logic based requirements representation meets the consolidated set of desirable features for a requirements specification language: Abstract representation of real-world, Not affected by design and development details, Validation in terms of observable behavior of the system, Ensuring consistency, Maintainability and Traceability, Well understood by the involved parties

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Educational institute

Project size: We identified a total of 8 entities along with their attributes in this sub-system.

RE Practice: Using a specification language

RE Phase: Specification

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time evaluating case study results

Lesson ID: LL084_11 [Teruel et al., 2011]

Journal: N/A

Conference: N/A

Workshop: EmpiRE

Year: 2011

Lesson: Use the CSRML notation for requirements modelling.

Source: Controlled experiment

Rationale: As for single user systems, a proper specification of software requirements is a very important issue to achieve the quality of the collaborative systems. Nevertheless, many of these requirements are from a non-functional nature because are related to the user's need of being aware of other users, that is, the workspace awareness.

Impact: the subjects groups that was given the CSRML models obtained a better score than those who try to understand the i* one. the CSRML notation improve the understandability of requirements model respect to i*.

Target Object: Language

Type: Positive

Expression: Implicit

Application Domain: University

Project size: 30 Computer Science students (15 Group 1; 15 Group 2)

RE Practice: Using a requirements model

RE Phase: Specification

Software Process: N/A

Project Date: May-11

Recording Date: N/A

Organisation Name: Univertisy of Castilla La Mancha (Albacete, Spain)

Repeatability: First time experiment results

Lesson ID: LL085_11 [Erra and Scanniello, 2011]

Journal: N/A

Conference: N/A

Workshop: EmpiRE

Year: 2011

Lesson: Use the brainstorming method for elicitation rather than the Think-Square-Pair method.

Source: Controlled experiment

Rationale: Both in the traditional and global software development contexts, requirements engineering aims at defining the features that the system must have (i.e., functional requirements) or constraints (i.e., quality or pseudo requirements) that it must satisfy to be accepted by customers. In order to model functional requirements, several approaches have been proposed in the past, and of these, behavioral modeling is a common part of the most broadly employed ones. Behavioral modeling includes the requirements elicitation phase in which stakeholders communicate and cooperate to solve problems and to represent functional requirements in terms of use case diagrams and use cases.

Impact: The study reveals a significant difference in terms of time needed to create use cases in favor of the face-to-face interaction with no significant impact on their quality.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: University

Project size: 27 were students of a Software Engineering course of the Bachelor program, while 9 were students of a Computer Graphic course of the Master program

RE Practice: Brainstorming

RE Phase: Elicitation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: University of Basilicata

Repeatability: First time experiment results

Lesson ID: LL086_11 [Isaacs and Berry, 2011]

Journal: N/A

Conference: N/A

Workshop: EmpiRE

Year: 2011

Lesson: Do not skip the requirements gathering process.

Source: Case study

Rationale: In the traditional software development lifecycle (SDLC), requirements engineering (RE) is arguably one of the most important stages. The role of RE has been described in literature as very important to identify stakeholders, detect problems, explore different solutions, and to decide what to implement

Impact: The lack of any requirements gathering process apparently led to missing functions in the product, reduced productivity among the projects members, and poor cost estimation. This lack converted a potentially profitable project into a liability. In the end, the project members completed the product, but much time was wasted. A requirements specification could have saved this time. the lack of requirements affected the effort, cost, and development time estimation process negatively. In particular, five of the seven project members strongly agreed that the lack of requirements affected their estimation, while two agreed that the lack made a difference.

Target Object: Policy

Type: Negative

Expression: Implicit

Application Domain: N/A

Project size: 600 employees in 26 offices around the world, and of these, from 225 to 250 are in O. Xs revenues in Fiscal 2011 from about 20 product suites and about 100 bundleable services was US 99.2 million. The average software development project in O had from three to five developers and one quality assurer, took 18 to 6 months, generated 2.8 MB of delivered source code.

RE Practice: N/A

RE Phase: Elicitation, analysis, specification, validation

Software Process: Agile SDLC

Project Date: Jul-12

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

<u>Lesson ID:</u> LL087_11 [Aceituna et al., 2011]

Journal: N/A

Conference: N/A

Workshop: EmpiRE

Year: 2011

Lesson: Use the NLtoSTD method for requirements validation.

Source: Experiment

Rationale: Requirements engineering is one of the most important and critical phases in the software development life cycle, and should be carefully performed to build high quality and reliable software. However, requirements are typically gathered through various sources and represented in natural language (NL), making requirements engineering a difficult, fault prone, and a challenging task.

Impact: For team effectiveness, the results show that NLtoSTD is beneficial at finding the incompleteness in the requirements when the subjects clearly understand the process of translating the NL to STD-BBs. Also, we find that the characteristics of the NL requirement specifications affected the effectiveness results The results clearly show that the fault checklist method is more efficient at finding faults. This is mainly because: 1) the fault checklist looks for ten types of faults whereas NLtoSTD focuses on detecting only two types of faults (MF and AI); and 2) four (out of 11) subjects using NLtoSTD found no true faults due to their misunderstanding of the translation process.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Case study setting: University. Used systems: A web-based tool for managing student elections, A campus event calendar, A help desk management system, An intelligent rating system for electronic entertainment media, An event registration system

Project size: 16 computer science graduate students. A web-based tool for managing student elections: 42 pages, A campus event calendar: 21 pages, A help desk management system: 28 pages, An intelligent rating system for electronic entertainment media: 17 pages, An event registration system: 33 pages

RE Practice: N/A

RE Phase: Validation

Software Process: N/A

Project Date: Jul-12

Recording Date: N/A

Organisation Name: North Dakota State University

Repeatability: First time experiment results

Lesson ID: LL088_11 [Reggio et al., 2011]

Journal: N/A

Conference: N/A

Workshop: EmpiRE

Year: 2011

Lesson: Use Reggio's et al method for detecting flaws in business process models expressed as activity diagrams

Source: Document analysis study

Rationale: Business process modelling is often used in the initial phases of traditional software development to reduce faulty requirements and as starting point for building SOA based applications. Often, modellers produce business process models without following recognized guidelines and opt for light models where nodes representing the actions are simply decorated with natural language text. The potential consequence of this practice is that the quality of built business process models may be low.

Impact: Preliminary results show the effectiveness of our manual method in revealing errors and style violations. Overall, we found 55 flaws (whereof 23 semantic errors) in the 14 analysed models. As far as effort is concerned, we can say that the effort for applying the method is not too much expensive (2 or 3 times the time to copy/duplicate the corresponding light version), al least this is true when an expert modeller applies it.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: N/A

Project size: 14 light business process models (our dataset)

RE Practice: N/A

RE Phase: Validation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time document analysis results

Lesson ID: LL089_11 [Kong and Hayes, 2011]

Journal: N/A

Conference: N/A

Workshop: EmpiRE

Year: 2011

Lesson: PSVM does not perform significantly better than VSM in tracing.

Source: Experiment

Rationale: The automatic generation of traceability links attempts to reduce the burden of building requirements traceability matrices (RTMs) that will be vetted by a human analyst before use in verification and validation tasks such as criticality assessment or change impact analysis. Information Retrieval (IR) techniques, notably the Vector Space Model (VSM), have been used with some success to build textual artifact traceability matrices. A limitation of the VSM is that it disregards word or term location and the relationship between words in the textual artifacts being traced.

Impact: Results showed that the PVSM had slightly higher MAP for two of the four datasets used in the experiment. Upon reviewing the candidate links, a number of false links were ranked high due to the presence of common terms but differed in one or two golden keywords. The PVSM shares this limitation with VSM in that both models do not consider the semantics of a sentence when weighting a document. The PVSM, however is more susceptible to overweighting these links since it is unable to determine the significance of the missing keyword when detecting terms in close proximity.

Target Object: Tool

Type: Neutral

Expression: Implicit

Application Domain: NASA-provided CM-1 (a science instrument) project, Pine is an open source email client, ChangeStyle is a Java- based style checker, EasyClinic is a collection of software artifacts used in the development of a software system to manage a medical ambulatory.

Project size: The experiment uses datasets selected based on answer set availability. CM1Subset1 is a subset of the NASA-provided CM-1 (a science instrument) project containing 22 high-level requirements, 53 low-level requirements, and 40 true links. Pine is an open source email client that has 49 high-level requirements, 133 use cases, and contains 246 true links. ChangeStyle is a Java- based style checker that has 32 high-level requirements, 17 test cases, and 23 true links. EasyClinic is a collection of software artifacts used in the development of a software system to manage a medical ambulatory. The experiment traces between the 30 use cases and 47 code classes in the collection, with 93 true links in the answerset.

RE Practice: Automatic tracing, Tracing

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time experiment results

Lesson ID: LL090_11 [Morandini et al., 2011]

Journal: N/A

Conference: N/A

Workshop: EmpiRE

Year: 2011

Lesson: Use Tropos4AS instead of Tropos for requirements analysis.

Source: Controlled experiment

Rationale: Requirements Engineering (RE) is a well-known discipline that studies processes, methods, languages and tools to support system engineers during the analysis of the requirements of the system they need to develop and maintain. Several goal-oriented modeling languages and methods (e.g., KAOS, i* and Tropos) have been proposed to analyze requirements and to generate clear and meaningful requirements specifications.

Impact: Tropos4AS is more effective than Tropos in describing requirements of self-adaptive systems, especially when the models are used by novice requirements engineers. Moreover, we also ob- served that Tropos4AS helps in particular the novice users, who outperformed the performance of expert Tropos modelers. Indeed, the performance of the expert Tropos modelers remained quite stable when using or not using the extensions provided by Tropos4AS.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Specifications of two soft- ware systems, which have some features of self-adaptivity and are thus suitable for modeling with both treatments: a Patient Monitoring Agent (PMA) and a Washing Machine Manager (WMM).

Project size: Twelve subjects, researchers and Ph.D. students working in a research center

RE Practice: Using modeling languages

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: FBK Center for Information Technology

Repeatability: First time experiment results

Lesson ID: LL091_11 [Salfischberger et al., 2011]

Journal: N/A

Conference: N/A

Workshop: IWSPM

Year: 2011

Lesson: Use the Functional Architecture Framework (FAF) to manage requirements.

Source: Case study

Rationale: As companies grow larger and gain more customers, the number of requests from the customer side increases. Managing very large scale requirements management requires special processes. Organizations need a structure to organize these requirements processes to be able to satisfy the stakeholders.

Impact: Implementing the full FAF allows a very high volume of requirements to be managed efficiently. An important factor in the implementation of the full FAF is the support of all departments.

Target Object: Technique/method

Type: Positive

Expression: Explicit

Application Domain: ERP Vendor

Project size: 70000 customers and 8000 employees. The database holds about 9000 market requirements and 2500 corresponding product requirements.

RE Practice: Using a framework

RE Phase: Managing

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

Lesson ID: LL092_11 [Svensson et al., 2011]

Journal: N/A

Conference: N/A

Workshop: IWSPM

Year: 2011

Lesson: Use the QUPER prototype tool for supporting release planning of quality requirements..

Source: Case study

Rationale: Release planning plays an important role for the success of a software product vendor that develops software- intensive incremental products. It is important that the software product is released to the market at the right time, and offers higher quality than the competitors. However, an especially challenging problem for a software product vendor is to set the right quality target in relation to future market demands and competitor products.

Impact: The study showed that the tool provides a clear overview of the current market situation by the generated roadmaps, and to reach an alignment between practitioners, e.g., product managers and developers, of what level of quality is actually needed. In general, all subjects agreed that the QUPER prototype tool would help in the important shift of focus from FR to QR by providing a clear and understandable representation of the market (competing products) as a basis for QR.

Target Object: Tool

Type: Positive

Expression: Implicit

Application Domain: Company that develops software and hardware for the mobile handset market.

Project size: The case company has more than 5,000 employees and develops their products for a global, competitive market using a product line approach. The companys requirements database consists of more than 20,000 requirements where approximately 25% of the requirements are quality requirements. Five practitioners, three product managers, one project manager, and one test manager participated in the tool evaluation.

RE Practice: N/A

RE Phase: Managing

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time evaluating case study results

Lesson ID: LL093_11 [**?**]

Journal: N/A

Conference: N/A

Workshop: MoDRE

Year: 2011

Lesson: Build sequence diagram using the Model-Driven Development.

Source: Case study

Rationale: Scenario modeling can be realized through different perspectives. In UML, scenarios are often modeled with activity models, in an early stage of development. Later, sequence diagrams are used to detail object interactions. The migration from activity diagrams to sequence diagrams is a repetitive and error-prone task. Model-Driven Development (MDD) can help streamlining this process, through transformation rules. Since the information in the activity model is insufficient to generate the corresponding complete sequence model, manual refinements are required.

Impact: a decrease in the number of operations required to build and refine the sequence model of approximately 64% when using MDD, when compared to the manual approach. The advantages, from a quality point of view, include: (i) a reduction in the effort building the sequence model, (ii) increased traceability among models (through the semi-automatic translation rules), (iii) error prevention when migrating from different scenarios notations, and (iv) support for reuse of sequence models design best practices, thus providing a good stepping stone for high quality scenario modeling.

Target Object: Technique/method, tool

Type: Positive

Expression: Implicit

Application Domain: A software system offering operations on photos, music and videos on mobile devices

Project size: N/A

RE Practice: Using model-driven development

RE Phase: Specification

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time evaluating case study results

Lesson ID: LL094_11 [Marincic et al., 2011]

Journal: N/A

Conference: N/A

Workshop: MoDRE

Year: 2011

Lesson: Perform the following two non-formal steps for model validation: testing the model itself by review, simulation and model-checking; and explaining the model to the model stakeholders.

Source: Case study

Rationale: The result of a model-based requirements verification shows that the model of a system satisfies (or not) formalised system requirements. The verification result is correct only if the model represents the system adequately. No matter what modelling technique we use, what precedes the model construction are non-formal activities. During these activities the modeller has to learn how the system works, what the requirements are, and to decide what is relevant to model and how to do it. Due to a partly non-formal nature of modelling steps, we do not have a formal proof that the model represents the system adequately. The most we can do is to increase the confidence in the model.

Impact: N/A

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: A company that produces printers for office use

Project size: A large company, with a few hundred people employed in research and development department.

RE Practice: N/A

RE Phase: Validation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

Lesson ID: LL095_11 [?]

Journal: N/A

Conference: N/A

Workshop: re@run.time

Year: 2011

Lesson:Use the Boilerplates method for requirements elicitation and by explicitly modeling the runtime requirements knowledge for further application using ontologies for knowledge representation.

Source: Case study

Rationale: Current industrial automation systems are becoming more and more complex, and typically involve different phases of engineering, such as design time and runtime. System requirements, which are usually elicited during design time by engineers, currently are not sufficiently represented at runtime, like the runtime enforcement of safety requirements for industrial automation systems. Such kind of enforcement usually is very hard to model and predict at de- sign time. Hence, the need exists to capture and manage safety requirements at design time and runtime, since safety requirements of industrial automation systems may lead to high risks if not addressed properly.

Impact: Major result was that the Boiler- plates and explicit engineering knowledge are well suited to capture and enforce runtime safety requirements of industrial automation systems. Initial results show that the analysis of different scenarios of possible failures of the industrial process plant model at runtime allows for a more efficient and effective elicitation and management of runtime safety requirements, which are not easily predicted or modeled at design time.the usage of a quite common method for requirements elicitation, such as the Boilerplate notation, generally enables an easier safety requirements elicitation at design time. On the other hand side, the transformation from Boilerplate notation into if- then-else rules is intuitively understandable and can be used to check whether a system is really following a set of rules using periodical rule checking mechanisms.

Target Object: Technique/method

Type: Positive

Expression: Explicit

Application Domain: Educational model of an industrial process plant for enforcing safety requirements at runtime

Project size: 2 scenarios

RE Practice: Using a framework

RE Phase: Elicitation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time evaluating case study results

<u>***Lesson ID:***</u> LL096_11 [Chopra and Singh, 2011]

Journal: N/A

Conference: N/A

Workshop:RES4

Year: 2011

Lesson: Use Colaba to design business processes.

Source: Case study

Rationale: Across-organizational processes naturally involve multiple stakeholders with distinct business interests. Yet, current process modeling approaches are conceptually centralized.

Impact: In this particular example, the seller and the buyer eventually arrive at a mutually acceptable proposal; in another instance, it may happen that they cannot. Nonetheless, stakeholders would be better off at resolving conflicts armed with Colaba than without. This argumentation instance, like all others, becomes part of the repository, and is available for future reference. Sub- sequently, new organizations who have a similar purpose in mind may exploit this knowledge for better designing their business processes, especially when guided by past practice and design rationales.

Target Object: Tool

Type: Positive

Expression: Implicit

Application Domain: Ace Semiconductors is a producer of silicon wafers used in the manufacturing of semiconductors. NanoCorp is manufacturer of semiconductors.

Project size: N/A

RE Practice: Using a social network

RE Phase: Negotiation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Ace Semiconductors and NanoCorp.

Repeatability: First time evaluating case study results

Lesson ID: LL097_11 [Hoffmann et al., 2011]

Journal: N/A

Conference: N/A

Workshop:RES4

Year: 2011

Lesson: Combine service-oriented requirements engineering with service bundling in a new approach for the design of user-centric portals: first, identifying user needs and matching them to existing adequate services provided by the organization; second, the decision makers negotiate all services according to user wishes and effort expectancy and determine the functionality of the portal; third, service bundling is used to build highly user-centric portal structures that are derived from the users characteristic life events or demographic situations.

Source: Case study

Rationale: Portals are platforms combining services from various sources for various user groups. Service bundling from marketing can serve as a new way to enable mass customization of portal services for heterogeneous user groups using existing services.

Impact: This approach is promising because of the possibility to validate requirements with the help of existing services. It also helps to determine the possible functionality of the portal in a fast manner. The integration of the service provider in early stages of the development process helps to avoid late and cost-intensive faults. Furthermore, the decision makers can base their decision on single functionalities on estimated effort and users demands. In addition, the new approach is supported through the user-centric development of portal structures based on services to which decision makers are already committed.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: University

Project size: The infrastructure of the university is comparable to that of other big organizations.

RE Practice: N/A

RE Phase: Elicitation, negotiation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: The Kassel University

Repeatability: First time evaluating case study results

Lesson ID: LL098_11 [Pasquale and Spoletini, 2011]

Journal: N/A

Conference: N/A

Workshop:RES4

Year: 2011

Lesson: Use the FLAGS language to represent requirements as fuzzy temporal formulas and identify partial violations at the temporal level and the monitoring framework to assess FLAGS formulas at runtime.

Source: Experiment

Rationale: Service compositions are an important family of self-adaptive systems, which need to cope with the variability of the environment (e.g., heterogeneous devices, changing context), and react to unexpected events (e.g., changing components) that may take place at runtime. To this aim, it is fundamental to continuously assess requirements while the system is executing and detect partial mismatches or handle uncertainty. Detecting the entity of a violation is very helpful, since it can guide the way applications adapt at runtime.

Impact: Generally we can claim that monitoring fuzzy requirements at runtime is feasible since it introduces an overhead, in terms of monitoring time, that is slightly greater than that measured to monitor crisp formulas.

Target Object: Language

Type: Positive

Expression: Implicit

Application Domain: N/A

Project size: N/A

RE Practice: N/A

RE Phase: Specification

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time evaluating experiment results

Lesson ID: LL099_11 [Bahrs and Nguyen, 2011]

Journal: N/A

Conference: N/A

Workshop:RES4

Year: 2011

Lesson: Use the Smarter architecture & engineering (SmarterAE) approach for requirements management. It is way of rethinking the requirements, architecture and systems engineering lifecycles on SOA, Modernization and Transformation projects.

Source: Industrial experience

Rationale: Many project struggle with the so called requirements hand off. This is the situation where requirements are documented in text and handed to a team for implementation. In the majority of projects, the developers struggle with semantics, clarity and on time delivery. This occurs on projects of various time durations and complexity from small embedded systems to large geographically dispersed systems of systems.

Impact: Reduction in costs by 33%. Reduction in time by 40%.Successful partitioning of teams across time zones and organizations. Successful delivery of Claims Processing and Border Management applications in 30 days, from requirement to execution. Predicted savings of 15, 25 and 40% over three years. 60% of projects producing the correct assets. Business process assets more difficult to produced than all other business architecture assets. Asset types and standards continuing to change

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Business Process Management (BPM) project. SOA project. Legacy Modernization project. Transaction Processing project. Business Agility project. Model-driven Architecture (MDA) project. Model-driven Architecture (MDA) project. Real-time mission critical systems project. Product development project

Project size: SmarterAE is typically valuable in large complex enterprise-wide projects.

RE Practice: N/A

RE Phase: Elicitation, managing

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: IBM

Repeatability: Numerous projects and customers are using the SmarterAE approach over the last two years.

Lesson ID: LL100_11 [Cortier et al., 2011]

Journal: N/A

Conference: N/A

Workshop: REVOTE

Year: 2011

Lesson: Voting systems should of course guarantee the confidentiality of the votes (no one should know that a voter has voted in a particular way) but also eligibility (only registered voters can vote, at most once), fairness (the result reflects the actual votes), and verifiability (voters can check that their votes have been counted).

Source: Case study

Rationale: In order to allow voters to vote from their home without using computers, voting systems have been proposed to improve the standard two-envelope system. They consist of somewhat hybrid systems, still using paper ballots but with barcodes (to facilitate the tallying phase using barcode readers) and identifiers that should ensure that votes cannot be linked to voters. They are typically used for elections with intermediate issues such as elections of representatives in unions, companies or many councils. Up to our knowledge and surprisingly, these systems have not been submitted to a careful security analysis (nor even design), in contrast to electronic voting protocols. Some official guidelines nevertheless exist. For example in France, the Commission Nationale de lInformatique et des Liberts (CNIL) which is an independent administrative authority whose mission is to guarantee that data processing complies with human rights, private life, or individual freedoms has recently issued recommendations about electronic voting. A similar recommendation has been issued for postal voting with barcodes.

Impact: N/A

Target Object: Technique/method

Type: Negative

Expression: Implicit

Application Domain: Postal voting system

Project size: Election involving about 30,000 voters.

RE Practice: N/A

RE Phase: Elicitation

Software Process: N/A

Project Date: Spring 2011

Recording Date: N/A

Organisation Name: Tagg Informatique

Repeatability: First time case study results

Lesson ID: LL101_11 [Gibson et al., 2011]

Journal: N/A

Conference: N/A

Workshop: REVOTE

Year: 2011

Lesson: Use operational protoyping when specifying voting systems interface requirements.

Source: Case study

Rationale: Poorly designed voting interfaces increase the effort required to vote and at worst they may interfere with the voters ability to vote as intended. Further, voters prefer a short and quick voting experience with a clear inverse relationship between effort and satisfaction.

Impact: A system that is developed without input from the targetted users has two main problems. Firstly, it is unlikely to do what the users would like it to do. Secondly, it is unlikely that the users will be enthusiastic about its deployment. Users are often very good at criticising a concrete system, whilst being unable to comment on abstract models. In particular, a concrete system helps users to better understand and express their needs. As a secondary benefit, a prototype can improve communication between the users and the developers. This is vital where the problem domain and solution domain are separated by different languages and concepts. In effect, a prototype acts like a common framework for discussion between the two different groups. Finally, encouraging user feedback increases the likelihood that future users will be enthusiastic about adopting the new system/technology. The main strength of our system is in usability. All our trials have shown high usability scores when compared with other voting systems. An advantage of our hybrid system over the traditional paper system, which it is designed to improve upon, is that the voter gets immediate feedback that their vote is marked as intended. The degree of feedback is currently limited to informing the voter as to whether their vote is considered, by the electronic system, to be valid, spoiled or blank. This feedback was specifically introduced to address the problem of incoherency between the paper and electronic counts with respect to identification of spoiled votes. Current trials are intended to show that this feedback may assist the voter in having their intent correctly recorded. This should demonstrate that the system is more accurate than a purely paper system.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Voting system

Project size: Election involving about 30,000 voters.

RE Practice: Prototyping

RE Phase: Elicitation, specification, analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

Lesson ID: LL001_12 [Yang et al., 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Use Yang et al.'s automatic detection of uncertainty in natural language requirements method to identify speculative sentences and determine the scope of uncertainty.

Source: Evaluating Experiment

Rationale: Uncertainty in requirements documents has several undesirable effects. It can lead to system behaviour that does not meet users' expectations, if no proper analysis of the root causes of the uncertainty is performed, and alternatives are not considered. It can also lead to untestable requirements and makes it difficult to plan and estimate development costs. It can cause developers to substitute their own preferences and expectations for the speculative requirements. In short, speculative requirements that survive till the implementation phase are potentially harmful.

Impact: N/A

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: N/A

Project size: 11 full-text requirements documents in which uncertainty cues and their scopes have been manually annotated according to established annotation guidelines

RE Practice: N/A

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time experimental results

Lesson ID: LL002_12 [Arora et al., 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Use Arora's et. al method and algorithms for identifying and resolving feature interactions early in the development life- cycle.

Source: Evaluating case study

Rationale: As new features are developed and integrated into the automobile, they interact with already existing features, sometimes resulting in undesirable behaviours. These undesirable behaviours are referred to as feature interactions, and they result in uncertainties in the system development process if not detected early in the development life-cycle.

Impact: Allows the specification of a system incrementally by gradually adding features and system level details of how features interact with each other, identifying undesirable forms of interactions, resolving them using scenarios, and checking for consistency of the system as a whole. The final result of following our methodology is a consistent and unambiguous specification of the system being specified.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Automotive

Project size: N/A

RE Practice: N/A

RE Phase: Specification, analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Global General Motors R&D

Repeatability: First time case study results

Lesson ID: LL003_12 [Charrada et al., 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Automatically identify outdated requirements using Charrada's et al. method.

Source: Evaluating case study

Rationale: Keeping requirements specifications up-to-date when systems evolve is a manual and expensive task. Soft- ware engineers have to go through the whole requirements document and look for the requirements that are affected by a change. Consequently, engineers usually apply changes to the implementation directly and leave requirements unchanged. Automatically identifying outdated requirements reduces the effort and time needed for the maintenance of requirements specifications significantly and thus helps preserve the knowledge contained in them.

Impact: Identification of Requirements-Related Changes: Using our comparing tool, 33 classes were identified, covering 12 of the 14 requirements-related changes. Among the 33 identified classes, 26 actually contained parts of the 14 changes. The other 7 classes were simple refactorings. Thus, our approach achieved a precision of 26/33=79% and a recall of 12/14=85.7%. Keyword Extraction and Tracing Results: The class tracing has a precision of 0.23 and a recall of 0.23 while our approach has a precision 0.38 and a recall of 0.38, which is an improvement of 66%. Furthermore, for a fixed recall value, the precision of our approach is at least twice as good as for the class-based approach.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Medical care project

Project size: 40 uses cases. Two version: 10 and 11. Java code considered only.

RE Practice: Tracing

RE Phase: Managing

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: North Carolina State University

Repeatability: First time case study results

<u>Lesson ID:</u> LL004_12 [Niu and Mahmoud, 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Use Niu's et al. approach for candidate link generation for the requirements tracing process.

Source: Evaluating exploratory case study

Rationale: Modern requirements tracing tools employ information retrieval methods to automatically generate candidate links. Due to the inherent trade-off between recall and precision, such methods cannot achieve a high coverage without also retrieving a great number of false positives, causing a significant drop in result accuracy. This approach improves the quality of candidate link generation for the requirements tracing process.

Impact: Approach greatly outperforms the base- line in generating the candidate traceability links for three requirements. Not only is recall maintained at a high level (larger than or equal to .93) by using our approach, but precision is markedly increased.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Software-intensive platform that provides technological solutions for service delivery and workforce development in a specific region of the United States.

Project size: 6 business requirements and Java code base containing over 500 classes.

RE Practice: Tracing

RE Phase: N/A

Software Process: Goal-oriented and agile process.

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

<u>Lesson ID:</u> LL005_12 [Gordon and Breaux, 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Use the requirements 'water marking' framework to align and reconcile requirements from multiple jurisdictions (municipalities, provinces, nations).

Source: Evaluating case study

Rationale: Companies that own, license, or maintain personal information face a daunting number of privacy and security regulations. Companies are subject to new regulations from one or more governing bodies, when companies introduce new or existing products into a jurisdiction, when regulations change, or when data is transferred across political borders. The approach is used to align and reconcile requirements from multiple jurisdictions (municipalities, provinces, nations) to produce a single high or low standard of care.

Impact: Reduced the number of requirements a company must comply with by 76% across 8 jurisdictions.

Target Object: Policy

Type: Positive

Expression: Implicit

Application Domain: U.S. data breach notification laws

Project size: Eight regulations based on guidance from a legal expert with 7 years of privacy and security law expertise to highlight regulations that have been a priority for U.S. companies

RE Practice: N/A

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

<u>***Lesson ID:***</u> LL006_12 [Maxwell et al., 2012a]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Use the Adaptability Framework to aid software engineers in predicting what areas of a proposed rule are most likely to evolve, allowing engineers to begin building towards the more stable sections of the rule.

Source: Evaluating case study

Rationale: Over time, laws change to meet evolving social needs. Requirements engineers that develop software for regulated domains, such as healthcare or finance, must adapt their software as laws change to maintain legal compliance. In the United States, regulatory agencies will almost always release a proposed regulation, or rule, and accept comments from the public. The agency then considers these comments when drafting a final rule that will be binding on the regulated domain.

Impact: This framework can aid software engineers in predicting what areas of a proposed rule are most likely to evolve, allowing engineers to begin building towards the more stable sections of the rule. 9 changes that we accurately predicted (true positives), 5 changes we predicted that were not accurate (false positives), 104 legal statements for which we predicted no change and for which no change occurred (true negatives), and 33 legal statements for which we predicted no change and which changed in the final rule (false negatives). Our framework correctly predicted 11 true positives and 104 true negatives for changes, with 5 false positives and 33 false negatives, or 115 correct predictions out of 153 total predictions, or 75% correct. This first analysis correctly predicted 11/16 (68%) of the areas of change, and 104/109 (95%) of the areas of no change, assisting software engineers to prioritize areas for development prior to release of the final rule.

Target Object: Policy

Type: Positive

Expression: Implicit

Application Domain: Healthcare in the US

Project size: The interim and final versions of the Initial Set of Standards, Implementation Specifications, and Certification Criteria for Electronic Health Record Technology (hereafter the EHR Certification Rule). The EHR Certification Rule is 4,736 words long and describes requirements that an EHR must satisfy in order to be certified under the Meaningful Use program

RE Practice: N/A

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Health Information Technology: Initial Set of Standards, Implementation Specifications, and Certification Criteria for Electronic Health Record Technology.

Repeatability: First time case study results

Lesson ID: LL007_12 [Yi et al., 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Use Yi's et al. approach to mine cross-tree binary constraints in the construction of feature models.

Source: Evaluating experiment

Rationale: Identifying features and then building a feature tree takes a lot of effort, and many semi-automated approaches have been proposed to help the situation. However, finding cross-tree constraints is often more challenging which still lacks the help of automation.

Impact: The approach successfully finds binary constraints at a high recall (near 100% in most cases). The precision is unstable and dependent on the test feature models. In most cases the requires constraints are better mined than the excludes constraints; a possible reason is that the rationale behind excludes is often beyond feature descriptions. Continuous feedback from human analysts benefits the mining process, especially for mining excludes constraints. Therefore in practice, our classifier should be used in an interactive way, that is, human analysts check only a few constraint candidates after each turn of mining, and then the classifier repeats the train-optimize-test process again.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Healthcare in the US

Project size: The weather station feature model: 22 features, 196 feature pairs, 6 requires, and 5 excludes. Graph Product Line feature model: 15 features, 91 feature pairs, 8 requires, and 5 excludes

RE Practice: N/A

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: S.P.L.O.T. repository

Repeatability: First time experimental results

Lesson ID: LL008_12 [Shaker et al., 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Use feature-oriented requirements modelling language (FORML) for modelling the behavioural requirements of a software product line.

Source: Evaluating exploratory and confirmatory case study

Rationale: To ease the task of adding new features to a set of existing requirements. feature modularity eases system development and evolution because features can be developed in isolation, in parallel, and by third-party vendors.

Impact: Decomposes a product lines requirements into feature modules, and provides language support for specifying tightly-coupled features as model fragments that extend and override existing feature modules.

Target Object: Technique/method

Type: Neutral

Expression: Implicit

Application Domain: Automotive and telephony

Project size: The automotive case study is an extension of AutoSoft and is adapted from a GM Feature Technical Specification for a family of automotive software features. The case study includes 11 features. The telephony case study is adapted from the Second Feature Interaction Contest, and comprises a telephone-service SPL with 15 features.

RE Practice: N/A

RE Phase: Specification, analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: General Motors

Repeatability: First time case study results

Lesson ID: LL009_12 [Greenyer et al., 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Use Greenyer's et al.technique for the scenario-based specification of component interactions based on Modal Sequence Diagrams.

Source: Evaluating case study

Rationale: Modern technical systems typically consist of multiple components and must provide many functions that are realized by the complex interaction of these components. Moreover, very often not only a single product, but a whole product line with different compositions of components and functions must be developed. To cope with this complexity, it is important that engineers have intuitive, but precise means for specifying the requirements for these systems and have tools for automatically finding inconsistencies within the requirements, because these could lead to costly iterations in the later development.

Impact: When using our ordering pattern, it turns out that the dedicated algorithm outperforms the enumerative method (Enum.) except when P1 and the 15-feature case are considered. Our theory is that the enumerative approach is more efficient for bigger specifications with a high feature- to-MSD ratio. In practice, however, it is more likely to have several MSDs per feature as in this technical example. Thus we expect our approach to be more efficient in most practical cases, but further evaluations are required. The enumerative algorithm does not return a concise formula that identifies the bad products, so the dedicated method also yields improvement in usability.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Automotive

Project size: N/A

RE Practice: N/A

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: RailCab

Repeatability: First time case study results

Lesson ID: LL010_12 [Gross and Doerr, 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: What are typical artifact types that should be contained in an SRS from the viewpoint of downstream development roles in order to accomplish their tasks? descriptions of goals and technical requirements are considered to be the most important artifact types, directly followed by descriptions of quality requirements, data requirements, and interaction and system function descriptions. Also, stakeholder and task descriptions were identified as being important, although slightly less important than other artifacts.

Source: Case study

Rationale: Software requirements specifications play a crucial role in software development projects. Especially in large projects, these specifications serve as a source of communication and information for a variety of roles involved in downstream activities like architecture, design, and testing. In order to create high-quality requirements specifications that fit the specific demands of successive document stakeholders, we need to better understand the particular information needs of downstream development roles.

Impact: All information that is important for architecture design was available. All important information is contained in a personal view. All important information can be found by browsing and bookmarking of important artifacts.

Target Object: Artifact: requirements

Type: Negative

Expression: Implicit

Application Domain: Eye-tracking case study: a real software project based on TORE

Project size: The specification (SRS) comprised three Word documents: Domain Requirements Specification (133 pages), basically comprising artifacts like stakeholder descriptions, goal descriptions, task descriptions, workflow descriptions (both as-is processes & to-be processes), as well as data descriptions. System Requirements Specification (140 pages), containing detailed interaction descriptions in the form of use cases, supplemented with detailed quality requirements, interaction data and technical requirements specifications. Annex Document (82 pages) comprising supplementary materials such as the elicitation guidelines used.

RE Practice: N/A

RE Phase: Elicitation, specification, documentation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

<u>Lesson ID:</u> LL011_12 [Gross and Doerr, 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Which notation should be used to specify artifacts of a certain type? Graphical notations are very useful for architects. If textual descriptions are used, the descriptions should be structured (e.g., by using bullets) with important aspects being highlighted.

Source: Case study

Rationale: Different roles have different information needs.

Impact: The representation of the information is suitable for each role (e.g., UML diagrams) dependent on the activities.

Target Object: Tool

Type: Negative

Expression: Implicit

Application Domain: Eye-tracking case study: a real software project based on TORE

Project size: The specification (SRS) comprised three Word documents: Domain Requirements Specification (133 pages), basically comprising artifacts like stakeholder descriptions, goal descriptions, task descriptions, workflow descriptions (both as-is processes & to-be processes), as well as data descriptions. System Requirements Specification (140 pages), containing detailed interaction descriptions in the form of use cases, supplemented with detailed quality requirements, interaction data and technical requirements specifications. Annex Document (82 pages) comprising supplementary materials such as the elicitation guidelines used.

RE Practice: N/A

RE Phase: Specification, documentation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

Lesson ID: LL012_12 [Gross and Doerr, 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: While descriptions of interactions, system functions, quality and technical requirements, stakeholders and goals were still considered as being (very) important, data requirements, as well as task and workflow descriptions were considered as less important. As-is workflows were even rated as totally unimportant.

Source: Questionnaire

Rationale: Software requirements specifications play a crucial role in software development projects. Especially in large projects, these specifications serve as a source of communication and information for a variety of roles involved in downstream activities like architecture, design, and testing. In order to create high-quality requirements specifications that fit the specific demands of successive document stakeholders, we need to better understand the particular information needs of downstream development roles.

Impact: All information that is important for architecture design was available. All important information is contained in a personal view. All important information can be found by browsing and bookmarking of important artifacts.

Target Object: Artifact: requirements

Type: Negative

Expression: Implicit

Application Domain: University: Software Engineering Course

Project size: The participants of this study were 13 students (9 computer science students and 4 economic science students with specialization in computer science) enrolled in a practical software engineering course

RE Practice: N/A

RE Phase: Elicitation, specification, documentation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: University of Kaiserslautern

Repeatability: First time questionnaire results

Lesson ID: LL013_12 [Gross and Doerr, 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: All artifact types were considered as being (very) important from the usability experts viewpoint with one exception: Specifications of interaction data and domain data were considered as being rather unimportant.

Source: Questionnaire

Rationale: Software requirements specifications play a crucial role in software development projects. Especially in large projects, these specifications serve as a source of communication and information for a variety of roles involved in downstream activities like architecture, design, and testing. In order to create high-quality requirements specifications that fit the specific demands of successive document stakeholders, we need to better understand the particular information needs of downstream development roles.

Impact: All information that is important for architecture design was available. All important information is contained in a personal view. All important information can be found by browsing and bookmarking of important artifacts.

Target Object: Artifact: requirements

Type: Negative

Expression: Implicit

Application Domain: Tutorial

Project size: The participants of this study were ten usability experts who attended a tutorial session conducted by one of the authors.

RE Practice: N/A

RE Phase: Elicitation, specification, documentation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time questionnaire results

Lesson ID: LL014_12 [Gross and Doerr, 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Usability engineers prefer relevant data to be specified together with the interaction descriptions (e.g., use case descriptions) rather than to be specified separately, e.g., in the form of a data model. visual representations of artifact types like goal descriptions, workflow descriptions are preferred over textual descriptions and that persona descriptions are very useful to support usability experts in their work.

Source: Questionnaire

Rationale: Different roles have different information needs.

Impact: The representation of the information is suitable for each role (e.g., UML diagrams) dependent on the activities.

Target Object: Artifact: requirements

Type: Negative

Expression: Implicit

Application Domain: Tutorial

Project size: The participants of this study were ten usability experts who attended a tutorial session conducted by one of the authors.

RE Practice: N/A

RE Phase: Specification, documentation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time questionnaire results

Lesson ID: LL015_12 [Gross and Doerr, 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Descriptions of supported stakeholders, information about their roles and responsibilities, frequent activities, typical work environment, age, gender, etc. is really helpful.

Source: Questionnaire

Rationale: Important information is missing in the SRS. Important information cant be found.

Impact: All information that is important for architecture design was available. All important information can be found by browsing and bookmarking of important artifacts.

Target Object: Artifact: requirements

Type: Negative

Expression: Implicit

Application Domain: Tutorial

Project size: The participants of this study were ten usability experts who attended a tutorial session conducted by one of the authors.

RE Practice: N/A

RE Phase: Elicitation, specification, documentation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time questionnaire results

Lesson ID: LL016_12 [Niknafs and Berry, 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: The effectiveness of a team in requirements idea generation is affected by the teams mix of domain familiarities.

Source: Controlled experiment

Rationale: The effectiveness of requirements engineering activities depends at least partially on the individuals involved. One of the factors that seems to influence an individuals effectiveness in requirements engineering activities is knowledge of the problem being solved, i.e., domain knowledge. While a requirements engineers having in-depth domain knowledge helps him or her to understand the problem easier, he or she can fall for tacit assumptions of the domain and might overlook issues that are obvious to domain experts.

Impact: Domain ignorance is effective in helping to generate at least two types of quality ideas, the relevant ideas and the feasible ideas.

Target Object: RE analysts

Type: Neutral

Expression: Implicit

Application Domain: University

Project size: Teams of size 3. Three members per team allows 4 types of teams: 3I: a team consisting of 3 DIs and 0 DAs, 2I: a team consisting of 2 DIs and 1 DAs, 1I: a team consisting of 1 DIs and 2 DAs, 0I: a team consisting of 0 DIs and 3 DAs.

RE Practice: N/A

RE Phase: Elicitation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: University of Waterloo

Repeatability: First time experimental results

Lesson ID: LL017_12 [Niknafs and Berry, 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: The effectiveness of a team in requirements idea generation is affected by the teams industrial experience.

Source: Controlled experiment

Rationale: The effectiveness of requirements engineering activities depends at least partially on the individuals involved. One of the factors that seems to influence an individuals effectiveness in requirements engineering activities is knowledge of the problem being solved, i.e., domain knowledge. While a requirements engineers having in-depth domain knowledge helps him or her to understand the problem easier, he or she can fall for tacit assumptions of the domain and might overlook issues that are obvious to domain experts.

Impact: Each team with 12 years of industrial experience performed a bit worse than any team with no industrial experience at all. Team performance in requirements idea generation drops dramatically for the teams with more than 2 years of industrial experience.

Target Object: RE analysts

Type: Neutral

Expression: Implicit

Application Domain: University

Project size: Teams of size 3. Three members per team allows 4 types of teams: 3I: a team consisting of 3 DIs and 0 DAs, 2I: a team consisting of 2 DIs and 1 DAs, 1I: a team consisting of 1 DIs and 2 DAs, 0I: a team consisting of 0 DIs and 3 DAs.

RE Practice: N/A

RE Phase: Elicitation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: University of Waterloo

Repeatability: First time experimental results

Lesson ID: LL018_12 [Lutz et al., 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Use Lamsweerdes goal-oriented requirements modeling to identify, specify, and analyze DNA nanodevices, and use PRISM model checking to verify both common properties across the family and properties that are specific to individual products.

Source: Illustrative example

Rationale: Challenges to doing requirements engineering in this domain include the error-prone nature of nanodevices carrying out their tasks in the probabilistic world of chemical kinetics, the fact that roughly a nanomole (a 1 followed by 14 0s) of devices are typically deployed at once, and the difficulty of specifying and achieving modularity in a realm where devices have many opportunities to interfere with each other.

Impact: Use of goal-oriented requirements engineering methods improved the modeling and analysis of requirements for a product family of sensing nanodevices by revealing new failure modes and facilitating reuse.

Target Object: Technique/method, tool

Type: Positive

Expression: Implicit

Application Domain: DNA nanotechnology

Project size: Product family of DNA nanodevices, i.e., a set of nanodevices that have a high degree of commonality and some key variations among them]. This product family is small, consisting initially of just three kinds of DNA origami pliers, nanomechanical devices developed to sense (detect) the presence of target molecules in a solution.

RE Practice: Use of PRISM model checker

RE Phase: Elicitation, analysis, validation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time application results

Lesson ID: LL019_12 [Cleland-Huang et al., 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Use Cleland-Huang's et al. race recommender system which pushes recommendations to project stakeholders as they create or modify traceable artifacts.

Source: Illustrative example and simulation

Rationale: In many software intensive systems traceability is used to support a variety of software engineering activities such as impact analysis, compliance verification, and requirements validation. However, in practice, traceability links are often created towards the end of the project specifically for approval or certification purposes. This practice can result in inaccurate and incomplete traces, and also means that traceability links are not available to support early development efforts.

Impact: Tracking trace obligations and generating trace recommendations throughout the active phases of a project can lead to early construction of traceability knowledge. Establishing lower threshold values for generating traceability links, creates more recommendations, results in traceability knowledge being constructed earlier in the project, delivers a higher percentage of useful recommendation events, but requires the developer to evaluate more traceability links per recommendation.

Target Object: Tool

Type: Positive

Expression: Implicit

Application Domain: Robotic system for supporting arm rehabilitation

Project size: The system included 6 hazards and 30 associated faults, a subset of which are depicted in Table II. It also included 40 functional requirements and 28 classes modeled in Enterprise Architect.

RE Practice: Tracing

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time simulation results

Lesson ID: LL020_12 [Knauss et al., 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Use Knauss' et al. auotmatic classifier to detect and classify clarification communication patterns.

Source: Evaluating case study

Rationale: In current project environments, requirements often evolve throughout the project and are worked on by stakeholders in large and distributed teams. Such teams often use online tools such as mailing lists, bug tracking systems or online discussion forums to communicate, clarify or coordinate work on requirements. In this kind of environment, the expected evolution from initial idea, through clarification, to a stable requirement, often stagnates. When project managers are not aware of underlying problems, development may pro- ceed before requirements are fully understood and stabilized, leading to numerous implementation issues and often resulting in the need for early redesign and modification.

Impact: Support managers in identifying requirements that exhibit problematic patterns of communication. Our method is not only intended for real- time application in a project to help diagnose requirements under development, but also to enhance project specific information. First, software practitioners can apply this method to analyze historical communication records and identify a catalogue of patterns in their project or organization. This may reveal particular communication practices that are process specific, or identify unexpected work practices. This information supports managers in decision-making with respect to supporting tools or process methodologies. Second, the application in real-time of our automatic pattern classifier to analyze current ongoing requirements discussions supports managers in examining the health of a requirement development based on the trajectory of clarification relative to other communication about the requirement. Project specific information such as development process, communication practices and tools allow the manager to decide whether the respective requirement does need attention and thus make timely decisions.

Target Object: Technique/method, tool

Type: Positive

Expression: Implicit

Application Domain: Software development environment

Project size: IBM Rational Team Concert (RTC) is a collaborative software development environment built on the JazzTMarchitecture. It integrates its own source-code control system, an issue-tracking system, and collaborative development tools. the RTC development team involved 151 active developers distributed over 16 different sites located in the United States, Canada, and Europe.

RE Practice: N/A

RE Phase: Analysis

Software Process: The project uses the agile and iterative Eclipse Way development process with six-week iteration cycles.

Project Date: N/A

Recording Date: N/A

Organisation Name: IBM

Repeatability: First time case study results

Lesson ID: LL021_12 [Tawhid et al., 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Model outcome-based regulations with the Goal-oriented Requirement Language

(GRL), generate questionnaires for the regional inspection activities where some of the questions target compliance and performance with a qualitative scale (rather than a binary assessment as in prescriptive regulations), and input the information collected by the inspection activities back into the model for compliance analysis.

Source: Industrial experience

Rationale: Ease the compliance burden on regulated parties and address some of the following issues: These regulations are about how things are to be done. Defining a prescriptive regime that applies equally well across a non-homogeneous group of regulated parties is challenging; Inspection activities assess compliance to specific regulations in a binary way (yes/no), which means it is hard to determine how close/far an organization is from being compliant, and how significant this is from a performance or effectiveness point of view.

Impact: This outcome-based approach is expected to help get a more precise understanding of who complies with what, while highlighting opportunities for improving existing regulatory elements. Improved accuracy. In an outcome-based context, a model much more adequately captures the interactions of potential solutions than natural language regulations. Furthermore, the model is an essential tool in discovering which measurements are needed to ensure compliance. OPF places requirements on goal-oriented modeling that requires changes to the GRL metamodel, introducing greater flexibility when converting real-world values into GRL evaluation values. a main lesson learned is that introducing a model-based requirement engineering process to regulation drafting requires cultural change from all involved stake- holders.

Target Object: Technique/method

Type: Negative

Expression: Implicit

Application Domain: Aviation transportation

Project size: N/A

RE Practice: N/A

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Transport Canada

Repeatability: First time case study results

Lesson ID: LL022_12 [Chernak, 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Use RCT (Requirements Composition Table) in software development projects for performing software change impact analysis for new releases.

Source: Industrial experience

Rationale: Commonly for complex applications, any particular team member may not have a complete knowledge and holistic view of the application functionality. For critical applications, a QA team commonly develops over time a sizable suite of regression tests and uses it for testing new releases. However, the project team frequently does not have a complete understanding of the test coverage provided by the regression suite and, specifically, where the coverage gaps are.

Impact: Understanding the impact of changes. Commonly, prior to using an RCT to perform change impact analysis the first time, the author asks developers and testers to perform this task the way that they have always done it and compile a list of impacted application features. The team then conducts a formal change impact analysis session using the RCT. The team compares the result of the formal RCT-driven analysis with the initial list of impacted features. The RCT-driven analysis frequently shows a much more complete (up to 50%) picture of the impact than does the initial list. Commonly, the initial list includes only core features and overlooks crosscutting concerns impacted by changes. Such a comparison is a good illustration of the RCT techniques effectiveness and can help the team to see its benefits.

Target Object: Technique/method

Type: Neutral

Expression: Explicit

Application Domain: Investment banks

Project size: RCT technique has been implemented for over a dozen Wall Street projects at three global investment banks. These projects included equity, fixed-income, and prime brokerage trading applications. There were three categories of sponsors of these engagements: 1. developers who needed to improve change impact analysis and better plan new releases, 2. testers who needed to assess coverage and identify gaps in their existing regression suites, and 3. business analysts, hired for renovation projects, who needed a holistic view of the legacy system to be replaced with the new application.

RE Practice: N/A

RE Phase: Analysis

Software Process: Traditional, use-case-driven or agile approaches.

Project Date: N/A

Recording Date: N/A

Organisation Name: Wall Street

Repeatability: First time case study results

Lesson ID: LL023_12 [**?**]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Categorize requirements in large systems engineering and integration project comprised of not only software but also hardware and activities in other disciplines into: Equipiment, Software, Dtandards, Functional, Constraint, Non-Functional, Location, Interface.

Source: Industrial experience

Rationale: The available set of requirement categories was not sufficient for our project.

Impact: Support the following roles and their responsibilities: Requirement engineers, System designers, Purchasing staff, Integration manager, Project managers, Compliance manager.

Target Object: Technique/method

Type: Negative

Expression: Explicit

Application Domain: System integration project that involves electrical, mechanical, civil, communication and power engineering.

Project size: The system consists of not only software, but also hardware such as HVAC equipment, fire detection and suppression, telephone switches, routers, CCTV cameras, PA speakers, ductwork and fiber cable. The software mainly focuses on monitoring and controlling of the equipment. The software development represents roughly 30% of the entire project effort. The hardware portion of the project task includes vendor selection, purchasing, installation, work site management, construction, and inspection. Another major characteristic of the project is that it is constrained by a large contract (i.e., contract requirements) which has about 4000 clauses.

RE Practice: N/A

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Siemens

Repeatability: First time case study results

<u>Lesson ID:</u> LL024_12 [**?**]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Do not perform the categorization on the contract requirements. Instead, we first clarified and refined the contract requirements into more proper technical requirements. Then we did the categorization on the technical requirements.

Source: Industrial experience

Rationale: The contract requirements are often not proper requirements in our project, the fact that a requirement can belong to multiple categories was a serious problem that led to much confusion.

Impact: Since we did not need to categorize the contract requirements in the new process, we also saved time reviewing those categorizations. In a contract-based project, such a review involves customers and many other stakeholders so the time savings is quite significant (e.g., 1-2 weeks). Another benefit from categorizing only technical requirements was that we avoided reviewing the consistencies between the categorizations of the contract requirements and the technical requirements. When using the old process, we needed to ensure the categorizations of technical requirements were consistent with the categorizations of the contract requirements.

Target Object: Technique/method

Type: Negative

Expression: Explicit

Application Domain: The system integration project we worked on involves electrical, mechanical, civil, communication and power engineering.

Project size: The system consists of not only software, but also hardware such as HVAC equipment, fire detection and suppression, telephone switches, routers, CCTV cameras, PA speakers, ductwork and fiber cable. The software mainly focuses on monitoring and controlling of the equipment. The software development represents roughly 30% of the entire project effort. The hardware portion of the project task includes vendor selection, purchasing, installation, work site management, construction, and inspection. Another major characteristic of the project is that it is constrained by a large contract (i.e., contract requirements) which has about 4000 clauses.

RE Practice: N/A

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Siemens

Repeatability: First time case study results

Lesson ID: LL025_12 [Berenbach et al., 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Use the Unified Requirements Modeling Language (URML) for requirements modelling during elicitation.

Source: Industrial experience

Rationale: Customer processes and product requirements were sufficiently complex to warrant modeling. During modeling sessions however problems were experienced with existing modeling languages and accompanying tools. Inability to connect products, product lines, features, and design models. Inability to connect goals, features, and requirements. Inability to capture hazards and threats in a model during early requirements elicitation. Lack of clear delineation of customer processes and system use cases, and overloading of the rectangle.

Impact: Integration of goals, features, and requirements enables resolution of conflicts before implementation. Integration of product line, features, and requirements helps with impact analysis of product line variation points. Integration of processes and dangers helps with identification of error prone processes. The language should support control and information flow. Using an UML profile might not be the optimal way to enable tool support for a domain-specific modeling language.

Target Object: Technique/method

Type: Negative

Expression: Implicit

Application Domain: Energy, software platform, mail sorting system for the U.S. Postal Service (USPS) (not exhaustive)

Project size: 6 core requirements from different projects conducted at Siemens

RE Practice: Modeling

RE Phase: Elicitation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Siemens

Repeatability: First time case study results

Lesson ID: LL026_12 [Mendizabal et al., 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Use Mendizabal's et al. practical approach for requirements elicitation and prioritization based on realistic user behaviors observation.

Source: Industrial experience

Rationale: N/A

Impact: Reduced the number of meetings with stakeholders and helped achieving consensus on prioritized requirements easily and effortlessly. Another important finding was that groups of stakeholders (located in different countries and assuming different responsibilities to the application) agreed with most requirements proposed.(i) reduction on time spent arguing internally over individual requirements; (ii) prevention of rigid and long negotiations with stakeholders and project team; (iii) conception of a preliminary requirements list that serves as a guide for reaching consensus among stakeholders on strategic business value; (iv) production of succinct documentation and artifacts that confer transparency to the requirements elicitation and prioritization process.

Target Object: Technique/method

Type: Negative

Expression: Implicit

Application Domain: The web based application is responsible for managing distributions packages with related material for the whole line of products of the company. Those distributions packages contain hardware drivers, operating systems, or any content related to a specific product (eg. laptop, server, etc).

Project size: Several features are available and different teams work on very specific parts of the system. In terms of application usage, the majority of accesses come from Asia and America. There is also a regular use from users located in Europe, but in minor scale. Due to the wide geographic spectrum of users and differences in timing zones, the application does not present idle periods of usage. There are about 1200 users accessing the application per day, and approximately 130 users accessing it per hour. During working days the minimum number of concurrent users observed was around 50 and the maximum number reached 231 users at the same hour.

RE Practice: N/A

RE Phase: Elicitation, Prioritisation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Dell

Repeatability: First time case study results

Lesson ID: LL027_12 [Hauksdottir et al., 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Use the adjustable requirements reuse approach where requirements can be adjusted without independently creating and storing the different variants of the requirement.

Source: Industrial experience

Rationale: In direct reuse the focus is on keeping this relationship and reusing requirements exactly as they are. To enable this method, it is necessary to separate requirements into parts that allow each project to select exactly the content and variation it needs. When working in a project, if the user would like to adjust an existing requirement or enter a variation of it, he or she shall adjust the company requirement or create a new variable part in the company repository, and then map it to the project. However, the users tend not to do this. In practice, they only add new project specific requirements that do not end up being added to the company requirements.

Impact: Using the adjustable approach the quality and readability of each requirement is improved since it is not split up to a general and a variable part. Since it is not required to document every variation in a separate node it keeps the structure of the requirements much more simple. Furthermore, not being bound by a rigorous change process should enable the users to make continuous improvements to the requirements. The main result is the second project was able to reuse in total over 80% of the requirements they documented. 43% of the requirements are directly reused but 38% are reused with adjustments. By allowing adjustable reuse we enable a higher level of requirement reuse than would have been possible through previous practice. This approach was found to be effective when the domain maturity is low and the significant set of requirements were changed from project to project.

Target Object: Technique/method

Type: Negative

Expression: Implicit

Application Domain: Solar

Project size: The first project to document requirements, documented about 530 requirements. The RS was ready in 21 weeks and it is estimated that around 6000 person hours were used for the job.

RE Practice: Reuse

RE Phase: Elicitation, Documentation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Danfoss Solar Inverters

Repeatability: Industry results

Lesson ID: LL028_12 [Hauksdottir et al., 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Use the Direct Requirements Reuse approach for requirements elicitation.

Source: Industrial experience

Rationale: When Word documents were "clone&owned, it was easy to fix a little here and there and everywhere in specifications, code, tests, but no cross- product compatibility could be guaranteed.

Impact: Reusing requirements without adjustments gives a clearer definition of new products. With present focus, each feature is evaluated, and only the defined, value- adding features are updated. Separating the general part and the defined variance of the product family, enables building specific dependencies in the reusable requirement model, identifying how a specific requirement variant is related to another. This allows for capturing engineering knowledge into the model and prevents users from making invalid selections. Direct reuse furthermore enabled structured coding and testing. When SW requirements are reused, the underlying architecture and code can also be reused, and it makes more sense to automate unit- and system-testing, when the same asset can be used untouched for a number of products. In other words, for the same effort, more features and test coverage can be obtained. Based on our experience, one can achieve much faster time to market and more reliable quality as a direct result of the approach taken. This approach allows high reuse potential and significant savings for stable domains, where most requirements tend to be small additions or minor changes of existing requirements.

Target Object: Technique/method

Type: Negative

Expression: Implicit

Application Domain: Drives

Project size: Four technology experts in different fields were stationed in Beijing and local staff were hired in for the twofold task of creating a new product family and building up an independent development site.

RE Practice: Reuse

RE Phase: Elicitation, Documentation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Danfoss Drives

Repeatability: Industry results

Lesson ID: LL029_12 [Hauksdottir et al., 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Define a process for maintaining and cleaning up legacy requirements by documenting the rationale for each requirement.

Source: Industrial experience

Rationale: It is difficult to evaluate the relevance of requirements, without knowledge of why they were originally introduced.

Impact: Ensures backward compatibility for all products in the drives domain.

Target Object: Technique/method

Type: Negative

Expression: Implicit

Application Domain: Solar, drives

Project size: N/A

RE Practice: N/A

RE Phase: Specification, Documentation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Danfoss

Repeatability: Industry results

Lesson ID: LL030_12 [Hauksdottir et al., 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Structure the company repository so that some requirements can be shared by the whole company or business segment, whereas others are only shared between similar product lines.

Source: Industrial experience

Rationale: Some requirements are product or business area specific, whereas others are company- or even world-wide defined and updated. Examples could be reliability measures and certification versus domain specific application and features.

Impact: Easier sharing of requirements.

Target Object: Technique/method

Type: Neutral

Expression: Explicit

Application Domain: Solar, drives

Project size: N/A

RE Practice: N/A

RE Phase: Documentation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Danfoss

Repeatability: Industry results

<u>Lesson ID:</u> LL031_12 [Hauksdottir et al., 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Handle variability by breaking larger requirements into smaller pieces.

Source: Industrial experience

Rationale: N/A

Impact: Fine- grained requirements allow facilitating a large range of variability, but are hard to maintain.

Target Object: Technique/method

Type: Neutral

Expression: Explicit

Application Domain: Solar, drives

Project size: N/A

RE Practice: N/A

RE Phase: Specification, documentation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Danfoss

Repeatability: Industry results

<u>Lesson ID:</u> LL032_12 [Hauksdottir et al., 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Do not model all the constraints into the requirements model.

Source: Industrial experience

Rationale: Many of the possible combinations of requirements are illegal in a way that such combination does not make any sense in terms of the application domain, marketing or even basic physics.

Impact: If one would model all the constraints so that only current products can be derived from the model, it would become fragile, and break when facing evolution. This would significantly increase the evolution cost for a limited derivation time benefit.

Target Object: Technique/method

Type: Negative

Expression: Implicit

Application Domain: Solar, drives

Project size: N/A

RE Practice: Modeling

RE Phase: Specification, documentation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Danfoss

Repeatability: Industry results

Lesson ID: LL033_12 [Hauksdottir et al., 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Tailor the requirements reuse approach to the application domain.

Source: Industrial experience

Rationale: The volatility of the domain has a profound effect on how requirements reuse initiatives should be done.

Impact: N/A

Target Object: Technique/method

Type: Positive

Expression: Explicit

Application Domain: Solar, drives

Project size: N/A

RE Practice: Reuse

RE Phase: Elicitation, Documentation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Danfoss

Repeatability: Industry results

Lesson ID: LL034_12 [Kukreja et al., 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Use Kukreja's et al. approach for selecting an appropriate requirements prioritization framework.

Source: Industrial experience

Rationale: Simple techniques do not lend themselves well to incorporating change requests i.e. when a new change request is received, it is difficult to ascertain its value and the need for incorporating it within the current release. Requirements engineers usually engage in extended negotiations with the client to understand the true priority of the requirement. There is no model to insert a requirement and see how it compares to the existing ones, to better understand its true priority and channelize the negotiation effort accordingly.

Impact: Satisfies the need for standardizing value centric prioritization practices across the organization.

Target Object: Technique/method

Type: Negative

Expression: Implicit

Application Domain: IT

Project size: Conducted interviews with 50 stakeholders ranging from business analysts to project, program and portfolio managers.

RE Practice: Use of a prioritisation framework

RE Phase: Prioritisation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: Industry results

Lesson ID: LL035_12 [Kukreja et al., 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: Factoring requirement dependencies in the prioritization to closely resemble implementation order is important.

Source: Industrial experience

Rationale: Without the ability of handling prerequisites the tool was deemed unusable by the requirements engineers.

Impact: N/A

Target Object: Technique/method

Type: Negative

Expression: Explicit

Application Domain: IT

Project size: Deployed for use by an independent testing team (for value based test case prioritization), business analysts of various projects and an independent product development team in the company.

RE Practice: Use of a prioritisation framework

RE Phase: Prioritisation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: Industry results

Lesson ID: LL036_12 [Kukreja et al., 2012]

Journal: N/A

Conference: IEEE RE

Workshop: N/A

Year: 2012

Lesson: A need for hierarchical prioritisation since it fit the mental model of the product team.

Source: Industrial experience

Rationale: N/A

Impact: Analysts would perform top-down decomposition from goals, to modules, to requirements where requirement priorities were influenced by the module it belonged to and the goal(s) it helped satisfy.

Target Object: Technique/method

Type: Positive

Expression: Explicit

Application Domain: IT

Project size: Deployed for use by an independent testing team (for value based test case prioritization), business analysts of various projects and an independent product development team in the company.

RE Practice: Use of a prioritisation framework

RE Phase: Prioritisation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: Industry results

<u>Lesson ID:</u> LL037_12 [Gross et al., 2012]

Journal: N/A

Conference: N/A

Workshop: EmpiRE

Year: 2012

Lesson: ACT (UML Activity Diagrams) are more effective and efficient than EPC (Event-driven Process Chains) from a requirements engineers viewpoint.

Source: Evaluating Experiment

Rationale: N/A

Impact: N/A

Target Object: Tool

Type: Neutral

Expression: Implicit

Application Domain: University

Project size:N/A

RE Practice: N/A

RE Phase: Analysis, documentation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time experimental results

<u>Lesson ID:</u> LL038_12 [Bjarnason et al., 2012]

Journal: N/A

Conference: N/A

Workshop: EmpiRE

Year: 2012

Lesson: The timeline (evidence-based timelines visualizing the project history) and its overview enabled a discussion of the whole life cycle including RE decisions made through-out and supports the project retrospective meeting to a high degree.

Source: Questionnaire

Rationale: Project retrospectives may support project teams in reflecting on how requirements are agreed upon and communicated throughout a project. However, time is rarely taken for group reflection after project completion. Furthermore, project events may be recalled differently due to memory bias.

Impact: The evidence- based timelines may act as integrators at the meetings and thereby create an environment productive to constructive reflection and sharing, similarly to the usage of whiteboards and post-its. Timelines are beneficial in providing a common background that motivates team members without previous information about the full development cycle into deeper analysis, thereby supporting reflection and observations of patterns at the project level.

Target Object: Tool

Type: Negative

Expression: Implicit

Application Domain: N/A

Project size:Three feature projects that had developed new functionality and delivered software to a release project within the past few months were selected for the evaluation. The respondents represent all roles present at the retrospective meetings, i.e. product manager (requirements responsible), project manager, line manager, architect, developer and tester.

RE Practice: N/A

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: Confirming previous findings

Lesson ID: LL039_12 [Morales-Ramirez et al., 2012]

Journal: N/A

Conference: N/A

Workshop: EmpiRE

Year: 2012

Lesson: Interviews to stakeholders play a major role as a technique for the identification of early requirement elements.

Source: Retrospective case study analysis

Rationale: N/A

Impact: N/A

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Socio-technical system in the ambient assisted living domain (ACube (Ambient Aware Assistance) project)

Project size: Project documentation consists of: the Carta dei Servizi document and transcription of 8 interviews (Int.1-8); spreadsheets containing lists of early-requirements elements with links to other documents; textual and visual descriptions of criticalities and scenarios; and GO models (initial and final Tropos early requirement models). Besides this, two ACube analysts (authors of this paper) are available to help understand projects material. 80 system requirements, of which 57 are functional.

RE Practice: Interviews

RE Phase: Elicitation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Social Residence

Repeatability: This confirms what is stated by Dieste et al., that interviewing is the most effective elicitation technique in collecting information about the domain.

Lesson ID: LL040_12 [Massacci et al., 2012]

Journal: N/A

Conference: N/A

Workshop: EmpiRE

Year: 2012

Lesson: The modeling supported by Massaci's et al. approach can be a useful decision support tool for decision makers during brainstorming and change assessment.

Source: Study within the research group, case study with master students, workshop with ATM experts

Rationale: The evolution of mission-critical requirements at enterprise level is known to be possible, but it is unknown whether it would happen: the known unknowns.

Impact: To capture what identified as the knowledge shared by multiple stakeholders about "where the enterprise is currently, "where the enterprise wishes to be in the future, and "what alternative designs are needed for the desired future state.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Air Traffic Management (ATM) domain

Project size: N/A

RE Practice: Modeling

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Deep Blue Srl

Repeatability: Confirming previous findings

Lesson ID: LL041_12 [Hussain et al., 2012]

Journal: N/A

Conference: N/A

Workshop: EmpiRE

Year: 2012

Lesson: Use LASR for annotating requirements document.

Source: Evaluating Experiment

Rationale: Annotation of software requirements documents is performed by experts during the requirements analysis phase to extract crucial knowledge from informally written textual requirements. Different annotation tasks target the extraction of different types of information and require the availability of experts specialized in the field. Large scale annotation tasks require multiple experts where the limited number of experts can make the tasks overwhelming and very costly without proper tool support.

Impact: Help in attaining more accurate annotations,and helped eliminating the need of running adjudication sessions to resolve disagreement among the annotators, and, thus, reducing the cost of large scale annotation.

Target Object: Tool

Type: Positive

Expression: Implicit

Application Domain: Business, academic, web

Project size: Six requirements documents belonging to three different problem domains. They were collected from both the industry and academia. We had two groups of annotators and one expert (in requirements annotation) annotating the above documents. One group (G1) consisting of four graduate students of the Master of Computer Science program who were trained to annotate software requirements documents manually. The expert led the training of the annotators, and also participated with them in the manual annotation experiment. The other group (G2) consisted of 26 undergraduate students of software engineering.

RE Practice: Annotation

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: SAP Labs, Montreal, Canada, Concordia University

Repeatability: First time experimental results

Lesson ID: LL042_12 [Smialek et al., 2012]

Journal: N/A

Conference: N/A

Workshop: MoDRE

Year: 2012

Lesson: Use Smialek's et al. approach of using RSL to convert requirements from use cases to working code.

Source: Evaluating Experiment

Rationale: Use cases are used in many methodologies to drive the software engineering process. Though, their transition to code was usually a mostly manual process. In the context of

MDD, use cases gain attention as first-class artifacts with representation notations allowing for automatic transformations to analysis and design models.

Impact: Scenarios gain precise runtime semantics. They can be fully automatically translated into dynamic executable code. The transformation can be easily changed to produce code in any object-oriented language and any architectural framework. significant gains in productivity.

Target Object: Tool

Type: Positive

Expression: Implicit

Application Domain: University

Project size: 28 post- graduate CS students attending a course on Model-Driven Software Development. they were formed into 8 groups consisting of 3-4 students each. All the groups were assigned a ready use case model of a Campus Management System, containing 12 use cases.

RE Practice: Modeling

RE Phase: Specification

Software Process: Agile

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time experimental results

Lesson ID: LL043_12 [Teka et al., 2012]

Journal: N/A

Conference: N/A

Workshop: MoDRE

Year: 2012

Lesson: When using goal change management techniques, the Fuzzy logic based reasoning provides concrete values for more detailed goal analysis and can handle aggregation of multiple contribution types while the TROPOS approach is suitable in providing high level goal analysis and handling of goal conflicts.
Source: Evaluating case study
Rationale: Common Enterprise Architecture (EA) frameworks like The Open Group Architecture Framework (TOGAF) and EA modeling languages like Archimate lacks support for analyzing goal and requirement change impacts in EA goal models.
Impact: To obtain a change impact analysis algorithm for managing evolving stakeholder goals in Enterprise Architecture (EA) designs.
Target Object: Technique/method
Type: Neutral
Expression: Explicit
Application Domain: Drinking water production organization
Project size: Ninety goals identified from relevant documents and interviews.
RE Practice: N/A
RE Phase: Analysis
Software Process: N/A
Project Date: N/A
Recording Date: N/A
Organisation Name: BiZZdesign B.V.
Repeatability: First time case study results

Lesson ID: LL044_12 [Torres et al., 2012]
Journal: N/A
Conference: N/A
Workshop: MoDRE

Year: 2012

Lesson: Use Torres' et al. approach to mitigate to obsolescence of quality specification models in service based systems.

Source: Evaluating case study

Rationale: Current approaches do not support updating of the specification to reflect changes in the service market, like newly available services or improved QoS of existing ones. Thus, even if the specification models reflect design-time acceptable requirements they may become obsolete and miss opportunities for system improvement by self-adaptation.

Impact: Allow engineers to build SBS that can be protected against market-caused obsolescence of their requirements specifications.

Target Object: Tool

Type: Positive

Expression: Implicit

Application Domain: Service-based systems

Project size: A set of ten case studies, each one was composed of several software requirements, which them- selves were constrained by multiple quality requirements.

RE Practice: N/A

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

Lesson ID: LL045_12 [Berenbach, 2012]

Journal: N/A

Conference: N/A

Workshop: MoDRE

Year: 2012

Lesson: Modeling did not help with non- functional requirements, they had to be treated separately, either as annotation (e.g. notes on diagrams), separate documentation or with mathematical simulations.

Source: Industrial experience

Rationale: N/A

Impact: N/A

Target Object: Technique/method

Type: Negative

Expression: Explicit

Application Domain: Control systems and simulators for chemical and power plants

Project size: N/A

RE Practice: Modeling

RE Phase: Specification, documentation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: 25 year retrospective

Lesson ID: LL046_12 [Berenbach, 2012]

Journal: N/A

Conference: N/A

Workshop: MoDRE

Year: 2012

Lesson: SysML solves the problem of including non- functional requirements in an analysis model by incorporating requirements and constraints as first class modeling concepts.

Source: Industrial experience

Rationale: N/A

Impact: N/A

Target Object: Technique/method

Type: Positive

Expression: Explicit

Application Domain: Information technology (IT) consulting

Project size: N/A

RE Practice: Modeling

RE Phase: Specification, documentation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: 25 year retrospective

Lesson ID: LL047_12 [Berenbach, 2012]

Journal: N/A

Conference: N/A

Workshop: MoDRE

Year: 2012

Lesson: When eliciting requirements, it is impractical to switch between goal, product line, threat and process modeling tools during elicitation. By having all the concepts available in a single toolbar, it is possible to indicate hazards and mitigations early in the requirements capture process.

Source: Industrial experience

Rationale: N/A

Impact: N/A

Target Object: Tool

Type: Negative

Expression: Explicit

Application Domain: N/A

Project size: Two projects

RE Practice: Modeling

RE Phase: Specification, documentation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Siemens

Repeatability: 25 year retrospective

Lesson ID: LL048_12 [Berenbach, 2012]

Journal: N/A

Conference: N/A

Workshop: MoDRE

Year: 2012

Lesson: Using symbols that represent their concepts improves communications with domain experts and other non-technical stakeholders.

Source: Industrial experience

Rationale: N/A

Impact: N/A

Target Object: Technique/method

Type: Positive

Expression: Explicit

Application Domain: N/A

Project size: Two projects

RE Practice: Modeling

RE Phase: Specification, documentation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Siemens

Repeatability: 25 year retrospective

<u>Lesson ID:</u> LL049_12 [Berenbach, 2012]

Journal: N/A

Conference: N/A

Workshop: MoDRE

Year: 2012

Lesson: A productivity improvement in requirements capture of from 30-60% with MDRE as opposed to textual requirements capture.

Source: Industrial experience

Rationale: N/A

Impact: N/A

Target Object: Tool

Type: Positive

Expression: Explicit

Application Domain: N/A

Project size: N/A

RE Practice: Modeling

RE Phase: Specification, documentation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: 25 year retrospective

Lesson ID: LL050_12 [Berenbach, 2012]

Journal: N/A

Conference: N/A

Workshop: MoDRE

Year: 2012

Lesson: When requirements are captured with models, for a variety of reasons it is necessary to maintain them in hierarchical databases.

Source: Industrial experience

Rationale: Because of the impedance mismatch between model structure (directed graph) and requirements database (tree structure).

Impact: N/A

Target Object: Technique/method

Type: Negative

Expression: Explicit

Application Domain: N/A

Project size: N/A

RE Practice: Modeling

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: 25 year retrospective

Lesson ID: LL051_12 [**?**]

Journal: N/A

Conference: N/A

Workshop: MoDRE

Year: 2012

Lesson: Use Mallet's paper approach to improve specification quality through systemization of specification structures based on architectural block diagrams, behavioural statecharts and propositional logic structures.

Source: Evaluating case study

Rationale: In practice the elicitation and specification of requirements remained largely unaffected by the introduction of model- based methods, while much effort has been spent on the introduction of functional specifications. As a consequence, where functional specifications exist, these are often created unsystematically and are of poor quality, leading to further problems during design, implementation and testing.

Impact: The resulting specifications provide connection points to sub-sequent model-based analysis, design and testing activities, such as sequence enumeration, model-based testing or model checking.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Car-manufacturer

Project size: 10 specification projects. Among the specified driver assistance systems are 4 parking systems, 2 break-assistance systems, and a camera- based traffic information system.

RE Practice: Modeling

RE Phase: Elicitation, specification

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: Results from 3 years

Lesson ID: LL052_12 [Merten et al., 2012]

Journal: N/A

Conference: N/A

Workshop: REET

Year: 2012

Lesson: Use Merten's et al. web-based software-based feedback agents (SBFA) system that uses reasoning to assists the user.

Source: Evaluating Experiment

Rationale: The students have to use an issue tracking software in combination with a Requirements Engineering (RE) tool to document and plan their work. Though the course starts with RE theory from elicitation via documentation and traceability, we found that the students find it difficult to combine different RE artifact types and to develop useful traces between them.

Impact: Provide feedback and give pro-active advice inside an RE tool, while the specification is created.

Target Object: Tool

Type: Positive

Expression: Implicit

Application Domain: University

Project size: 8 students

RE Practice: N/A

RE Phase: Elicitation, specification

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Bonn-Rhine-Sieg University

Repeatability: First time questionnaire results

Lesson ID: LL053_12 [Braun et al., 2012]

Journal: N/A

Conference: N/A

Workshop: RELAW

Year: 2012

Lesson: Natural Language before Performance Model does not impact the quality of regulations or only very late in the regulatory process.

Source: Industrial experience

Rationale: The status quo is that performance modeling is not routinely included in the regulatory process, which may lead to lack of clarity, inconsistencies, and difficulties measuring and hence assessing compliance.

Impact: There is no or little improvement in terms of inconsistencies in the regulations, the understandability of the regulations, and the measurability of desired regulation outcomes. It is likely that some measures will be difficult to quantify, leading to problems for effectively enforcing the regulations.

Target Object: Technique/method

Type: Negative

Expression: Implicit

Application Domain: Aviation Security

Project size: Transport Canada is engaged in a multi-year modernization process to review and renew its Aviation Security regulations

RE Practice: Modeling

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Transport Canada

Repeatability: First time industrial experience

<u>Lesson ID:</u> LL054_12 [Braun et al., 2012]

Journal: N/A

Conference: N/A

Workshop: RELAW

Year: 2012

Lesson: Use Performance Model before Natural Language approach for drafting of regulations as the performance model is translated into natural language.

Source: Industrial experience

Rationale: The status quo is that performance modeling is not routinely included in the regulatory process, which may lead to lack of clarity, inconsistencies, and difficulties measuring and hence assessing compliance.

Impact: This approach is that it is inherently outcome-focused, as the structure of the performance model makes the regulatory team (i) focus on the desired outcome long before regulations are written, (ii) think of exceptions and conditions, and (iii) decide on what and how the outcome should be measured with indicators. Indicators will be measurable, hence enabling the regulator to effectively enforce regulations. The performance model gives a global, holistic view of all regulations, improves understanding of the relationships of various regulations, and therefore makes it easier to evolve regulations whether they need to changed, retracted, or added. The focus on the whole and not only on a single regulatory element alleviates the barrier to removing regulations in fear of unintended consequence, because it is now possible to demonstrate the effect of removing one regulation with the performance model. Models also make it easier to document the evolution of regulations.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Aviation Security

Project size: Transport Canada is engaged in a multi-year modernization process to review and renew its Aviation Security regulations

RE Practice: Modeling

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Transport Canada

Repeatability: First time industrial experience

Lesson ID: LL055_12 [Braun et al., 2012]

Journal: N/A

Conference: N/A

Workshop: RELAW

Year: 2012

Lesson: Use the Natural Language and Performance Model at the Same Time approach for drafting of regulations.

Source: Industrial experience

Rationale: The status quo is that performance modeling is not routinely included in the regulatory process, which may lead to lack of clarity, inconsistencies, and difficulties measuring and hence assessing compliance.

Impact: This iterative approach allows for continued verification that the modelers interpretation of the regulations is in line with the intent of the authors of the regulations. Furthermore, it is possible to verify that the desired outcome is indeed measurable as indicators are explicitly modeled. The performance model aids in drafting regulations and vice versa.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Aviation Security

Project size: Transport Canada is engaged in a multi-year modernization process to review and renew its Aviation Security regulations

RE Practice: Modeling

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Transport Canada

Repeatability: First time industrial experience

Lesson ID: LL056_12 [Braun et al., 2012]

Journal: N/A

Conference: N/A

Workshop: RELAW

Year: 2012

Lesson: Use Performance Model Only approach for drafting of regulations.

Source: Industrial experience

Rationale: The status quo is that performance modeling is not routinely included in the regulatory process, which may lead to lack of clarity, inconsistencies, and difficulties measuring and hence assessing compliance.

Impact: There is no need to maintain two versions of the same regulations.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Aviation Security

Project size: Transport Canada is engaged in a multi-year modernization process to review and renew its Aviation Security regulations

RE Practice: Modeling

RE Phase: N/A

Software Process: Iterative approach

Project Date: N/A

Recording Date: N/A

Organisation Name: Transport Canada

Repeatability: First time industrial experience

Lesson ID: LL057_12 [Sapkota et al., 2012]

Journal: N/A

Conference: N/A

Workshop: RELAW

Year: 2012

Lesson: Use the extended version of the Semantic-ART framework to extract essential meaning from a regulatory text.

Source: Evaluating case study

Rationale: Extraction of meaningful text from regulatory guidelines comes with many research challenges such as dealing with different document-format, implicit document-structure, textual ambiguity and complexity

Impact: Extracting essential meaning from the regulatory text helps in the automation of the Compliance Management (CM) process. The annotation generated by the tool was compared with the manually created annotation of the same text, and the result was very close to the manual annotation.

Target Object: Tool

Type: Positive

Expression: Implicit

Application Domain: Pharmaceutical industry

Project size: 50 regulatory sentences were selected from the Eudralex-5

RE Practice: Annotation

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Eudralex

Repeatability: First time case study results

<u>Lesson ID:</u> LL058_12 [**?**]

Journal: N/A

Conference: N/A

Workshop: RELAW

Year: 2012

Lesson: Requirements engineers can benefit from applying (commitment, privilege, and right) analysis (CPR) analysis rather than goal-based analysis or non-method-assisted analysis to produce compliance requirements.

Source: Experiment

Rationale: In the United States, organizations can be held liable by the Federal Trade Commission for the statements they make in their privacy policies. Thus, organizations must include their privacy policies as a source of requirements in order to build systems that are policy-compliant.

Impact:The CPR subjects produced a higher median number of expected compliance requirements requirements derived from the policy by the experimenter (the first author) and two other requirements engineers against which we compare the subject-produced requirements. The requirements produced by the CPR subjects had better correctness and completeness with respect to expected compliance requirements than the requirements produced by goal-based and control subjects.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: University

Project size: There were twelve subjects in the CPR condition group and eleven subjects in the goal-based and control condition groups, respectively. The majority of the subjects (30 - CPR: 11, GB: 10, control: 9) were computer science majors. Most subjects (24 - 8, 9, 7) were pursuing Masters degree. The youngest subject was 19 years old, and the oldest subject was 49 years old. The median ages for the CPR, goal-based, and control subjects were 23, 23, and 22, respectively. Most subjects (23 - 8, 6, 9) were male. It was a one-page portion of a legitimate policy from a popular social networking site, Facebook. This portion of the policy discussed the information that the organization receives, including: information about the user; content; transactional information; site activity information; access device and browser information; and cookie information.

RE Practice: N/A

RE Phase: Elicitation, analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: North Carolina State University (NCSU)

Repeatability: First time experimental results

Lesson ID: LL059_12 [Sharma and Biswas, 2012]

Journal: N/A

Conference: N/A

Workshop: RePa

Year: 2012

Lesson: Use the norm analysis patterns for automated requirements validation.

Source: Evaluating case study

Rationale: Requirements validation is an integral activity of Requirements Engineering. An early detection of mismatch between the observable behaviour of the real-world and the interpreted behaviour of the information system after requirements analysis is essential to the success of the software developed.

Impact: Earlier transformation to norms improved the understanding of the requirements of the information system.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Student registration, grading process, library

Project size: N/A

RE Practice: Patterns

RE Phase: Validation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

Lesson ID: LL060_12 [Sharma and Biswas, 2012]

Journal: N/A

Conference: N/A

Workshop: RePa

Year: 2012

Lesson: When the norms were subjected to reasoning and inferencing, it was found that both these norms have similar consequent and different antecedent. Such a situation should be verified from stakeholders before moving onto the next stage of software design.

Source: Pilot study

Rationale: N/A

Impact: N/A

Target Object: Technique/method

Type: Negative

Expression: Implicit

Application Domain: Library

Project size: Four subjects

RE Practice: Patterns

RE Phase: Validation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time pilot study results

<u>Lesson ID:</u> LL061_12 [Sharma and Biswas, 2012]

Journal: N/A

Conference: N/A

Workshop: RePa

Year: 2012

Lesson: Norms themselves cannot always uncover underlying assumptions.

Source: Pilot study

Rationale: N/A

Impact: N/A

Target Object: Technique/method

Type: Negative

Expression: Implicit

Application Domain: Library

Project size: Four subjects

RE Practice: Patterns

RE Phase: Validation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time pilot study results

<u>Lesson ID:</u> LL062_12 [Behnam et al., 2012]

Journal: N/A

Conference: N/A

Workshop: RePa

Year: 2012

Lesson: Use the Goal-oriented Pattern Family (GoPF) framework with 'indicators' to create a pattern family.

Source: Case study

Rationale: As regulators start evolving regulations towards an outcome- based approach, it becomes important to reuse knowledge about existing problems and solutions, and patterns are known to be a means of increasing reusability.

Impact: The aviation screening pattern family captures knowledge about problems and solutions at a given time. This helps regulatory parties by enabling the reuse of goal and business process model building blocks on one hand and by shedding light on rationales of past decisions on the other hand. stakeholders confirmed that patterns in such a family are valuable vessels for reusing the knowledge for goal and business process modeling in other areas of screening. it is also expected that the knowledge captured here can be potentially reused in other similar domains such as aviation safety or non-aviation screening domains.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Aviation security regulatory compliance

Project size: Three different regulation units: passenger, carry-on baggage, and hold-baggage screening.

RE Practice: Patterns

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Transport Canada

Repeatability: First time case study results

Lesson ID: LL063_12 [Daramola et al., 2012]

Journal: N/A

Conference: N/A

Workshop: RePa

Year: 2012

Lesson: Use the ReqSec tool to aid RA in writing security requirements.

Source: Evaluating Experiment

Rationale: The task of specifying and managing security requirements (SR) is a challenging one. Usually SR are often neglected or considered too late leading to poor design, and cost overruns. Also, there is scarce expertise in managing SR, because most requirements engineering teams do not include security experts, which leads to prevalence of too vague or overly specific SR.

Impact: High rating for perceived ease of use, and serendipity - generally demonstrates the potential of the tool to first, simplify, and significantly aid the RA during the SRS process, particularly when the RA is not highly experienced. Second, facilitate a reduction in the effort expended on SRS, particularly as the process progresses by enabling reuse of most frequently-used boilerplate patterns for new SR formulations. Third, ensure that quality SR are formulated, in a consistent way, and without ambiguity.

Target Object: Tool

Type: Positive

Expression: Implicit

Application Domain: University

Project size: 7 subjects. Master degree students of software engineering

RE Practice: Patterns

RE Phase: Elicitation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Norwegian University of Science and Technology, Covenant University

Repeatability: First time experimental results

Lesson ID: LL064_12 [Penzenstadler and Eckhardt, 2012]

Journal: N/A

Conference: N/A

Workshop: RESS

Year: 2012

Lesson: Use a requirements engineering content model that serves as reference for requirements elicitation and documentation on the different levels of abstraction (SoS level as well as single system level).

Source: Case study

Rationale: Requirements engineering for systems of systems faces extremely distributed requirements engineering activities that involve a multitude of stakeholders, while the surrounding SoS is not necessarily in the focus of single system developers working on small units within the SoS. This often results in isolated requirements engineering approaches which, in turn, lead to requirements that can hardly be integrated with the other units of the SoS in order to keep them consistent. Furthermore, problems arise with incomplete and/or redundant contents, consistency, and traceability.

Impact: Instantiated in a concrete artefact model, contents are documented in a traceable way while at the same time ensuring coverage of an agreed set of contents. The benefits are consistency within the requirements and eased communication between the stakeholders. The content model has led to common wording and efficient collaboration between the partners and across the domains, thereby showing first indicators of successfully approaching some of the challenges of requirements engineering for systems of systems. The application of an artefact model eases communication, improves consistency, and provides traceability of the contents.

Target Object: Tool

Type: Positive

Expression: Implicit

Application Domain: Systems of Systems (SoS). ARAMiS (Automotive, Railway, and Avionics in Multicore Systems). aims to create the technological basis to further improve security, traffic-efficiency, and comfort in the mobility domains automotive, avionic, and railway by utilizing multi-core technologies

Project size: The projects duration is 3 years, it has a total budget of 36,5 mio Euro. The project is structured in 6 sub-projects: scenarios and requirements, continuous development method, system design, hardware, software, and demonstrators. Results that are common to all application domains are captured in the so-called Domain Common. Our research group has the academic sub-project lead in "scenarios and requirements together with AUDI as industrial lead.

RE Practice: N/A

RE Phase: Elicitation, Documentation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: German government

Repeatability: Confirming previous findings

<u>Lesson ID:</u> LL065_12 [Loft et al., 2012]

Journal: N/A

Conference: N/A

Workshop: TwinPeaks

Year: 2012

Lesson: A number of iterations through requirements considerations, software architecture considerations, and hardware considerations resulted in a more detailed specification of the requirements, accommodated by extensive changes to the software architecture, and respecting the given hardware constraints.

Source: Case study

Rationale: N/A

Impact: The original requirement was satisfied in full; it was an important requirement, and the solution demanded a major rewrite of many lines of code.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Graphical user interface for a home control device.

Project size: The project team size varied between 5 and 10 developers, and the total time consumption has been 18,000 hours. The hours are roughly distributed with 50% used for implementation, and 50% used for other activities including analysis and design. 100,000 lines of code have been written.

RE Practice: N/A

RE Phase: Elicitation

Software Process: Iterative, Scrum-like approach to development, with iterations of length approximately 4-5 weeks

Project Date: N/A

Recording Date: N/A

Organisation Name: Mjlner Informatics A/S (Mjlner)

Repeatability: First time case study results

Lesson ID: LL066_12 [Loft et al., 2012]

Journal: N/A

Conference: N/A

Workshop: TwinPeaks

Year: 2012

Lesson: Making architectural choices early is risky.

Source: Case study

Rationale: N/A

Impact: Ended up causing both partially unsatisfied requirements and deviations from the architecture.

Target Object: Technique/method

Type: Negative

Expression: Implicit

Application Domain: Graphical user interface for a home control device.

Project size: The project team size varied between 5 and 10 developers, and the total time consumption has been 18,000 hours. The hours are roughly distributed with 50% used for implementation, and 50% used for other activities including analysis and design. 100,000 lines of code have been written.

RE Practice: N/A

RE Phase: N/A

Software Process: Iterative, Scrum-like approach to development, with iterations of length approximately 4-5 weeks

Project Date: N/A

Recording Date: N/A

Organisation Name: Mjlner Informatics A/S (Mjlner)

Repeatability: First time case study results

Lesson ID: LL067_12 [Wu et al., 2012]

Journal: N/A

Conference: N/A

Workshop: TwinPeaks

Year: 2012

Lesson: It is important for the architect to negotiate requirements with the customer (maybe directly, or indirectly through the project leader) and try to persuade the customer to accommodate current architecture design if the new requirement changes are customer-specific or too large scaled.

Source: Case study

Rationale: The architect is the one that knows the most about the architecture, so she is the one that should be responsible for the consistency of the PLA. She may be facing the customer directly or convincing the product leader that the requirements are problematic with the architecture and not worthy fulfilled.

Impact: The requirements (denied directly or postponed temporarily) are highly possible to be raised in the future. Therefore, it is necessary to keep track of denied and postponed requirement change requests, and probably the architect should get somehow prepared for architectural changes.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Software product line (SPL) development. Wingsoft Examination System Product Line (WES-PL). Oral examination system for Shanghai Municipal Education and Examination Authority (SMEEA)

Project size: 10 member products, 51 major versions that have been delivered to customer or archived in the repository between December 2003 and May 2012.

RE Practice: N/A

RE Phase: Negotiation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Wingsoft Company

Repeatability: First time case study results

Lesson ID: LL068_12 [Wu et al., 2012]

Journal: N/A

Conference: N/A

Workshop: TwinPeaks

Year: 2012

Lesson: For future marketing, it is useful to collect various customer needs and examine whether combining different needs and variant components are meaningful.

Source: Case study

Rationale: There are two types of requirement elicitation in our case study: customer-oriented and future-market-oriented. For a customer, the exact needs are vague and hard to be expressed at the outset. This provides a space for architects and product leaders to build a concrete mental projection of the system by showing existing product instances and current architecture.

Impact: It will help product leader to build a dominant role in future requirement elicitation and also save effort negotiating requirements with future customers.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Software product line (SPL) development. Wingsoft Examination System Product Line (WES-PL). Oral examination system for Shanghai Municipal Education and Examination Authority (SMEEA)

Project size: 10 member products, 51 major versions that have been delivered to customer or archived in the repository between December 2003 and May 2012.
RE Practice: N/A
RE Phase: Elicitation
Software Process: N/A
Project Date: N/A
Recording Date: N/A
Organisation Name: Wingsoft Company
Repeatability: First time case study results

Lesson ID: LL069_12 [Petrov et al., 2012]
Journal: N/A
Conference: N/A
Workshop: TwinPeaks
Year: 2012
Lesson: Use the forward and backward inferred macro-architectural requirements approach to make inferred macro-architectural requirements explicit.
Source: Case study
Rationale: Traditionally the flow of authoritative information and control in requirements and software engineering is from requirements to architecture, design, development, implementation and testing. Iterative, spiral and agile methods, among others, have introduced increments and iterations in eliciting and discovering requirements within the project life cycle. Yet the authoritative flow of information across organizational boundaries within the enterprise continues to be from requirements to architecture to design.
Impact: Version 3 of the product was built on an architecture that was both "fit for purpose as well as "fit to context as defined by the macro- architecture and the forward-inferred and backward-inferred macro-architectural requirements.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: N/A

Project size: One representative sample case study. This is one of about twenty representative samples that we studied to date.

RE Practice: N/A

RE Phase: Elicitation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

Lesson ID: LL070_12 [Savio and Suryanarayana, 2012]

Journal: N/A

Conference: N/A

Workshop: TwinPeaks

Year: 2012

Lesson: It may be possible to accommodate changes to a FR by relaxing the QR by a minimal extent such that it is not noticed, or noticed only to a very small degree by the end user.

Source: Case study

Rationale: An end-user not only expects to see new functionality delivered with every release, he also wants the existing user- perceived quality to remain the same. The key criterion here is user-perceived quality, which is operationally different from the actual QR of the system.

Impact: Using such an approach can often avoid major changes to the existing architecture.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Intelligent electronic device (IED) configuration application (henceforth referred to as ConfigApp)

Project size: ConfigApp consists of various editors that allow a user to configure different aspects of an IED before it is deployed in the field. To optimize on the costs and resources required, ConfigApp builds upon the services provided by an existing engineering platform (henceforth referred to as the Platform) that is developed and maintained by another department at STS. The Platform provides a number of features that can be leveraged by ConfigApp, including a set of GUI controls for building GUIs, XML-based modeling of domain-specific data, persistent data storage, and logging and tracing mechanisms.

RE Practice: N/A

RE Phase: Elicitation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Siemens

Repeatability: First time case study results

Lesson ID: LL071_12 [Savio and Suryanarayana, 2012]

Journal: N/A

Conference: N/A

Workshop: TwinPeaks

Year: 2012

Lesson: Facilitating a dialogue between the PM and the SA. imperative that the stakeholders who decide what the solution to the end-users problem is to be i.e., the Product Manager and the System Architect communicate with each other through both formal and informal means.

Source: Case study

Rationale: Software engineering is a not only a technical endeavor, but a social process in which the dimensions of social interaction, communication, cooperation and co-ordination, are as important as the technical aspects.

Impact: Enable the PM to be aware of the potential repercussions on architecture due to changes in requirements, and that this awareness must be inculcated before changes to requirements are accepted.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Intelligent electronic device (IED) configuration application (henceforth referred to as ConfigApp)

Project size: ConfigApp consists of various editors that allow a user to configure different aspects of an IED before it is deployed in the field. To optimize on the costs and resources required, ConfigApp builds upon the services provided by an existing engineering platform (henceforth referred to as the Platform) that is developed and maintained by another department at STS. The Platform provides a number of features that can be leveraged by ConfigApp, including a set of GUI controls for building GUIs, XML-based modeling of domain-specific data, persistent data storage, and logging and tracing mechanisms.

RE Practice: N/A

RE Phase: Negotiation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Siemens

Repeatability: First time case study results

Lesson ID: LL072_12 [Wang et al., 2012]

Journal: N/A

Conference: REFSQ

Workshop: N/A

Year: 2012

Lesson: Use the simulation approach DepRVSim to predict the impact of requirement volatility on software project plans.

Source: Case study

Rationale: Requirement volatility is a common and inevitable project risk which has severe consequences on software projects. When requirement change occurs, a project manager wants to analyze its impact so as to better cope with it. As the modification to one requirement can cause changes in its dependent requirements and its dependency relationship, the impact analysis can be very complex.

Impact: DepRVSim can predict correctly in the probability that simulation results have K man hours offset from real effort deviation of around 45% and approximately 70%. Similar with effort deviation information, the results in Table 9 show that for 5 and 10 hours offset from real schedule deviation, DepRVSim can reach a correct rate of 49% and 70%.

Target Object: Tool

Type: Positive

Expression: Implicit

Application Domain: Software project management tool: Qone

Project size: More than 600 thousand source lines of code, this product has been developed and maintained for more than 7 years. More than 300 Chinese software organizations are using this tool to manage their projects. There are 24 requirements (R1$\tilde{R}$24) generated through the requirement phase in this release.It has 10 requirement changes.

RE Practice: N/A

RE Phase: Analysis

Software Process: Iterative process

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

<u>Lesson ID:</u> LL073_12 [Anh et al., 2012]

Journal: N/A

Conference: REFSQ

Workshop: N/A

Year: 2012

Lesson: Non-functional requirements are satisfactorily achieved by using OSS components.

Source: Survey

Rationale: There is considerable flexibility in requirements specifications (both functional and non-functional), as well as in the features of available OSS components. This allows a collaborative matching and negotiation process between stakeholders such as: customers, software contractors and OSS communities, regarding desired requirements versus available and thus reusable OSS components.

Impact: N/A

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Target is software-intensive organizations that adopt OSS in producing software product. This population includes organizations with different sizes and in different application domains. The application domain covers a wide variety of domains, including Communication system, Information system, Web application and Public-sector support, with a dominant of Public sector support in five cases.

Project size: 64 companies from our contact list were selected and contacted by phone call and email, in which fifteen stakeholders (developers or project leaders), who represented for 15 projects from Norway, Sweden and Spain, agreed to participate in the survey. The team size ranges from two to 250 people

RE Practice: N/A

RE Phase: Analysis

Software Process: The project life cycles include ad hoc development, waterfall, iterative development and agile, with a prevalence of the agile model in seven projects.

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First-time survey results

Lesson ID: LL074_12 [Anh et al., 2012]

Journal: N/A

Conference: REFSQ

Workshop: N/A

Year: 2012

Lesson: Customer involvement could enhance functional mismatch resolution while OSS community involvement could improve non-functional mismatch resolution.

Source: Survey

Rationale: There is considerable flexibility in requirements specifications (both functional and non-functional), as well as in the features of available OSS components. This allows a collaborative matching and negotiation process between stakeholders such as: customers, software contractors and OSS communities, regarding desired requirements versus available and thus reusable OSS components.

Impact: Three collaborative resolving requirement mismatch involve customers, OSS community and commercial vendor, alternatively. Keeping changes in components synchronized with OSS community is beneficial for fixing and maintaining these components. In resolving requirement mismatch, community involvement would not only reduce the developers effort in maintaining the components but also bringing more confidence on component quality as “given enough eyeballs, all bugs are shallow.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Target is software-intensive organizations that adopt OSS in producing software product. This population includes organizations with different sizes and in different application domains. The application domain covers a wide variety of domains, including Communication system, Information system, Web application and Public-sector support, with a dominant of Public sector support in five cases.

Project size: 64 companies from our contact list were selected and contacted by phone call and email, in which fifteen stakeholders (developers or project leaders), who represented for 15 projects from Norway, Sweden and Spain, agreed to participate in the survey. The team size ranges from two to 250 people

RE Practice: N/A

RE Phase: Analysis

Software Process: The project life cycles include ad hoc development, waterfall, iterative development and agile, with a prevalence of the agile model in seven projects.

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First-time survey results

Lesson ID: LL075_12 [Engelsman and Wieringa, 2012]

Journal: N/A

Conference: REFSQ

Workshop: N/A

Year: 2012

Lesson: Use the GORE language 'Light ARMOR' for designing ER (enterprise-architecture) C to trace back and forth between business goals and enterprise architecture components.

Source: Case study

Rationale: Ideally, all elements of an enterprise architecture can be traced to business goals ad vice versa, but in practice, this is not the case.

Impact: Use for (1) estimating impact of change and (2) justifying the presence of an architecture component.

Target Object: Tool

Type: Positive

Expression: Explicit

Application Domain: Drinking water production facility in the Netherlands

Project size: The company is responsible for the production and delivery of fresh drinking water to 1.2 million people and transports 73 billion liters of drinking water each year. It has about 500 employees divided over three divisions, viz. Production, Sales and Environment.

RE Practice: Tracing

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

Lesson ID: LL076_12 [Poort et al., 2012]

Journal: N/A

Conference: REFSQ

Workshop: N/A

Year: 2012

Lesson: The application of verification is positively correlated with IT project success. More specifically: IT projects that apply verification early in the development life cycle are significantly more successful than IT projects that apply verification late in the development life cycle.

Source: Survey

Rationale: Not properly taking NFRs into account is considered to be among the most expensive and difficult of errors to correct once an information system is completed and it is rated as one of the ten biggest risks in requirements engineering. NFRs are widely seen as the driving force for shaping IT systems architectures.

Impact: N/A

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: IT services

Project size: The invitation to participate in the survey was sent out by e-mail to around 350 members of the Netherlands (NL) Architecture Community of Practice (ACoP) of the ABC company. The ACoP consists of experienced professionals practicing architecture at various levels (business, enterprise, IT, software, and systems architecture) in project or consultancy assignments. The survey was closed after 16 days. By that time, 133 responses were collected. After elimination of duplicates (1), incomplete responses (51) and responses from respondents that indicated they had not fulfilled the role of architect on their latest project (41), 39 responses remained.

RE Practice: Prototyping, simulation, analysis, testing

RE Phase: Verification

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First-time survey results

Lesson ID: LL077_12 [Fricker and Schumacher, 2012]

Journal: N/A

Conference: REFSQ

Workshop: N/A

Year: 2012

Lesson: Use variability modeling that allows abstracting from requirements with AND, OR, and REQUIRES relationships to structure the release planning inputs.

Source: Case study

Rationale: Requirements catalogues for software release planning are often not complete and homogeneous. Current release planning approaches, however, assume such commitment to detail at least implicitly.

Impact: The feature tree, in comparison with a flat backlog of requirements, reduced complexity of release planning. The abstraction from requirements to features reduced the total number of elements to be considered by a factor 10.3. The feature tree and the roadmap were the key instruments used for deciding what to implement and when to implement. The feature tree provided a basis to discuss the scope of pilot projects with the stakeholders identified in the stakeholder tree. Stakeholder needs that could not directly be addressed led to discovering new potential features. In comparison to a flat list of requirements, the feature tree allowed building a mental model of the solution. The reduced number of features allowed building a shared vocabulary with stakeholders, the color coding visualizing growth of the solution, and

AND-OR feature dependencies understanding design options. This focused discussions and communication with stakeholders on aspects that were essential for planning. Decisions could be taken together with these stakeholders, which led to trust in the plans and in the product organization.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Software as a service for managing media such as text, sound, pictures, and movies

Project size: Responsible for the development was a product manager, a project manager, and a team of up to five developers. The requirements catalogue was managed in a word processor document and used as a basis for release planning. It contained 108 requirements. The requirements were grouped into 12 sections and 19 subsections or themes. In average, a group contained 3.6 requirements and was allocated to 1.93 releases.

RE Practice: Modeling, using feature trees

RE Phase: Analysis

Software Process: Agile

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability:First time case study results

Lesson ID: LL078_12 [Raspotnig and Opdahl, 2012]

Journal: N/A

Conference: REFSQ

Workshop: N/A

Year: 2012

Lesson: Combine FMEA with Failure Sequence Diagrams (FSD), a specialized version of Misuse Sequence Diagrams (MUSD).

Source: Case study

Rationale: In air traffic management (ATM) safety assessments are performed with traditional techniques such as failure mode and effect analysis (FMEA). As system modelling is becoming an increasingly important part of developing ATM systems, techniques that integrate safety aspects and modelling are needed.

Impact: FSD increased the understanding among the participants of how the system worked. FMEA should be used to give the structure of the analysis. more time was spent, but that they felt more sure about the analysis being thorough.participants were able to use FSD with little prior training. FSD is not able to cover all weaknesses of FMEA, especially not the assessment of multiple failures. FSD addresses components and their interactions in particular, which we conclude is an improvement of the FMEA technique and the overall safety assessment.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Air traffic management

Project size: FMEA team was established, with a facilitator, a secretary, three systems engineers and an air traffic control officer. Several of the participants were familiar with UML, but only one of them had previous experience with sequence diagrams. all participants received a document describing the FMTP system and the overall system, relevant safety documentation and a procedure for conducting the FMEA. The latter consisted of a worksheet with the columns component number, component, failure mode, causal factor, immediate effect, system effect, current controls and recommended action. Furthermore, it included a list of typical failure modes for components and software

RE Practice: Modeling

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Air Navigation Service Provider

Repeatability:First time case study results

Lesson ID: LL079_12 [Knauss and Schneider, 2012]

Journal: N/A

Conference: REFSQ

Workshop: N/A

Year: 2012

Lesson: Use the model based on heuristic critiques to encode new experiences and check requirements documentation.

Source: Case study

Rationale: Despite significant advances in requirements engineering (RE) research and practice, software developing organisations still struggle to create requirements documentation in sufficient quality and in a repeatable way. The notion of good-enough quality is domain and project specific. Software developing organisations need concepts that i) allow adopting a suitable set of RE methods for their domain and projects and ii) allow improving these methods continuously.

Impact: 89% correct answers: 100% of the changes at existing heuristic rules were correct and 86% of the newly created heuristic rules. Small changes lasted less than 2 minutes. A new heuristic rule could be created in less than 7 minutes.

Target Object: Tool

Type: Positive

Expression: Implicit

Application Domain: University

Project size: Seven volunteers., two of them still in their Bachelors (3rd and 5th year / regular: 3 years)

RE Practice: Use of automatic checks

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

<u>Lesson ID:</u> LL080_12 [Lauesen, 2012]

Journal: N/A

Conference: REFSQ

Workshop: N/A

Year: 2012

Lesson: Use task description for user interface requirements.

Source: Industrial experience

Rationale: Although use cases are supposed to explain the context, they rarely do in practice. Some of the user-story aspects are missing. When would the user do this? And what will he do afterwards? Notice that a system that just reports Registration rejected fully meets the requirement expressed by the use case.

Impact: Task descriptions are an alternative that combines the best parts of user stories and use cases.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Electronic Land Registry system

Project size: Based on historical data, it was estimated that 5 million registrations would be handled per year, corresponding to about 2 per second during peak load in daytime. (Denmark has around 5 million inhabitants).

RE Practice: Use of task description

RE Phase: Specification

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Danish government

Repeatability: Industry results

<u>Lesson ID:</u> LL081_12 [Lauesen, 2012]

Journal: N/A

Conference: REFSQ

Workshop: N/A

Year: 2012

Lesson: Do not restrict the definition of the user interface to the domain expert (a land registration judge) and the supplier's designer without any usability testing.

Source: Industrial experience

Rationale: Usability experts know that a user interface designed in this way is only understandable to a domain expert. And this turned out to be the case also in this project. The lawyers and real-estate agents didn't understand.

Impact: Lead to an interface that is only understood by the domain expert and not the average user.

Target Object: Technique/method

Type: Negative

Expression: Implicit

Application Domain: Electronic Land Registry system

Project size: Based on historical data, it was estimated that 5 million registrations would be handled per year, corresponding to about 2 per second during peak load in daytime. (Denmark has around 5 million inhabitants).

RE Practice: N/A

RE Phase: Elicitation, specification

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Danish government

Repeatability: Industry results

Lesson ID: LL082_12 [Zhu and Herrmann, 2012]

Journal: N/A

Conference: N/A

Workshop: Workshop on Creativity in Requirements Engineering

Year: 2012

Lesson: Use the meta-design approach for eliciting requirements from stakeholders.

Source: Case study

Rationale:Future uses and problems cannot be completely anticipated at the software design time, thus requiring software environments that can be evolved at use time. The co-evolution of systems and users social practices challenges requirements engineering (RE). Since it is unrealistic to come up with fully described requirements for yet unknown problems and a continuously changing context, it is necessary to extend the RE-process in use time, providing possibilities to accommodate emergent new requirements.

Impact: A meta-design approach not only enables requirements engineering at use time but also enhances different levels of creativity: 1) opportunistic programming as bricologe at the meta-design level, in that meta-designers constantly evolved the MikiWiki design environment opportunistically to cope with emergent socio-technical issues without needing to change server-side code; and 2) creativity-in-use at the design and use level, in that designers and users invent their own ways to use MikiWiki which are not envisioned by meta-designers. In addition, a more visual-based approach is appropriate to involve different design communities and enhance creativity.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Mobile version of a micro-survey tool, the Creativity Barometer. The purpose of the Creativity Barometer is to conduct surveys to continuously understand and assess the climate of a companys creativity support. The Creativity Barometer allows companies to periodically repeat surveys and get instant feedback continuously. After a pre-specified time period (e.g. eight months), the company can summarize the feedback and plan interventions to improve the creativity climate.

Project size: The design sessions involved 11 participants - four female and seven male, aged from 25 to 55 years, and comprising MA, MSc and PhD students as well as associate professors. 5 design sessions, which were organized to involve different types of participants. Group 1 and 2 consisted of two designers; group 3 consisted of two users and two designers from the previous design session; group 4 was made purely of two users; group 5 consisted of one designer and two users.

RE Practice: N/A

RE Phase: Elicitation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Information and Technology Management Group at the Ruhr-University

Repeatability: First time case study results

Lesson ID: LL083_12 [Savio and P.C., 2012]

Journal: N/A

Conference: N/A

Workshop: Workshop on Creativity in Requirements Engineering

Year: 2012

Lesson: Use the 'Pictionade' game to communicate requirements among stakeholders.

Source: Case study

Rationale: The various issues involved in communicating requirements across multiple stakeholders and stakeholder groups have been well documented in literature and in experience reports. Despite this, however, most stakeholders involved in a project seem largely unaware of what the potential consequences of these issues can be. The manner in which stakeholders communicate requirements to each other affects the subsequent requirements management activities, and has a direct impact on the final form and scope of the stated requirement.

Impact: We were able to determine that there were several categories of end users for the system, and were encouraged to think from each of the groups perspectives, and visualize their notion of the look and feel of the system. Since text based representations were kept to a minimum, discussion times were cut down considerably we were able to come to a quick consensus and visual clarity was provided on most of the interfaces. Due to this, we were able to elicit and communicate several requirements for the user interface of the software system which we may have otherwise overlooked. We were also able to determine use case scenarios, and develop a context diagram with actors and end users from the pictures.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Management of an industrial automation plant

Project size: N/A

RE Practice: Communication

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

<u>Lesson ID:</u> LL084_12 [Savio and P.C., 2012]

Journal: N/A

Conference: N/A

Workshop: Workshop on Creativity in Requirements Engineering

Year: 2012

Lesson: The overall system or product and its purpose must be described as clearly as possible before communicating its features.

Source: Case study

Rationale: N/A

Impact: N/A

Target Object: Technique/method

Type: Neutral

Expression: Explicit

Application Domain: An online book shopping portal and a smart phone

Project size: 9 PMs, having several years of experience on a wide range of industry projects. We divided them into two teams - team 'A, comprising three PMs, and team 'B with six PMs. Each person in both teams assumed one of three roles the artist, the actor or the interpreter. A few high level requirements for two products were given only to the artists from each team.

a few high level requirements from the end users perspective, for two example products - an online book shopping portal and a smart phone.

RE Practice: Applying ‘Pictionades’

RE Phase: Elicitation, specification

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Siemens

Repeatability: N/A

Lesson ID: LL085_12 [Savio and P.C., 2012]

Journal: N/A

Conference: N/A

Workshop: Workshop on Creativity in Requirements Engineering

Year: 2012

Lesson: Several rounds of communication may be required to obtain clarity on a single point.

Source: Case study

Rationale: This point was demonstrated when we observed the participants trying to refine their articulation of the information that they wanted to communicate, and provide more intelligibility on expression techniques, when it was observed that the recipient of the information hadnt fully grasped the information.

Impact: N/A

Target Object: Technique/method

Type: Neutral

Expression: Explicit

Application Domain: An online book shopping portal and a smart phone

Project size: 9 PMs, having several years of experience on a wide range of industry projects.

We divided them into two teams - team ‘A, comprising three PMs, and team ‘B with six PMs. Each person in both teams assumed one of three roles the artist, the actor or the interpreter. A few high level requirements for two products were given only to the artists from each team. a few high level requirements from the end users perspective, for two example products - an online book shopping portal and a smart phone.

RE Practice: Applying ‘Pictionades’, Communication

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Siemens

Repeatability: N/A

Lesson ID: LL086_12 [Savio and P.C., 2012]

Journal: N/A

Conference: N/A

Workshop: Workshop on Creativity in Requirements Engineering

Year: 2012

Lesson: Establishing well understood terminology that can be understood irrespective of stakeholders backgrounds.

Source: Case study

Rationale: This point was observed when we saw the participants instinctively making use of hand signals such as a raised palm for stop and start over, a quick shake of the hand/head for no, incorrect, a thumbs up sign for right and so on, when they realized that the other participants easily understood these gestures.

Impact: An effective means in which communication can be made more effective.

Target Object: Technique/method

Type: Neutral

Expression: Explicit

Application Domain: An online book shopping portal and a smart phone

Project size: 9 PMs, having several years of experience on a wide range of industry projects. We divided them into two teams - team ‘A, comprising three PMs, and team ‘B with six PMs. Each person in both teams assumed one of three roles the artist, the actor or the interpreter. A few high level requirements for two products were given only to the artists from each team. a few high level requirements from the end users perspective, for two example products - an online book shopping portal and a smart phone.

RE Practice: Applying ‘Pictionades’, Communication

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Siemens

Repeatability: N/A

Lesson ID: LL087_12 [Savio and P.C., 2012]

Journal: N/A

Conference: N/A

Workshop: Workshop on Creativity in Requirements Engineering

Year: 2012

Lesson: It is important to communicate what the system is not supposed to do, in addition to what features and functions it must exhibit.

Source: Case study

Rationale: N/A

Impact: This would help in the formulation of both positive and negative use-case scenarios of the system. This learning is indirectly helpful for stakeholders in determining what components should go into the system, interface, environment and domain.

Target Object: Technique/method

Type: Neutral

Expression: Explicit

Application Domain: An online book shopping portal and a smart phone

Project size: 9 PMs, having several years of experience on a wide range of industry projects. We divided them into two teams - team ‘A, comprising three PMs, and team ‘B with six PMs. Each person in both teams assumed one of three roles the artist, the actor or the interpreter. A few high level requirements for two products were given only to the artists from each team. a few high level requirements from the end users perspective, for two example products - an online book shopping portal and a smart phone.

RE Practice: Applying ‘Pictionades’, Communication

RE Phase: N/A

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Siemens

Repeatability: N/A

Lesson ID: LL088_12 [Savio and P.C., 2012]

Journal: N/A

Conference: N/A

Workshop: Workshop on Creativity in Requirements Engineering

Year: 2012

Lesson: The importance of continuous validation and feedback on communicated items as and when possible cannot be underestimated.

Source: Case study

Rationale: N/A

Impact: N/A

Target Object: Technique/method

Type: Neutral

Expression: Explicit

Application Domain: An online book shopping portal and a smart phone

Project size: 9 PMs, having several years of experience on a wide range of industry projects. We divided them into two teams - team ‘A, comprising three PMs, and team ‘B with six PMs. Each person in both teams assumed one of three roles the artist, the actor or the interpreter. A few high level requirements for two products were given only to the artists from each team. a few high level requirements from the end users perspective, for two example products - an online book shopping portal and a smart phone.

RE Practice: Applying ‘Pictionades’

RE Phase: Validation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Siemens

Repeatability: N/A

Lesson ID: LL089_12 [Calefato et al., 2012]

Journal: EMSE

Conference: N/A

Workshop: N/A

Year: 2012

Lesson: Requirements elicitation is the task where computer-mediated communication tools have most opportunity for successful application.

Source: Controlled experiment

Rationale: Effective communication is crucial to system design. Especially at the requirements stage, system design is a social and communication-intensive activity that relies on an effective collaboration of stakeholders with diverse professional and cultural backgrounds. Whether engineered or naturally emerging and agreed upon during a negotiation process, requirements demand increased communication during elicitation and negotiation. Effective communication is vital during these activities to overcome the semantic gap between users and designers, as well as to reconcile the aspects of the design process affected by human and organizational factors. Computer-mediation has the potential to overcome problems of group dynamics in large groups; as well, software teams increasingly develop software in predominantly distributed settings and rely on computer- mediated collaborative tools to mediate their design activities. Geographical, organizational, and cultural distance brings additional challenges to effective communication and results in misunderstandings, the loss of opportunities for rich interaction, and a reduction in frequency of both formal and informal communication.

Impact: N/A

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: UVic Centre for Scholastic Entertainment Edu Game, Equipment and Patient Tracking for St. Peter Hospital, Bus Tracking System, consulting Groupwork System, University of Vancouver Island Room Organization System, SysCal Shared Calendar

Project size: The course involved thirty-eight students working in six project teams in the development of six realistic software projects. each group composed of five to eight randomly-selected student. outcome was a requirements specification (RS)

RE Practice: Use of text-based synchronous communication

RE Phase: Elicitation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: University of Victoria

Repeatability: These results confirm the predictions of socio-psychological theories that the depersonalization effect induced by the use of less-rich and less-social media limits domination, group/social pressure, and other dysfunctional aspects intrinsic to F2F group communication and that are specific to requirements group approaches.

Lesson ID: LL090_12 [Calefato et al., 2012]

Journal: EMSE

Conference: N/A

Workshop: N/A

Year: 2012

Lesson: Use text-based communication rather than F2F communication to discuss conflicting issues.

Source: Controlled experiment

Rationale: Effective communication is crucial to system design. Especially at the requirements stage, system design is a social and communication-intensive activity that relies on an effective collaboration of stakeholders with diverse professional and cultural backgrounds. Whether engineered or naturally emerging and agreed upon during a negotiation process, requirements demand increased communication during elicitation and negotiation. Effective communication is vital during these activities to overcome the semantic gap between users and designers, as well as to reconcile the aspects of the design process affected by human and organizational factors. Computer-mediation has the potential to overcome problems of group dynamics in large groups; as well, software teams increasingly develop software in predominantly distributed settings and rely on computer- mediated collaborative tools to mediate their design activities.

Geographical, organizational, and cultural distance brings additional challenges to effective communication and results in misunderstandings, the loss of opportunities for rich interaction, and a reduction in frequency of both formal and informal communication.

Impact: N/A

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: UVic Centre for Scholastic Entertainment Edu Game, Equipment and Patient Tracking for St. Peter Hospital, Bus Tracking System, consulting Groupwork System, University of Vancouver Island Room Organization System, SysCal Shared Calendar

Project size: The course involved thirty-eight students working in six project teams in the development of six realistic software projects. each group composed of five to eight randomly-selected student. outcome was a requirements specification (RS)

RE Practice: Use of text-based synchronous communication

RE Phase: Elicitation, negotiation, communication

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: University of Victoria

Repeatability: These results confirm the predictions of socio-psychological theories that the depersonalization effect induced by the use of less-rich and less-social media limits domination, group/social pressure, and other dysfunctional aspects intrinsic to F2F group communication and that are specific to requirements group approaches.

Lesson ID: LL091_12 [Calefato et al., 2012]

Journal: EMSE

Conference: N/A

Workshop: N/A

Year: 2012

Lesson: Use a F2F medium to familiarize with others, and to express complex ideas and to understand others opinions.

Source: Controlled experiment

Rationale: Effective communication is crucial to system design. Especially at the requirements stage, system design is a social and communication-intensive activity that relies on an effective collaboration of stakeholders with diverse professional and cultural backgrounds. Whether engineered or naturally emerging and agreed upon during a negotiation process, requirements demand increased communication during elicitation and negotiation. Effective communication is vital during these activities to overcome the semantic gap between users and designers, as well as to reconcile the aspects of the design process affected by human and organizational factors. Computer-mediation has the potential to overcome problems of group dynamics in large groups; as well, software teams increasingly develop software in predominantly distributed settings and rely on computer- mediated collaborative tools to mediate their design activities. Geographical, organizational, and cultural distance brings additional challenges to effective communication and results in misunderstandings, the loss of opportunities for rich interaction, and a reduction in frequency of both formal and informal communication.

Impact: N/A

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: UVic Centre for Scholastic Entertainment Edu Game, Equipment and Patient Tracking for St. Peter Hospital, Bus Tracking System, consulting Groupwork System, University of Vancouver Island Room Organization System, SysCal Shared Calendar

Project size: The course involved thirty-eight students working in six project teams in the development of six realistic software projects. each group composed of five to eight randomly-selected student. outcome was a requirements specification (RS)

RE Practice: Use of text-based synchronous communication

RE Phase: Elicitation, negotiation, communication

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: University of Victoria

Repeatability: These results confirm the predictions of socio-psychological theories that the depersonalization effect induced by the use of less-rich and less-social media limits domination, group/social pressure, and other dysfunctional aspects intrinsic to F2F group communication and that are specific to requirements group approaches.

Lesson ID: LL092_12 [Calefato et al., 2012]

Journal: EMSE

Conference: N/A

Workshop: N/A

Year: 2012

Lesson: Use TXT for having better task facilitation specifically in structured discussion, proper documentation, and visibility of decisions made.

Source: Controlled experiment

Rationale: Effective communication is crucial to system design. Especially at the requirements stage, system design is a social and communication-intensive activity that relies on an effective collaboration of stakeholders with diverse professional and cultural backgrounds. Whether engineered or naturally emerging and agreed upon during a negotiation process, requirements demand increased communication during elicitation and negotiation. Effective communication is vital during these activities to overcome the semantic gap between users and designers, as well as to reconcile the aspects of the design process affected by human and organizational factors. Computer-mediation has the potential to overcome problems of group dynamics in large groups; as well, software teams increasingly develop software in predominantly distributed settings and rely on computer- mediated collaborative tools to mediate their design activities. Geographical, organizational, and cultural distance brings additional challenges to effective communication and results in misunderstandings, the loss of opportunities for rich interaction, and a reduction in frequency of both formal and informal communication.

Impact: N/A

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: UVic Centre for Scholastic Entertainment Edu Game, Equipment and Patient Tracking for St. Peter Hospital, Bus Tracking System, consulting Groupwork System, University of Vancouver Island Room Organization System, SysCal Shared Calendar

Project size: The course involved thirty-eight students working in six project teams in the development of six realistic software projects. each group composed of five to eight randomly-selected student. outcome was a requirements specification (RS)

RE Practice: Use of text-based synchronous communication

RE Phase: Elicitation, negotiation, communication

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: University of Victoria

Repeatability: These results complement findings from previous GSS- related research that groups interacting on text-based channels often outperform collocated groups in tasks of idea generation because of the ability to input ideas in parallel.

Lesson ID: LL093_12 [Calefato et al., 2012]

Journal: EMSE

Conference: N/A

Workshop: N/A

Year: 2012

Lesson: Lean media offer less support to achieving common ground during requirements negotiations than during elicitations.

Source: Controlled experiment

Rationale: Effective communication is crucial to system design. Especially at the requirements stage, system design is a social and communication-intensive activity that relies on an effective collaboration of stakeholders with diverse professional and cultural backgrounds. Whether engineered or naturally emerging and agreed upon during a negotiation process, requirements demand increased communication during elicitation and negotiation. Effective communication is vital during these activities to overcome the semantic gap between users and designers, as well as to reconcile the aspects of the design process affected by human and organizational factors. Computer-mediation has the potential to overcome problems of group dynamics in large groups; as well, software teams increasingly develop software in predominantly distributed settings and rely on computer- mediated collaborative tools to mediate their design activities. Geographical, organizational, and cultural distance brings additional challenges to effective communication and results in misunderstandings, the loss of opportunities for rich interaction, and a reduction in frequency of both formal and informal communication.

Impact: N/A

Target Object: Technique/method

Type: Negative

Expression: Implicit

Application Domain: UVic Centre for Scholastic Entertainment Edu Game, Equipment and Patient Tracking for St. Peter Hospital, Bus Tracking System, consulting Groupwork System, University of Vancouver Island Room Organization System, SysCal Shared Calendar

Project size: The course involved thirty-eight students working in six project teams in the development of six realistic software projects. each group composed of five to eight randomly-selected student. outcome was a requirements specification (RS)

RE Practice: Use of text-based synchronous communication

RE Phase: Elicitation, negotiation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: University of Victoria

Repeatability: add to the previous findings that TXT negotiations represent a poor task/technology fit.

Lesson ID: LL094_12 [Wnuk et al., 2012]

Journal: EMSE

Conference: N/A

Workshop: N/A

Year: 2012

Lesson: The linguistic method is not more efficient in consolidating requirements than the searching and filtering method.

Source: Controlled experiment

Rationale: Large market-driven software companies continuously receive large numbers of requirements and change requests from multiple sources. The task of analyzing those requests against each other and against already analyzed or implemented functionality then recording similarities between them, also called the requirements consolidation task, may be challenging and time consuming.

Impact: N/A

Target Object: Technique/method, tool

Type: Negative

Expression: Implicit

Application Domain: Requirements specifications were produced as a part of a course "Software Development of Large Systems

Project size: 45 subjects, working in pairs on the same set of requirements as in the original study. Two requirements sets containing 30 and 160 requirements respectively, were imported to ReqSimlieA and Telelogic Doors

RE Practice: N/A

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Lund University

Repeatability: Contradicts the findings of the original study

Lesson ID: LL095_12 [Wnuk et al., 2012]

Journal: EMSE

Conference: N/A

Workshop: N/A

Year: 2012

Lesson: Assisted method (lexical similarity) can deliver more correct links and miss fewer links than the manual method (searching and filtering).

Source: Controlled experiment

Rationale: Large market-driven software companies continuously receive large numbers of requirements and change requests from multiple sources. The task of analyzing those requests against each other and against already analyzed or implemented functionality then recording similarities between them, also called the requirements consolidation task, may be challenging and time consuming.

Impact: N/A

Target Object: Technique/method, tool

Type: Positive

Expression: Implicit

Application Domain: Requirements specifications were produced as a part of a course "Software Development of Large Systems

Project size: 45 subjects, working in pairs on the same set of requirements as in the original study. Two requirements sets containing 30 and 160 requirements respectively, were imported to ReqSimlieA and Telelogic Doors

RE Practice: N/A

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Lund University

Repeatability: Confirms the previous results

Lesson ID: LL096_12 [Alrajeh et al., 2012]

Journal: N/A

Conference: ICSE

Workshop: N/A

Year: 2012

Lesson: Use Alrajeh's tool-supported technique for generating a set of obstacle conditions guaranteed to be complete and consistent with respect to the known domain properties.

Source: Case study

Rationale: Missing requirements are known to be among the major causes of software failure.

They often result from a natural inclination to conceive over-ideal systems where the software-to-be and its environment always behave as expected. Obstacle analysis is a goal-anchored form of risk analysis whereby exceptional conditions that may obstruct system goals are identified, assessed and resolved to produce complete requirements. Various techniques have been proposed for identifying obstacle conditions systematically. Among these, the formal ones have limited applicability or are costly to automate.

Impact: Improvement over existing methods through automation and detection of a wider class of obstacles. provides support for eliciting new domain properties. produced finer sub-obstacles depending on the granularity of the provided domain properties.

Target Object: Tool

Type: Positive

Expression: Implicit

Application Domain: Safety-critical system

Project size: Medium-sized

RE Practice: N/A

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: London Ambulance Service

Repeatability: First time case study results

<u>***Lesson ID:***</u> LL097_12 [Post et al., 2012]

Journal: RE

Conference: N/A

Workshop: N/A

Year: 2012

Lesson: Behavioral requirements can be specified via the specification pattern system (SPS) and 3 further patterns by Post et al.

Source: Case study

Rationale: To allow an automatic formal analysis of requirements, the requirements have to be formalized first. However, logical formalisms are seldom accessible to stakeholders in the automotive context. Konrad and Cheng proposed a specification pattern system (SPS) represented in a restricted English grammar that can be automatically translated to logics, but looks like natural language.

Impact: N/A

Target Object: Technique/method

Type: Neutral

Expression: Implicit

Application Domain: Automotive

Project size: Five requirements documents. 289 informal behavioral requirements taken from automotive BOSCH projects.

RE Practice: Use of specification pattern system (SPS)

RE Phase: Specification

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: BOSCH

Repeatability: First time case study results

Lesson ID: LL098_12 [Schneider et al., 2012]

Journal: RE

Conference: N/A

Workshop: N/A

Year: 2012

Lesson: Use Schneider's et al. tool/approach to support reuse of existing experiences that are relevant for security while eliciting and analyzing security requirements.

Source: Case study

Rationale: More and more software projects today are security-related in one way or the other. Requirements engineers without expertise in security are at risk of over- looking security requirements, which often leads to security vulnerabilities that can later be exploited in practice. Identifying security-relevant requirements is labor-intensive and error-prone.

Impact: Very good results in cases where the classifier is applied to the requirements from the same source as it was trained with. We obtained poor results in cases where the classifier was applied to a different requirements specification than the one it was trained with. we enable people to exchange experiences about security-relevant requirements while they write and discuss project requirements. At the same time, the approach enables participating stakeholders to learn while they write requirements. This can increase security awareness and facilitate learning on both individual and organizational levels. support reuse of existing experiences that are relevant for security.

Target Object: Technique/method, tool

Type: Positive

Expression: Implicit

Application Domain: Telecommunication

Project size: N/A

RE Practice: N/A

RE Phase: Elicitation, analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: European Telecommunications Standards Institute

Repeatability: First time case study results

Lesson ID: LL099_12 [Lauesen and Kuhail, 2012]

Journal: RE

Conference: N/A

Workshop: N/A

Year: 2012

Lesson: Use task description rather than use cases for specifying the requirements that deal with the interaction between a human user and the system.

Source: Case study

Rationale: N/A

Impact: With use cases, the customers present problems disappear unless the analyst can see a solution to the problem. The consequence is that when the customer looks for a new system, he will not take into account how well the new system deals with the problems. Even if the analyst has specified a solution, a better solution may not get the merit it deserves because the corresponding problem is not visible in the use cases. Task descriptions avoid this by allowing the analyst to state a problem as one of the steps, with the implicit requirement that a solution is wanted (a problem requirement). use cases in practice produce too restrictive requirements. Task descriptions do not specify a dialog but only what user and system need to do together. Tasks are also a good basis for designing the user interface because the developer can focus on designing screens that conveniently show the data needed during the task. He can add functionality and the dialog later. Use cases had the advantage of spreading with OOA/OOD and powerful consultants. Using tasks instead requires a lot of change in present practice and tools.

Target Object: Technique/method

Type: Positive, negative

Expression: Implicit

Application Domain: Hotline (help desk)

Project size: 15 replies, eight used traditional use cases that specified a dialog between user and system. Seven used a related technique, task description, which specified the customers needs without specifying a dialog.

RE Practice: Use of use cases and task descriptions

RE Phase: Specification

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

Lesson ID: LL100_12 [Maxwell et al., 2012b]

Journal: RE

Conference: N/A

Workshop: N/A

Year: 2012

Lesson: Use Maxwell's et al. taxonomy to identify conflicting compliance requirements due to cross-references in legal texts.

Source: Multi Case study

Rationale: Companies must ensure their software complies with relevant laws and regulations to avoid the risk of costly penalties, lost reputation, and brand damage result- ing from non-compliance. Laws and regulations contain internal cross-references to portions of the same legal text, as well as cross-references to external legal texts. These cross-references introduce ambiguities, exceptions, as well as other challenges to regulatory compliance. Requirements engineers need guidance as to how to address cross- references in order to comply with the requirements of the law.

Impact: Aid requirements engineers in identifying compliance requirements that appear to conflict so that these conflicts may subsequently be resolved. 5 conflicts were identified.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Health care, finance

Project size: 177 total cross-references within the HIPAA Privacy Rule. 360 total cross-references within the GLB Act. a total of 367 cross- references within the GLBA Financial Privacy Rule

RE Practice: N/A

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: U.S. Health Insurance Portability and Accountability Act (HI- PAA) Privacy Rule, the GrammLeachBliley Act (GLBA), and the GLBA Financial Privacy Rule

Repeatability: First time case study results

Lesson ID: LL101_12 [Maxwell et al., 2012b]

Journal: RE

Conference: N/A

Workshop: N/A

Year: 2012

Lesson: When a compliance requirement expressed in one legal text is more restrictive than the corresponding compliance requirement expressed in another legal text, requirements engineers should choose to follow the more restrictive of the two.

Source: Multi Case study

Rationale: Companies must ensure their software complies with relevant laws and regulations to avoid the risk of costly penalties, lost reputation, and brand damage result- ing from non-compliance. Laws and regulations contain internal cross-references to portions of the same legal text, as well as cross-references to external legal texts. These cross-references introduce ambiguities, exceptions, as well as other challenges to regulatory compliance. Requirements engineers need guidance as to how to address cross- references in order to comply with the requirements of the law.

Impact: In complying with the more restrictive text, the system complies with both.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Health care, finance

Project size: 177 total cross-references within the HIPAA Privacy Rule. 360 total cross-references within the GLB Act. a total of 367 cross- references within the GLBA Financial Privacy Rule

RE Practice: N/A

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: U.S. Health Insurance Portability and Accountability Act (HI- PAA) Privacy Rule, the GrammLeachBliley Act (GLBA), and the GLBA Financial Privacy Rule

Repeatability: First time case study results

Lesson ID: LL102_12 [Maxwell et al., 2012b]

Journal: RE

Conference: N/A

Workshop: N/A

Year: 2012

Lesson: Conflicts between obligations and privileges can be resolved by not exercising legal privileges.

Source: Multi Case study

Rationale: Because an obligation trumps a privilege due to its priority.

Impact: N/A

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Health care, finance

Project size: 177 total cross-references within the HIPAA Privacy Rule. 360 total cross-references within the GLB Act. a total of 367 cross- references within the GLBA Financial Privacy Rule

RE Practice: N/A

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: U.S. Health Insurance Portability and Accountability Act (HI- PAA) Privacy Rule, the GrammLeachBliley Act (GLBA), and the GLBA Financial Privacy Rule

Repeatability: First time case study results

Lesson ID: LL103_12 [Fitzgerald et al., 2012]

Journal: RE

Conference: N/A

Workshop: N/A

Year: 2012

Lesson: Use Fitzgerald's et al. failure prediction model

Source: Evaluating experiment

Rationale: Online feature request management systems are popular tools for gathering stakeholders change requests during system evolution. Deciding which feature requests require attention and how much upfront analysis to perform on them is an important problem in this context: too little upfront analysis may result in inadequate functionalities being developed, costly changes, and wasted development effort; too much upfront analysis is a waste of time and resources. Early predictions about which feature requests are most likely to fail due to insufficient or inadequate upfront analysis could facilitate such decisions.

Impact: An early failure prediction approach can benefit projects by guiding upfront requirements analysis

Target Object: Technique/method, tool

Type: Positive

Expression: Implicit

Application Domain: Apache web server, the Eclipse development environment, Firefox web browser, the KDE operating system the Netbeans development environment, Thunderbird email client, and the Wiki- media content management system.

Project size: 7 large-scale projects. The projects were all large in size, ranging from 5,000 to 50,000 feature requests.

RE Practice: N/A

RE Phase: Elicitation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time experimental results

<u>Lesson ID:</u> LL104_12 [Milne and Maiden, 2012]

Journal: RE

Conference: N/A

Workshop: N/A

Year: 2012

Lesson: Using Milne's et al. frame work to revealing the reality of power and politics in organisations or projects.

Source: Case study

Rationale: Power and politics are acknowledged as factors that can impact on the RE process .

Impact: Will not necessarily lead to the production of better requirements. A time-consuming approach. An intrusive approach

Target Object: Technique/method

Type: Negative

Expression: Implicit

Application Domain: Website for a publishing company

Project size: The total project budget was in the region of £1 million, and the project duration was around 1 year from initial scoping meetings through to implementation. The project involved a large number of staff from within the organisation, together with external consultants and third party suppliers.

RE Practice: N/A

RE Phase: Analysis

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

Lesson ID: LL105_12 [McGee and Greer, 2012]

Journal: RE

Conference: N/A

Workshop: N/A

Year: 2012

Lesson: Use the requirements change taxonomy for change classification and measurement.

Source: Case study

Rationale: Changes to software requirements not only pose a risk to the successful delivery of software applications but also provide opportunity for improved usability and value. Increased understanding of the causes and consequences of change can support requirements management and also make progress towards the goal of change anticipation.

Impact: Feasibly practical and will aid understanding of software evolution during development as well as providing opportunities for retrospective project analysis to aid future process and technique tailoring.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: IT services

Project size: Employs 300 staff, has offices in England and Ireland. The project has an estimated cost in excess of a million pounds, comprises on average 15 software developers and analysts

RE Practice: N/A

RE Phase: Analysis

Software Process: Waterfall

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

<u>Lesson ID:</u> LL106_12 [Atladottir et al., 2012]

Journal: RE

Conference: N/A

Workshop: N/A

Year: 2012

Lesson: Use a low-fidelty prototype to elicit requirements.

Source: Case study

Rationale: Identifying accurate user requirements early in the design cycle is of the utmost importance in system development.

Impact: That users who have the system described to them with the aid of a low-fidelity prototype are able to come up with a greater number of new or changed requirements than users who work with a high- fidelity prototype. Working with the Hi-Fi prototype appeared to be a hindrance for users, as they become so involved with the details of working the system that they became distracted from contemplating new or changed features that it may have been appropriate for them to suggest.

Target Object: Tool

Type: Positive

Expression: Implicit

Application Domain: HCD-suite, is innovative software that supports the training management life cycle in a telecommunication company

Project size: The study included fourteen participants, all employees of a telecommunication company, and their ages ranged from 24 to 60.

RE Practice: Protoyping

RE Phase: Elicitation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: First time case study results

Lesson ID: LL107_12 [Sakhnini et al., 2012]

Journal: RE

Conference: N/A

Workshop: N/A

Year: 2012

Lesson: Use Power- Only EPMcreatefor requirements elicitation rather than full EPMcrea tand brainstorming.

Source: Controlled experiment

Rationale: Creativity is often needed in requirements elicitation, i.e., requirement idea generation; and techniques to enhance creativity are believed to be useful.

Impact: The results of the first experiment indicate that Power- Only EPMcreate is more effective, by the quantity and quality of the ideas generated, than the full EPMcreate, which is, in turn, more effective than brainstorming.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: High school website

Project size: Six groups, two of which used POEPMcreate, two of which used EPMcreate, and two of which used brainstorming. The second experiment compared the requirement ideas for the very same CBS generated by eight groups, four of which used POEPMcreate and four of which used EPMcreate.

RE Practice: Brainstorming

RE Phase: Elicitation

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: Sir John A MacDonald High School

Repeatability: Results of two controlled experiments

Lesson ID: LL108_12 [Svensson et al., 2012]

Journal: RE

Conference: N/A

Workshop: N/A

Year: 2012

Lesson: Use the QUality PERformance (QUPER) model with estimations of benefit and cost of quality targets in relation to market expectations as a basis for the architecting of quality requirements.

Source: Case study

Rationale: Quality requirements play a critical role in driving architectural design and are an important issue in software development. Therefore, quality requirements need to be considered, specified, and quantified early during system analysis and not later in the development phase in an ad-hoc fashion.

Impact: In general, QUPER does not only help in creating a more aligned view of quality requirements, but also to use one method to measure all quality requirements. All subjects confirmed that QUPER would support and coordinate the early decision-making process, e.g., release planning. The QUPER model is aimed to facilitate the elicitation, specification, quantification, and prioritization of QR.

Target Object: Technique/method

Type: Positive

Expression: Implicit

Application Domain: Electronic payment-processing: payment terminals, transaction processing, and development of saving- and customer-card systems

Project size: Company employs more than 250 employees, has more than 120,000 customers and business partners

RE Practice: N/A

RE Phase: Elicitation, Specification, Prioritization

Software Process: N/A

Project Date: N/A

Recording Date: N/A

Organisation Name: N/A

Repeatability: Confirm previous results from the mobile handset

Appendix D: Lesson Object and Tool Support

This appendix includes initial ideas for a lesson learnt tool in Requirements Engineering.

The tool is anticipated to support the following concepts:

- **Lesson Object:** A lesson object would encapsulate all the attributes of a lesson (see Chapter 3 and 4) and applicable operations.
- **Relationships:** Defines the relationship among the objects (i.e., lessons). Example relationships include: parent-child relationships (i.e., a lesson can be a parent of another lesson), child-child relationships (i.e., two lessons are related in some way), include relationships (i.e., a lesson includes another lessons), etc.
- **Operations:** Includes the operations that can be performed on the objects and their relationships. Operation may be simple ones such as: searching, creating, editing, deleting, etc. or more complex operations including but not limited to: inheritance (a lesson inherits all the attributes of a parent lesson), generalisation, specialisation, etc.

Appendix E: Published REFSQ 2013 Paper

This appendix includes the published paper in the proceedings of the 19th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2013), titled “**Maps of Lessons Learnt in Requirements Engineering: A Research Preview**” authored by Noorwali and Madhavji [Noorwali and Madhavji, 2013a]. Copyright permission has been granted by Springer-Verlag to include the paper in the thesis.

Paper starts on the next page

Maps of Lessons Learnt in Requirements Engineering: A Research Preview

Ibtehal Noorwali and Nazim H. Madhavji

University of Western Ontario, London, Canada
`inoorwal@uwo.ca, madhavji@gmail.com`

Abstract. [Context and Motivation] "Those who cannot remember the past are condemned to repeat it" -- George Santayana. From the survey we conducted of requirements engineering (RE) practitioners, over 70% seldom use RE lessons in the RE process, though 85% of these would use such lessons if readily available. Our observation, however, is that, RE lessons are scattered, mainly implicitly, in the literature and practice, which, obviously, does not help the situation. **[Problem/Question]** Approximately 90% of the survey participants stated that not utilising RE lessons has significant negative impact on product quality, productivity, project delays and cost overruns. **[Principal Ideas]** We propose "maps" (or profiles) of RE lessons which, once populated, would highlight weak (dark) and strong (bright) areas of RE (and hence RE theories). Such maps would thus be: (a) a driver for research to "light up" the darker areas of RE and (b) a guide for practice to benefit from the brighter areas. **[Contribution]** The key contribution of this work is the concept of "maps" of RE lessons.

Keywords: requirements engineering, lesson maps, lessons learnt, software quality, empirical study.

1 Introduction

The importance of learning from past experiences has been stressed upon in the literature [1, 5]. Yet, in a survey we conducted of 50 RE practitioners [12], 70% of the respondents indicated that they seldom use RE lessons; 85% of these would use such lessons if readily available; and 90% of them stated that not utilising RE lessons can have significant negative impact on product quality, productivity, project delays and cost overruns. This motivated us to investigate further on the topic of RE lessons.

An important goal of our research is to determine the state of lessons learnt (LL) in RE. LLs can exist in various sources (e.g. literature, project documents, researchers and practitioners, etc.). In attempting to achieve the aforementioned goal, we propose, in this research preview paper, the concept of "*lesson maps*[1]" which, when populated

[1] By "map" we mean "a diagram or collection of data showing the spatial arrangement or distribution of something over an area" (New Oxford American Dictionary). It is not a road map.

with lessons elicited from the literature and practice, would expose weaker (darker) and stronger (brighter) areas of RE. In this paper, we describe the proof of concept of lesson maps with example lessons identified from published literature. The paper does not depict fully populated maps, which is part of our ongoing research. The populated maps are anticipated to promulgate research in the weaker areas and improve practice in the brighter areas of RE.

Section 2 discusses related work. Section 3 describes the concept of lesson maps in requirement engineering. Section 4, gives an example of a sample map. Section 5, discusses the implications of the lesson maps and threats to validity. Section 6 concludes the paper and describes future work.

2 Related Work

Though LL are known in non-software disciplines (such as management [11], education [4], medicine [13], and others), in this section we first touch upon LL in software engineering (SE) followed by LL in RE.

The literature on lessons learnt in SE can be roughly categorized into (i) discovering and sharing lessons learnt and (ii) process and software technologies to support lessons learnt. Examples of the former category include the experience gained at NASA's Software Engineering Laboratory (see Basili et al. [2]) and the experience described by Boehm [5]. Examples of the latter category include: Abdel-Hamid and Madnick's [1] post mortem diagnostic tool to learn from project failures; the approximate reasoning-based approach [15]; Case-Based Reasoning (CBR) approach [14]; and the Experience Factory Framework [3]. The process and software technologies are used in organizational settings.

Unfortunately, in RE not much attention has been paid to lessons learnt. While some literature discusses lessons learnt explicitly [6, 7], much of it is implicit [8] making it difficult to utilise lessons in practice.

3 The Concept of a Map of RE Lessons

In an attempt to create a discipline surrounding lessons learnt in RE, we propose the concept of a *map* of lessons learnt in RE. With reference to the definition of a map in section 1, a map of RE lessons is based on two types of elements: (i) the content (i.e. the lessons) and (ii) the context (i.e. specific attributes selected by the user). Example context attributes are: RE practice, RE phase, process type, application domain, project size, rationale, source, and others. In principle, therefore, it is possible to produce many permutations of lesson maps, e.g.: RE practices; RE practices X RE phases; RE practices X RE phases X application domains; project size X RE phases X sources; application domain X process type; etc. The actual rendering of a map in various permutations is a matter of technological support, which is outside the scope of this concept paper.

After populating a map with some lessons learnt, it can be indicative of the 'state' of lessons learnt in RE (in a project, organisation, body of knowledge, etc.) identified by scarce (dark) and abundant (bright) areas of the map (see Table 1).

Table 1. An example map with context attributes X and Y

Let us assume that context attributes X and Y (selected by the user) are process activities and practices in RE, respectively, where, they are depicted here as a table but could be depicted in another form (e.g. hierarchically). LL1, LL2, etc., are the lessons learnt relating to specific process activities and practices. Examples of dark areas are: X3Y2 and X4Y2 and of bright areas are: X1Y1and X2Y3.

	X1	X2	X3	X4	. . . .
Y1	LL1 LL2 LL3	LL7 LL8	LL13	LL6	
Y2	LL16	LL4 LL5			
Y3	LL17 LL18	LL9 LL10 LL11 LL12	LL14 LL15 LL6	LL6	

4 Example

With reference to Table 2, we can see three lessons spread along RE phases (e.g. elicitation, analysis, etc.): LL1, LL2, LL3.

LL1 [6]: *Lesson:* "Systematically validate and verify requirements by documenting the rationale for requirements." *Related RE phase:* Requirements validation. *Related RE practice:* documentation. *Domain:* enterprise resource planning systems. *Expression:* explicit. *Type:* negative. *Rationale:* Doing so let 39 out of our 67 teams eliminate as much as 43 percent of the stated requirements.

LL2 [6]: *Lesson:* "use prototypes for validation only if you also do process walk-throughs." *Related RE phase:* Requirements validation. *Related RE practice:* prototyping. *Domain:* enterprise resource planning systems. *Expression:* explicit. *Type:* negative. *Rationale:* "In three subprojects, we observed a tendency to rely exclusively on prototypes to negotiate requirements, which led to prototyping spirals in which the teams never built the actual solution."

LL3 [9]: *Lesson:* "use the adjustable requirements reuse approach where requirements can be adjusted without independently creating and storing the different variants of the requirement." *Related RE phase:* elicitation. *Related RE practice:* reuse. *Domain:* Fluid control equipment, pump, seal & valve manufacturing. *Expression:* implicit. *Type:* negative. *Rationale:* "quality and readability of each requirement is improved since it is not split up to a general and a variable part. Since it is not required to document every variation in a separate node it keeps the structure of the requirements much more simple."

With reference to LL1, *Lesson* denotes the content of the lesson. Context attributes are such items as: Related RE phase, Related RE practice, Domain, etc. Expression indicates whether a lesson was explicitly expressed as a lesson learnt in the literature, or the context and surrounding literature had to be analysed to elicit the lesson. Type can mean a positive lesson (one learnt from a successful past experience) or a negative lesson (one learnt from an unsuccessful past experience). There are some other context attributes not included in the lessons here because these are either not known to the creator of the lesson or are empty. Examples are: related lessons involved in solving a particular problem such as hazard analysis in a safety critical system; contradictory lessons; specialization and generalization relationships, etc. LL2 and LL3 have similar structure and attributes.

Assuming the user chooses 'RE phases' from the full set of context attributes, the resultant map would be as shown in Figure 1. With the choice of additional context attributes, the resultant map would contain corresponding entries of lessons. Table 2 shows the map with context attributes 'RE phases' and 'RE practices'.

Upon analysing the map in Table 2, we note that in the elicitation phase, most of the lessons learnt are positive; whereas, in the validation phase, most of the lessons learnt are negative. This could be helpful in the practice of RE. Positive experience, for example, would exude higher confidence in the way elicitation is carried out from descriptive experiences of the RE community; whereas, negative experience would suggest caution in the way requirements are validated. Also, if the lessons identified in the map are found useful in a particular process type (e.g. iterative process), this could lead to savings in costs, time and product quality in other projects in similar process contexts. Caution is in order where process contexts differ (e.g. agile process).

Elicitation	Analysis	Specification	Validation
LL3			LL1
			LL2

Fig. 1. An example of a map of RE lessons with context attribute 'RE phases'

Table 2. An example of a map of RE lessons with context attributes 'RE phases' and 'RE practices'

	Elicitation	Analysis	Specification	Validation
Documentation				LL1
Prototyping				LL2
Using checklists				
Reuse	LL3			

5 Discussion

Implications of this research are anticipated for both practice and research. In industry, use of lesson maps could be felt on project costs, time, and quality. In research, the maps could help in generating new RE theories by identifying weak and strong areas of LL across RE sub-processes and practices. Because patterns and anti-patterns are built upon recurring events, situations, problems, etc., they seem to be good candidates to be associated with lessons learnt in RE.

We identify two threats to validity that may be relevant when building lesson maps: internal (researcher bias) and external validity. Researcher bias can be present during elicitation of lessons learnt from archival sources and practice. External validity can be threatened if the lessons are not generalised enough for use in other contexts. These threats can be mitigated to some degree by obtaining feedback from researchers and practitioners to validate the maps and elicited lessons and by identifying and analyzing the context of each lesson.

6 Conclusion and Future Work

In this research preview, we introduce the concept of *maps* for lessons learnt in requirements engineering. A map consists of actual lessons and the context of these lessons (e.g., RE phases, RE practices, application domains, implicit/explicit lessons, etc.) – see section 3. In section 4, we give an illustrative example of a map (with several lessons [6,9]) that is anticipated to be of benefit to both practitioners and researchers in RE. Based on the concept of the map and the example (described in sections 3 and 4), we conclude that it is a promising stepping-stone towards defining the state of lessons learnt in the field of RE. As next steps in this research, we intend to further explore the concept of the map and subsequently elicit lessons learnt, from various sources in order to gain an understanding of the state of lessons learnt in RE. Further, we have begun to build technological support to operationalise lesson maps for use in RE projects.

References

1. Abdel-Hamid, T.K., Madnick, S.E.: The Elusive Silver Lining: How we Fail to Learn from Software Development Failures. J. MIT Sloan Management Review 32(1), 39–48 (1990)
2. Basili, V.R., McGarry, F.E., Pajerski, R., Zelkowitz, M.V.: Lessons Learned from 25 Years of Process Improvement: The Rise and Fall of the NASA Software Engineering Laboratory. In: International Conference on Software Engineering, pp. 69–79. ACM, Orlando (2002)
3. Basili, V.R., Tesoriero, R., Costa, P., Lindvall, M., Rus, I., Shull, F., Zelkowitz, M.: Building an Experience Base for Software Engineering: A report on the first CeBASE eWorkshop. In: Product-Focused Software Process Improvement, Kaiserslautern, pp. 110–125 (2001)

4. Bodycott, P., Walker, A.: Teaching Abroad: Lessons Learned about Inter-Cultural Understanding for Teachers in Higher Education. J. Teaching in Higher Education 5(1), 79–94 (2000)
5. Boehm, B.: A View of 20th and 21st Century Software Engineering. In: International Conference on Software Engineering, pp. 12–29. ACM, Shanghai (2006)
6. Damian, D.: Stakeholders in Global Requirements Engineering: Lessons Learned from Practice. IEEE Software 24(2), 21–27 (2007)
7. Daneva, M.: ERP Requirements Engineering Practice: Lessons Learned. IEEE Software Journal 21(2), 26–33 (2004)
8. Ebert, C.: Understanding the Product Life Cycle: Four Key Requirements Engineering Techniques. IEEE Software Journal 23(3), 19–25 (2006)
9. Hauksdottir, D., Vermehren, A., Savolainen, J.: Requirements Reuse at Danfoss. In: 20th IEEE Requirements Engineering Conference, pp. 309–314. IEEE, Chicago (2012)
10. Kotonya, G., Sommerville, I.: Requirements Engineering: Processes and Techniques. John Wiley, New York (1998)
11. Lee, M.: Making Lessons Learned a Worthwhile Investment. J. PM World Today 5(7) (2008)
12. Noorwali, I., Madhavji, N.H.: A Survey of Lessons Learnt in Requirements Engineering. Technical Report No. 750, Dept. of Computer Science, University of Western Ontario (2012)
13. Rogers, D.A., Elstein, A.S., Bordage, G.: Improving Continuing Medical Education for Surgical Techniques: Applying the Lessons Learned in the First Decade of Minimal Access Surgery. J. Annals of Surgery 233(2), 159–166 (2001)
14. Sary, C., Mackey, W.: A Case-Based Reasoning Approach for the Access and Reuse of Lessons Learned. In: Fifth Annual Symposium of the National Council on Systems Engineering, St. Louis, pp. 249–256 (1995)
15. Vandeville, J.V., Shaikh, M.A.: A Structured Approximate Reasoning-Based Approach for Gathering "Lessons Learned" Information from System Development Projects. J. Systems Engineering 2(4), 242–247 (1999)
16. Weber, R., Aha, D.W., Becerra-Fernandez, I.: Intelligent Lessons Learned Systems. J. Expert Systems with Applications 20(1), 17–34 (2001)
17. Wellman, J.: Lessons Learned about Lessons Learned. J. Organization Development 25(3), 65–72 (2007)

Bibliography

[Abdelhamid and Madnick, 1990] Abdelhamid, T. K. and Madnick, S. E. (1990). The elusive silver lining: How we fail to learn from software development failures. *MIT Sloan Management Review*, 32(1):39–48.

[Aceituna et al., 2011] Aceituna, D., Do, H., Walia, G. S., and Lee, S.-W. (2011). Evaluating the use of model-based requirements verification method: A feasibility study. In *Proceedings of the 1st International Workshop on Empirical Requirements Engineering (EmpiRE)*, pages 13–20, Trento, Italy. IEEE.

[Adam, 2011] Adam, S. (2011). Towards faster application engineering through better informed elicitation – a research preview. In Fricker, S. and Seyff, N., editors, *Proceedings of the 1st International Requirements Engineering Efficiency Workshop (REEW)*, Essen, Germany.

[Allen, 2003] Allen, R. (2003). *The Penguin English Dictionary*. Penguin, revised edition edition.

[Alrajeh et al., 2012] Alrajeh, D., Kramer, J., van Lamsweerde, A., Russo, A., and Uchitel, S. (2012). Generating obstacle conditions for requirements completeness. In *Proceedings of the 34th International Conference on Software Engineering (ICSE)*, pages 705–715, Zurich. IEEE, IEEE.

[Andrade et al., 2007] Andrade, J., Ares, J., Garcia, R., Pazos, J., Rodriquez, S., Rodriguez-Paton, A., and Silva, A. (2007). Towards a lessons learned system for critical software. *Reliability Engineering and System Safety*, 92(7):902–913.

[Andrade et al., 2013] Andrade, J., Ares, J., Martinez, M.-A., Pazos, J., Rodriquez, S., Romera, J., and Suarez, S. (2013). An architectural model for software testing lesson learned systems. *Information and Software Technology*, 55(1):18–34.

[Anh et al., 2012] Anh, N. D., Cruzex, D. S., Conradi, R., Host, M., Franch, X., and Ayala, C. (2012). Collaborative resolution of requirements mismatches when adopting open source components. In Regnell, B. and Damian, D., editors, *Proceedings of the 18th International Working Conference on Requirements Engineering: Foundation for Software Quality*, LNCS, pages 77–93, Essen, Germany. S.

[Arora et al., 2012] Arora, S., Sampath, P., and S, R. (2012). Resolving uncertainty in automotive feature interactions. In *Proceedings of the 20th IEEE International Requirements Engineering Conference*, pages 21–30, Chicago, Illinois. IEEE, IEEE.

[Asnar et al., 2011] Asnar, Y., Giorgini, P., and Mylopoulos, J. (2011). Goal-driven risk assessment in requirements engineering. *Requirements Engineering*, 16(2):101–116.

[Atladottir et al., 2012] Atladottir, G., Hvannberg, E. T., and Gunnarsdottir, S. (2012). Comparing task practicing and prototype fidelities when applying scenario acting to elicit requirements. *Requirements Engineering*, 17(3):157–170.

[Bahrs and Nguyen, 2011] Bahrs, P. and Nguyen, T. (2011). Smarter architecture and engineering: Game changer for requirements management: A position paper. In *Proceedings of the 1st Workshop on Requirements for Systems, Services, and Systems of Systems*, pages 1–5, Trento, Italy. IEEE.

[Ballejos and Montagna, 2011] Ballejos, L. C. and Montagna, J. M. (2011). Modeling stakeholders for information systems design processes. *Requirements Engineering*, 16(4):281–296.

[Basili et al., 1994] Basili, V. R., Caldiera, G., and Rombach, H. D. (1994). The experience factory. *Encycolpedia of Software Engineering*, 2 Volume Set:469–476.

[Basili et al., 2002] Basili, V. R., McGarry, F. E., Pajerski, R., and Zelkowitz, M. V. (2002). Lessons learned from 25 years of process improvement: The rise and fall of the nasa software engineering laboratory. In *Proceedings of the 24th International Conference on Software Engineering (ICSE)*, pages 69–79, Orlando, Florida, USA. IEEE, IEEE.

[Behnam et al., 2012] Behnam, S. A., Amyot, D., Mussbacher, G., Braun, E., Cartwright, N., and Saucier, M. (2012). Using the goal-oriented pattern family framework for modelling outcome-based regulations. In *Proceedings of the 2nd International Workshop on Requirements Patterns (RePa)*, pages 35–40, Chicago, Illinois. IEEE, IEEE.

[Berenbach, 2012] Berenbach, B. (2012). A 25 year retrospective on model-driven requirements engineering. In *Proceedings of the 2nd International Workshop on Model-Driven Requirements Engineering (MoDRE)*, pages 87–91, Chicago, Illinois. IEEE, IEEE.

[Berenbach et al., 2012] Berenbach, B., Schneider, F., and Naughton, H. (2012). The use of a requirements modeling language for industrial applications. In *Proceedings of the 20th IEEE International Requirements Engineering Conference*, pages 285–290, Chicago, Illinois. IEEE, IEEE.

[Birk and Tautz, 1998] Birk, A. and Tautz, C. (1998). Knowledge management of software engineering lessons learned. Technical Report 002.98/E, Fraunhofer IESE, Germany.

[Bjarnason et al., 2012] Bjarnason, E., Berntsson, S., and Regnell, B. (2012). Evidence-based timelines for project retrospectives a method for assessing requirements engineering in context. In *Proceedings of the 2nd IEEE International Workshop on Empirical Requirements Engineering (EmpiRE)*, pages 17–24, Chicago, Illinois. IEEE, IEEE.

[Bjarnason et al., 2011] Bjarnason, E., Wnuk, K., and Regnell, B. (2011). Requirements are slipping through the gaps - a case study on causes and effects of communication gaps in large-scale software development. In *Proceedings of the 19th International Requirements Engineering Conference*, pages 37–46, Trento, Italy. IEEE, IEEE.

[Bodycott and Walker, 2001] Bodycott, P. and Walker, A. (2001). Teaching abroad: Lessons learned about inter-cultural understanding for teacher in higher education. *Teaching in Higher Education*, 5(1):79–94.

[Boehm, 2006] Boehm, B. W. (2006). A view of 20th and 21st century software engineering. In *Proceedings of the 28th International Conference on Software Engineering (ICSE)*, pages 12–29, Shanghai, China. IEEE, ACM.

[Borges et al., 2011] Borges, R. V., Garcez, A. d., Lamb, L. C., and Nuseibeh, B. (2011). Learning to adapt requirements specifications of evolving systems (nier track). In *Proceedings of the 33rd International Conference on Software Engineering*, pages 856–859, Waikiki, Honolulu. IEEE, IEEE/ACM.

[Bornstein et al., 2004] Bornstein, R. F., Lewis-Beck, M. S., Bryman, A., and Liao, T. F. (2004). *The SAGE Encyclopedia of Social Science Research Methods*. Sage Publications.

[Boulila et al., 2011] Boulila, N., Hoffmann, A., and Herrmann, A. (2011). Using storytelling to record requirements: Elements for an effective requirements elicitation approach. In *Proceedings of the 4th International Workshop on Multimedia and Enjoyable Requirements Engineering (MERE'11)*, pages 9–16, Trento, Italy. IEEE.

[Boutkova, 2011] Boutkova, E. (2011). Pragmatic variability management in requirement specifications with ibm rational doors. In Fricker, S. and Seyff, N., editors, *Proceedings of the 1st International Requirements Engineering Efficiency Workshop (REEW)*, Essen, Germany.

[Boutkova and Houdek, 2011] Boutkova, E. and Houdek, F. (2011). Semi-automatic identification of features in requirement specifications. In *Proceedings of the 19th International Requirements Engineering Conference*, pages 313–318, Trento, Italy. IEEE, IEEE.

[Braun et al., 2012] Braun, E., Cabot, J., Shamsaei, A., Behnam, S. A., Richards, G., Mussbacher, G., Alhaj, M., and Tawhid, R. (2012). Drafting and modeling of regulations: Is it being done backwards? In *Proceedings of the 5th International Workshop on Requirements Engineering and Law (RELAW)*, pages 1–6, Chicago, Illinois. IEEE, IEEE.

[Brill and Knauss, 2011] Brill, O. and Knauss, E. (2011). Structured and unobtrusive observation of anonymous users and their context for requirements elicitation. In *Proceedings of the 19th International Requirements Engineering Conference*, pages 175–184, Trento, Italy. IEEE, IEEE.

[Calefato et al., 2012] Calefato, F., Damian, D., and Lanubile, F. (2012). Computer-mediated communication to support distributed requirements elicitations and negotiations tasks. *Empirical Software Engineering*, 17(6):640–674.

[Cambridge, 2013a] Cambridge (2013a). Cambridge dictionaries online. http://dictionary.cambridge.org/dictionary/british/lesson.

[Cambridge, 2013b] Cambridge (2013b). Cambridge dictionaries online. http://machaut.uchicago.edu/cgi-bin/WEBSTER.sh?WORD=lesson.

[Carvallo and Franch, 2011] Carvallo, J. P. and Franch, X. (2011). Requirements negotiation for multilayer system components. In *Proceedings of the 19th International Requirements Engineering Conference*, pages 285–290, Trento, Italy. IEEE, IEEE.

[Charrada et al., 2012] Charrada, E. B., Koziolek, A., and Glinz, M. (2012). Identifying outdated requirements based on source code changes. In *Proceedings of the 20th IEEE International Requirements Engineering Conference*, pages 61–70, Chicago, Illinois. IEEE, IEEE.

[Chen et al., 2011] Chen, B., Peng, X., Yu, Y., and Zhao, W. (2011). Are your sites down? requirements-driven self-tuning for the survivability of web systems*. In *Proceedings of the 19th International Requirements Engineering Conference*, pages 219–228, Trento, Italy. IEEE, IEEE.

[Chernak, 2012] Chernak, Y. (2012). Requirements composition table explained. In *Proceedings of the 20th IEEE International Requirements Engineering Conference*, pages 273–278, Chicago, Illinois. IEEE, IEEE.

[Chillarege, 1999] Chillarege, R. (1999). Software testing best practices. Technical report, IBM Research.

[Chopra and Singh, 2011] Chopra, A. K. and Singh, M. P. (2011). Colaba: Collaborative design of cross-organizational processes. In *Proceedings of the 1st Workshop on Requirements for Systems, Services, and Systems of Systems*, pages 36–43, Trento, Italy. IEEE.

[Cleland-Huang et al., 2012] Cleland-Huang, J., Mader, P., Mirakholi, M., and Amornborvornwong, S. (2012). Breaking the big-bang practice of traceability: Pushing timely trace recommendations to project stakeholders. In *Proceedings of the 20th IEEE International Requirements Engineering Conference*, pages 231–240, Chicago, Illinois. IEEE, IEEE.

[Collins, 2013] Collins (2013). Collins dictionary. http://www.collinsdictionary.com/dictionary/english/lesso

[Cortier et al., 2011] Cortier, V., Detrey, J., Gaudry, P., Sur, F., and Thome, E. (2011). Ballot stuffing in a postal voting system. In *Proceedings of the 1st Workshop on Requirements Engineering for E-Voting Systems*, pages 27–36, Trento, Italy. IEEE.

[Creswell, 2008] Creswell, J. W. (2008). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. Sage Publications, 3rd edition.

[Damian, 2007] Damian, D. (2007). Stakeholders in global requirements engineering: Lessons learned from practice. *IEEE Software Journal*, 24(2):21–27.

[Daneva, 2004] Daneva, M. (2004). Erp requirements engineering practice: Lessons learned. *IEEE Software Journal*, 21(2):26–33.

[Daramola et al., 2012] Daramola, O., Sindre, G., and Stalhane, T. (2012). Pattern-based security requirements specification using ontologies and boilerplates. In *Proceedings of the 2nd*

International Workshop on Requirements Patterns (RePa), pages 54–59, Chicago, Illinois. IEEE, IEEE.

[Daramola et al., 2011] Daramola, O., Stalhane, T., Sindre, G., and Omoronyia, I. (2011). Enabling hazard identification from requirements and reuse-oriented hazop analysis. In *Proceedings of the 4th International Workshop on Managing Requirements Knowledge*, pages 3–11, Trento, Italy. IEEE.

[Dekhtyar et al., 2011] Dekhtyar, A., Dekhtyar, O., Holden, J., Hayes, J. H., Cuddeback, D., and Kong, W.-K. (2011). On human analyst performance in assisted requirements tracing: Statistical analysis. In *Proceedings of the 19th International Requirements Engineering Conference*, pages 111–120, Trento, Italy. IEEE, IEEE.

[Dick and Woods, 1997] Dick, J. and Woods, E. (1997). Lessons learned from rigorous system software development. *Information and Software Technology*, 39(8):551–560.

[Dictionary.com, 2013a] Dictionary.com (2013a). Dictionary.com. http://dictionary.reference.com/browse/lesson.

[Dictionary.com, 2013b] Dictionary.com (2013b). Dictionary.com. http://dictionary.reference.com/browse/map?s=t.

[Dieste and Juristo, 2011] Dieste, O. and Juristo, N. (2011). Systematic review and aggregation of empirical studies on elicitation techniques. *IEEE Transactions on Software Engineering*, 37(2):283–304.

[Dietsch et al., 2011] Dietsch, D., Arenis, S. F., Westphal, B., and Podelski, A. (2011). Disambiguation of industrial standards through formalization and graphical languages. In *Proceedings of the 19th International Requirements Engineering Conference*, pages 265–270, Trento, Italy. IEEE, IEEE.

[DOE-STD-7501-99, 1999] DOE-STD-7501-99 (1999). The doe corporate lessons learned program. Technical Report DOE-STD-7501-99, United States Department of Energy, Washington, DC.

[El-Sharkawy and Schmid, 2011] El-Sharkawy, S. and Schmid, K. (2011). A heuristic approach for supporting product innovation in requirements engineering: A controlled experiment. In *Proceedings of the 17th International Working Conference on Requirements Engineering: Foundation for Software Quality*, LNCS, pages 78–93, Essen, Germany. Springer-Verlag.

[Engelsman and Wieringa, 2012] Engelsman, W. and Wieringa, R. (2012). Goal-oriented requirements engineering and enterprise architecture: Two case studies and some lessons learned. In Regnell, B. and Damian, D., editors, *Proceedings of the 18th International Working Conference on Requirements Engineering: Foundation for Software Quality*, LNCS, pages 306–320, Essen, Germany. Springer-Verlag.

[Ernst et al., 2011] Ernst, N. A., Borgida, A., and Jureta, I. J. (2011). Finding incremental solutions for evolving requirements. In *Proceedings of the 19th International Requirements Engineering Conference*, pages 15–24, Trento, Italy. IEEE, IEEE.

[Erra and Scanniello, 2011] Erra, U. and Scanniello, G. (2011). Assessing think-pair-square in distributed modeling of use case diagrams. In *Proceedings of the 1st International Workshop on Empirical Requirements Engineering (EmpiRE)*, pages 77–84, Trento, Italy. IEEE.

[Fitzgerald et al., 2012] Fitzgerald, C., Letier, E., and Finkelstein, A. (2012). Early failure prediction in feature request management systems: An extended study. *Requirements Engineering*, 17(2):117–132.

[Fricker and Schumacher, 2012] Fricker, S. and Schumacher, S. (2012). Release planning with feature trees: Industrial case. In Regnell, B. and Damian, D., editors, *Proceedings of the 18th*

International Working Conference on Requirements Engineering: Foundation for Software Quality, LNCS, pages 288–305, Essen, Germany. Springer-Verlag.

[Gacitua et al., 2011] Gacitua, R., Sawyer, P., and Gervasi, V. (2011). Relevance-based abstraction identification: Technique and evaluation. *Requirements Engineering*, 16(3):251–265.

[Gervasi and Zowghi, 2011] Gervasi, V. and Zowghi, D. (2011). Mining requirements links. In *Proceedings of the 17th International Working Conference on Requirements Engineering: Foundation for Software Quality*, LNCS, pages 196–201, Essen, Germany. Springer-Verlag.

[Gibson et al., 2011] Gibson, J. P., MacNamara, D., and Ken, O. (2011). Just like paper and the 3-colour protocol: A voting interface requirements engineering case study. In *Proceedings of the 1st Workshop on Requirements Engineering for E-Voting Systems*, pages 66–75, Trento, Italy. IEEE.

[Gonzales and Leroy, 2011] Gonzales, C. K. and Leroy, G. (2011). Eliciting user requirements using appreciative inquiry. *Empirical Software Engineering*, 16(6):733–772.

[Gordon and Breaux, 2011] Gordon, D. G. and Breaux, T. D. (2011). Comparing requirements from multiple jurisdictions. In *Proceedings of the 4th International Workshop on Requirements Engineering and Law (RELAW)*, pages 43–49, Trento, Italy. IEEE.

[Gordon and Breaux, 2012] Gordon, D. G. and Breaux, T. D. (2012). Reconciling multi-jurisdictional legal requirements: A case study in requirements water marking. In *Proceedings of the 20th IEEE International Requirements Engineering Conference*, pages 91–100, Chicago, Illinois. IEEE, IEEE.

[Goulao et al., 2011] Goulao, M., Moreira, A., Araujo, J., and Santos, J. P. (2011). Streamlining scenario modeling with model-driven development: a case study. In *Proceedings of the Model-Driven Requirements Engineering Workshop (MoDRE)*, pages 55–63, Trento, Italy. IEEE.

[Greenyer et al., 2012] Greenyer, J., Sharifloo, A. M., Cordy, M., and Heymans, P. (2012). Efficient consistency checking of scenario-based product-line specifications. In *Proceedings of the 20th IEEE International Requirements Engineering Conference*, pages 161–170, Chicago, Illinois. IEEE, IEEE.

[Gross and Doerr, 2012] Gross, A. and Doerr, J. (2012). What you need is what you get! the vision of view-based requirements specifications. In *Proceedings of the 20th IEEE International Requirements Engineering Conference*, pages 171–180, Chicago, Illinois. IEEE, IEEE.

[Gross et al., 2012] Gross, A., Jurkiewicz, J., Doerr, J., and Nawrocki, J. (2012). Investigating the usefulness of notations in the context of requirements engineering: Research agenda and lessons learned. In *Proceedings of the 2nd IEEE International Workshop on Empirical Requirements Engineering (EmpiRE)*, pages 9–16, Chicago, Illinois. IEEE, IEEE.

[Hauksdottir et al., 2012] Hauksdottir, D., Vermehren, A., and Savolainen, J. (2012). Requirements reuse at danfoss. In *Proceedings of the 20th IEEE International Requirements Engineering Conference*, pages 309–314, Chicago, Illinois. IEEE, IEEE.

[Heaven and Letier, 2011] Heaven, W. and Letier, E. (2011). Simulating and optimising design decisions in quantitative goal models. In *Proceedings of the 19th International Requirements Engineering Conference*, pages 79–88, Trento, Italy. IEEE, IEEE.

[Helferich and Mautsch, 2011] Helferich, A. and Mautsch, L. O. (2011). Defining product lines and product variants based on prioritization of customer seg- ments and customer requirements. In Krams, B. and Schockert, S., editors, *Proceedings of the 2nd Workshop on Requirements Prioritization for Customer-Oriented Software- Development (RePriCo)*, Essen, Germany.

[Hoffmann et al., 2011] Hoffmann, A., Peters, C., and Leimeister, J. M. (2011). Improving corporate portal design by using service- oriented requirements engineering and service bundling. In *Proceedings of the 1st Workshop on Requirements for Systems, Services, and Systems of Systems*, pages 44–49, Trento, Italy. IEEE.

[Huber, 1991] Huber, G. P. (1991). Organizational learning: The contributing processes and the literatures. *Organization Science*, 2(1):88–115.

[Hussain et al., 2012] Hussain, I., Ormandjieva, O., and Kosseim, L. (2012). Lasr: A tool for large scale annotation of software requirements. In *Proceedings of the 2nd IEEE International Workshop on Empirical Requirements Engineering (EmpiRE)*, pages 57–60, Chicago, Illinois. IEEE, IEEE.

[Isaacs and Berry, 2011] Isaacs, D. and Berry, D. M. (2011). Developers want requirements, but their project manager doesn't; and a possibly transcendent hawthorne effect. In *Proceedings of the 1st International Workshop on Empirical Requirements Engineering (EmpiRE)*, pages 37–44, Trento, Italy. IEEE.

[Jalili et al., 2011] Jalili, Y. A., Nasrin, S., Fakhimi, S., and Khakhian, Y. B. (2011). How can we systematically manage lessons learned in projects? In *Proceedings of the 2nd International Conference on Construction and Project Management*, volume 15, Singapore. IACSIT Press.

[Johnson and Christensen, 2007] Johnson, B. and Christensen, L. B. (2007). *Educational Research: Quantitative, Qualitative, and Mixed Approaches*. Sage Publications.

[Jones, 2011] Jones, P. (2011). Can requirements tool vendors tell us about user needs? In *Proceedings of the 4th International Workshop on Managing Requirements Knowledge*, pages 31–34, Trento, Italy. IEEE.

[Kamalrudin et al., 2011] Kamalrudin, M., Hosking, J., and Grundy, J. (2011). Improving requirements quality using essential use case interaction patterns. In *Proceedings of the 33rd International Conference on Software Engineering*, pages 531–540, Waikiki, Honolulu. IEEE, IEEE/ACM.

[Kaner et al., 2001] Kaner, C., Bach, J., and Pettichord, B. (2001). *Lessons Learned in Software Testing: A Context Driven Approach*. Wiley, 1 edition.

[Kitchenham, 2004] Kitchenham, B. (2004). Procedures for performing systematic reviews. Joint Technical Report TR/SE-0401, Keele University, Australia.

[Kitchenham et al., 2002] Kitchenham, B., Pfleeger, S. L., Hoaglin, Pickard, L. M., Jones, P. W., C., D., El-Emam, K., and Rosenberg, J. (2002). Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on Software Engineering*, 28(8):721–734.

[Knauss et al., 2012] Knauss, E., Damian, D., Poo-Caamano, G., and Cleland-Huang, J. (2012). Detecting and classifying patterns of requirements clarifications. In *Proceedings of the 20th IEEE International Requirements Engineering Conference*, pages 251–260, Chicago, Illinois. IEEE, IEEE.

[Knauss et al., 2011] Knauss, E., Houmb, S., and Schneider, K. (2011). 4 supporting requirements engineers in recognising security issues. In *Proceedings of the 17th International Working Conference on Requirements Engineering: Foundation for Software Quality*, LNCS, pages 4–18, Essen, Germany. Springer-Verlag.

[Knauss and Schneider, 2012] Knauss, E. and Schneider, K. (2012). Supporting learning organisations in writing better requirements documents based on heuristic critiques. In Regnell, B. and Damian, D., editors, *Proceedings of the 18th International Working Conference on Requirements Engineering: Foundation for Software Quality*, LNCS, pages 165–171, Essen, Germany. Springer-Verlag.

[Kof and Penzenstadler, 2011] Kof, L. and Penzenstadler, B. (2011). Faster from requirements documents to system models: Interactive semi-automatic translation. In Fricker, S. and Seyff, N., editors, *Proceedings of the 1st International Requirements Engineering Efficiency Workshop (REEW)*, Essen, Germany.

[Kong and Hayes, 2011] Kong, W.-K. and Hayes, J. H. (2011). Proximity-based traceability: An empirical validation using ranked retrieval and set-based measures. In *Proceedings of the 1st International Workshop on Empirical Requirements Engineering (EmpiRE)*, pages 45–52, Trento, Italy. IEEE.

[Kotonya and Sommerville, 1998] Kotonya, G. and Sommerville, I. (1998). *Requirements Engineering: Processes and Techniques*. John Wiley and Sons, England.

[Kukreja et al., 2012] Kukreja, N., Payyavula, S. S., Boehm, B. W., and Padmanabhuni, S. (2012). Selecting an appropriate framework for value-based requirements prioritization: A case study. In *Proceedings of the 20th IEEE International Requirements Engineering Conference*, pages 303–308, Chicago, Illinois. IEEE, IEEE.

[Lauesen, 2012] Lauesen, S. (2012). Why the electronic land registry failed. In Regnell, B. and Damian, D., editors, *Proceedings of the 18th International Working Conference on Requirements Engineering: Foundation for Software Quality*, LNCS, pages 1–15, Essen, Germany. Springer-Verlag.

[Lauesen and Kuhail, 2012] Lauesen, S. and Kuhail, M. A. (2012). Task descriptions versus use cases. *Requirements Engineering*, 17(1):3–18.

[Lee, 2008] Lee, M. (2008). Making lessons learned a worthwhile investment. *PM World Today*, 10(7).

[Li et al., 2011] Li, Y., Narayan, N., Helming, J., and Maximilian, K. (2011). A domain specific requirements model for scientific computing (nier track). In *Proceedings of the 33rd International Conference on Software Engineering*, pages 848–851, Waikiki, Honolulu. IEEE, IEEE/ACM.

[Liaskos et al., 2011] Liaskos, S., McIlraith, S. A., Sohrabi, S., and Mylopoulos, J. (2011). Representing and reasoning about preferences in requirements engineering. *Requirements Engineering*, 16(3):227–249.

[Lindvall et al., 2001] Lindvall, M., Frey, M., Costa, P., and Tesoriero, R. (2001). Lessons learned about structuring and describing experience for three experience bases. In *Proceedings of the Third International Workshop on Advances in Learning Software Organizations*, LSO '01, pages 106–119, London, UK. Springer-Verlag.

[Loconsole et al., 2011] Loconsole, A., Gruber, H., Nae, A., and Regnell, B. (2011). Construction and evaluation of an algorithmic and distributed prioritization method. In Krams, B. and Schockert, S., editors, *Proceedings of the 2nd Workshop on Requirements Prioritization for Customer-Oriented Software- Development (RePriCo)*, Essen, Germany.

[Loft et al., 2012] Loft, M. S., Nielsen, S. S., Norskov, K., and Jorgensen, J. B. (2012). Interplay between requirements, software architecture, and hardware constraints in the development of a home control user interface. In *Proceedings of the 1st International Workshop on the Twin Peaks of Requirements and Architecture (TwinPeaks)*, pages 1–6, Chicago, Illinois. IEEE, IEEE.

[Luna et al., 2011] Luna, E. R., Rossi, G., and Garrigos, I. (2011). Webspec: A visual language for specifying interaction and navigation requirements in web applications. *Requirements Engineering*, 16(4):297–321.

[Lutz et al., 2012] Lutz, R., Lutz, J., Lathrop, J., Klinge, T., Mathur, D., Stull, D., Bergquist, T., and Henderson, E. (2012). Requirements analysis for a product family of dna nanodevices. In *Proceedings of the 20th IEEE International Requirements Engineering Conference*, pages 211–220, Chicago, Illinois. IEEE, IEEE.

[Macmillan, 2013a] Macmillan (2013a). Macmillan dictionary. http://www.macmillandictionary.com/dictionary/american/lesson.

[Macmillan, 2013b] Macmillan (2013b). Macmillan dictionary. http://www.macmillandictionary.com/dictionary/british/map.

[Mahaux et al., 2011] Mahaux, M., Heymans, P., and Saval, G. (2011). Discovering sustainability requirements: An experience report. In *Proceedings of the 17th International Working Conference on Requirements Engineering: Foundation for Software Quality*, LNCS, pages 19–33, Essen, Germany. Springer-Verlag.

[Marincic et al., 2011] Marincic, J., Mader, A., and Wieringa, R. (2011). Validation of embedded system verification models. In *Proceedings of the Model-Driven Requirements Engineering Workshop (MoDRE)*, pages 48–54, Trento, Italy. IEEE.

[Markov et al., 2011] Markov, G. A., Hoffmann, A., and Creighton, O. (2011). Requirements engineering process improvement: An industrial case study. In *Proceedings of the 17th International Working Conference on Requirements Engineering: Foundation for Software Quality*, LNCS, pages 34–47, Essen, Germany. Springer-Verlag.

[Mashkoor and Jacquot, 2011] Mashkoor, A. and Jacquot, J.-P. (2011). Utilizing event-b for domain engineering: A critical analysis. *Requirements Engineering*, 16(3):191–207.

[Massacci et al., 2012] Massacci, F., Nagaraj, D., Paci, F., Tran, L. M. S., and Tedeschi, A. (2012). Assessing a requirements evolution approach: Empirical studies in the air traffic management domain. In *Proceedings of the 2nd IEEE International Workshop on Empirical Requirements Engineering (EmpiRE)*, pages 49–56, Chicago, Illinois. IEEE, IEEE.

[Massey et al., 2011] Massey, A. K., Smith, B., Otto, P. N., and Anton, A. I. (2011). Assessing the accuracy of legal implementation readiness decisions. In *Proceedings of the 19th International Requirements Engineering Conference*, pages 207–216, Trento, Italy. IEEE, IEEE.

[Maxwell et al., 2012a] Maxwell, J. C., Anton, A. I., and Swire, P. (2012a). Managing changing compliance requirements by predicting regulatory evolution: An adaptability framework. In *Proceedings of the 20th IEEE International Requirements Engineering Conference*, pages 101–110, Chicago, Illinois. IEEE, IEEE.

[Maxwell et al., 2012b] Maxwell, J. C., Anton, A. I., Swire, P., Riaz, M., and McGraw, C. M. (2012b). A legal cross-references taxonomy for reasoning about compliance requirements. *Requirements Engineering*, 17(2):99–115.

[McGee and Greer, 2012] McGee, S. and Greer, D. (2012). Towards an understanding of the causes and effects of software requirements change: Two case studies. *Requirements Engineering*, 17(2):133–155.

[McLaughlin, 1987] McLaughlin, M. W. (1987). Learning from experience: Lessons from policy implementation. *Educational Evaluation and Policy Analysis*, 9(2):171–178.

[Mendizabal et al., 2012] Mendizabal, O. M., Spier, M., and Saad, R. (2012). Log-based approach for performance requirements elicitation and prioritization. In *Proceedings of the 20th IEEE International Requirements Engineering Conference*, pages 297–302, Chicago, Illinois. IEEE, IEEE.

[Merriam-Webster, 2013] Merriam-Webster (2013). Merriam-webster. http://www.merriam-webster.com/dictionary/lesson.

[Merten et al., 2011] Merten, T., Juppner, D., and Delater, A. (2011). Improved representation of traceability links in requirements engineering knowledge using sunburst and netmap visualizations. In *Proceedings of the 4th International Workshop on Managing Requirements Knowledge*, pages 17–21, Trento, Italy. IEEE.

[Merten et al., 2012] Merten, T., Schafer, T., and Bursner, S. (2012). Using re knowledge to assist automatically during requirement specification. In *Proceedings of the 7th International Workshop on Requirements Engineering Education and Training (REET 2012)*, pages 9–13, Chicago, Illinois. IEEE, IEEE.

[Milne and Maiden, 2012] Milne, A. and Maiden, N. (2012). Power and politics in requirements engineering: Embracing the dark side? *Requirements Engineering*, 17(2):83–98.

[Morales-Ramirez et al., 2012] Morales-Ramirez, I., Vergne, M., Morandini, M., and Sabatucci, L. (2012). Revealing the obvious? a retrospective artefact analysis for an ambient assisted-living project. In *Proceedings of the 2nd IEEE International Workshop on Empirical Requirements Engineering (EmpiRE)*, pages 41–48, Chicago, Illinois. IEEE, IEEE.

[Morandini et al., 2011] Morandini, M., Marchetto, A., and Perini, A. (2011). Requirements comprehension: A controlled experiment on conceptual modeling methods. In *Proceedings of the 1st International Workshop on Empirical Requirements Engineering (EmpiRE)*, pages 53–60, Trento, Italy. IEEE.

[Niknafs and Berry, 2012] Niknafs, A. and Berry, D. M. (2012). The impact of domain knowledge on the effectiveness of requirements idea generation during requirements elicitation. In *Proceedings of the 20th IEEE International Requirements Engineering Conference*, pages 181–190, Chicago, Illinois. IEEE, IEEE.

[Niu and Mahmoud, 2012] Niu, N. and Mahmoud, A. (2012). Enhancing candidate link generation for requirements tracing: The cluster hypothesis revisited. In *Proceedings of the 20th IEEE International Requirements Engineering Conference*, pages 81–90, Chicago, Illinois. IEEE, IEEE.

[Nolan et al., 2011] Nolan, A. J., Abrahao, S., Clements, P., and Pickard, A. (2011). Managing requirements uncertainty in engine control systems development. In *Proceedings of the 19th International Requirements Engineering Conference*, pages 259–264, Trento, Italy. IEEE, IEEE.

[Noorwali and Madhavji, 2012] Noorwali, I. and Madhavji, N. H. (2012). A survey of lessons learnt in requirements engineering. Technical Report 750, University of Western Ontario, London, Ontario.

[Noorwali and Madhavji, 2013a] Noorwali, I. and Madhavji, N. H. (2013a). Lessons learnt in requirements engineering: A research preview. In Doerr, J. and Opdahl, A., editors, *Proceedings of the 19th International Working Conference on Requirements Engi- neering: Foundation for Software Quality (REFSQ 13)*, LNCS 7830, pages 119–124, Essen, Germany. Springer-Verlag Berlin Heidelberg.

[Noorwali and Madhavji, 2013b] Noorwali, I. and Madhavji, N. H. (2013b). Lessons learnt in requirements engineering: A survey. http://edu.surveygizmo.com/s3/1131891/A-Survey-of-Lessons-Learnt-in-Requirements-Engineering.

[Ohashi et al., 2011] Ohashi, K., Kurihara, H., Tananaka, Y., and Yamamoto, R. (2011). A means of establishing traceability based on a uml model in business application development. In *Proceedings of the 19th International Requirements Engineering Conference*, pages 279–284, Trento, Italy. IEEE, IEEE.

[Olson and Lyles, 2011] Olson, B. J. and Lyles, A. (2011). Biofuels licensing: Lessons learned from the monoclonal antibody business cycle. *The Licensing Journal*, 31(9):1–6.

[Oxford, 2004] Oxford (2004). *The Oxford ESL Dictionary*. Oxford University Press, New York.

[Oxford, 2013a] Oxford (2013a). Oxford advanced learner's dictionary. http://oald8.oxfordlearnersdictionaries.com/dictionary/lesson.

[Oxford, 2013b] Oxford (2013b). Oxford dictionaries. http://oxforddictionaries.com/definition/english/lesson.

[Oxford, 2013c] Oxford (2013c). Oxford dictionaries. http://oxforddictionaries.com/definition/english/map?q=map.

[Pasquale and Spoletini, 2011] Pasquale, L. and Spoletini, P. (2011). Monitoring fuzzy temporal requirements for service compositions: Motivations, challenges and experimental results*. In *Proceedings of the 1st Workshop on Requirements for Systems, Services, and Systems of Systems*, pages 63–69, Trento, Italy. IEEE.

[Patton, 2001] Patton, M. Q. (2001). Evaluation, knowledge management, best practices, and high quality lessons learned. *American Journal of Evaluation*, 22(3):329–336.

[Pearsall and Trumble, 1995] Pearsall, J. and Trumble, B. (1995). *The Oxford Encyclopedic English Dictionary*. Oxford University Press, New York, second edition.

[Penzenstadler and Eckhardt, 2012] Penzenstadler, B. and Eckhardt, J. (2012). A requirements engineering content model for cyber-physical systems. In *Proceedings of the 2nd International Workshop on Requirements Engineering for Systems and Systems and Quality Requirements (RESS)*, pages 20–29, Chicago, Illinois. IEEE, IEEE.

[Petrov et al., 2012] Petrov, P., Buy, U., and Nord, R. L. (2012). Enhancing the software architecture analysis and design process with inferred macro-architectural requirements. In *Proceedings of the 1st International Workshop on the Twin Peaks of Requirements and Architecture (TwinPeaks)*, pages 20–26, Chicago, Illinois. IEEE, IEEE.

[Pires et al., 2011] Pires, P. F., Delicato, F. C., Cobe, R., Batista, T., Davis, J. G., and Song, J. H. (2011). Integrating ontologies, model mriven, and cnl in a multi-viewed approach for requirements engineering. *Requirements Engineering*, 16(2):133–160.

[Pitula and Radhakrishna, 2011] Pitula, K. and Radhakrishna, T. (2011). On eliciting requirements from end-users in the ict4d domain. *Requirements Engineering*, 16(4):323–351.

[Poort et al., 2012] Poort, E. R., Martens, N., Weerd, I. v. d., and Vliet, H. v. (2012). How architects see non-functional requirements: Beware of modifiability. In *Proceedings of the 18th International Working Conference on Requirements Engineering: Foundation for Software Quality*, LNCS, pages 37–51, Essen, Germany. Springer-Verlag.

[Post et al., 2012] Post, A., Menzel, I., Hoenicke, J., and Podelski, A. (2012). Automotive behavioral requirements expressed in a specification pattern system: A case study at bosch. *Requirements Engineering*, 17(1):19–33.

[Post et al., 2011] Post, A., Menzel, I., and Podelski, A. (2011). Applying restricted english grammar on automotive requirements—does it work? a case study. In *Proceedings of the 17th International Working Conference on Requirements Engineering: Foundation for Software Quality*, LNCS, pages 166–180, Essen, Germany. Springer-Verlag.

[Raspotnig and Opdahl, 2012] Raspotnig, C. and Opdahl, A. (2012). Supporting failure mode and effect analysis: A case study with failure sequence diagrams. In Regnell, B. and Damian, D., editors, *Proceedings of the 18th International Working Conference on Requirements Engineering: Foundation for Software Quality*, LNCS, pages 117–131, Essen, Germany. Springer-Verlag.

[Rauf et al., 2011] Rauf, R., Antkiewicz, M., and Czarnecki, K. (2011). Logical structure extraction from software requirements documents. In *Proceedings of the 19th International Requirements Engineering Conference*, pages 101–110, Trento, Italy. IEEE, IEEE.

[Reggio et al., 2011] Reggio, G., Leotta, M., and Ricca, F. (2011). "precise is better than light" a document analysis study about quality of business process models. In *Proceedings of the 1st International Workshop on Empirical Requirements Engineering (EmpiRE)*, pages 61–68, Trento, Italy. IEEE.

[Reifer et al., 2004] Reifer, D. J., Basili, V. R., Boehm, B. W., and Clark, B. (2004). Cots-based systems - twelve lessons learned about maintenance. *LNCS COTS-Based Software Systems*, 2959:137–145.

[Rogers et al., 2001] Rogers, D. A., Elstein, A. S., and Borgade, G. (2001). Improving continuing medical education for surgical techniques: Applying the lessons learned in the first decade of minimal access surgery. *Annals of Surgery*, 233(2):159–166.

[Runeson and Host, 2009] Runeson, P. and Host, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131–164.

[Sakhnini et al., 2012] Sakhnini, V., Mich, L., and Berry, D. M. (2012). The effectiveness of an optimized epmcreate as a creativity enhancement technique for web site requirements elicitation. *Requirements Engineering*, 17(3):171–186.

[Salfischberger et al., 2011] Salfischberger, T., van de Weerd, I., and Brinkkemper, S. (2011). The functional architecture framework for organizing high volume requirements management. In *Proceedings of the 5th International Workshop on Software Product Management (IWSPM)*, pages 17–26, Trento, Italy. IEEE.

[Salkind, 2007] Salkind, N. J. (2007). *Encyclopedia of Measurement and Statistics*. Sage Publications.

[Sampath et al., 2011] Sampath, P., Arora, S., and S, R. (2011). Evolving specifications formally. In *Proceedings of the 19th International Requirements Engineering Conference*, pages 5–14, Trento, Italy. IEEE, IEEE.

[Sapkota et al., 2012] Sapkota, K., Aldea, A., Muhammad, Y., Duce, D. A., and Banares-Alcantara, R. (2012). Extracting meaningful entities from regulatory text: Towards automating regulatory compliance. In *Proceedings of the 5th International Workshop on Requirements Engineering and Law (RELAW)*, pages 29–32, Chicago, Illinois. IEEE, IEEE.

[Sary and Mackey, 1995] Sary, C. and Mackey, W. (1995). A case-based reasoning approach for the access and reuse of lessons learned. In *Proceedings of the 5th Annual Symposium of the National Council on Systems Engineering*, pages 249–256, St.Louis.

[Savio and P.C., 2012] Savio, D. and P.C., A. (2012). 'pictionades': Enhancing stakeholders' awareness about issues in requirements communication. In Daneva, M., Doerr, J., Hermann, A., and Schneider, K., editors, *Proceedings of the 2nd Workshop on Creativity in Requirements Engineering*, LNCS, pages 105–113, Essen, Germany. Springer-Verlag.

[Savio and Suryanarayana, 2012] Savio, D. and Suryanarayana, G. (2012). How to avoid taking three lefts when you can go right: Making the architectural perspective count. In *Pro-*

ceedings of the 1st International Workshop on the Twin Peaks of Requirements and Architecture (TwinPeaks), pages 31–35, Chicago, Illinois. IEEE, IEEE.

[Schmidt et al., 2012] Schmidt, J. Y., Anton, A. I., and Earp, J. B. (2012). Assessing identification of compliance requirements from privacy policies. In *Proceedings of the 5th International Workshop on Requirements Engineering and Law (RELAW)*, pages 52–61, Chicago, Illinois. IEEE, IEEE.

[Schmidt et al., 2011] Schmidt, J. Y., Anton, A. I., Williams, L., and Ott, D. (2011). The role of data use agreements in specifying legally compliant software requirements. In *Proceedings of the 4th International Workshop on Requirements Engineering and Law (RELAW)*, pages 1–4, Trento, Italy. IEEE.

[Schneider, 2011] Schneider, K. (2011). Focusing spontaneous feedback to support system evolution. In *Proceedings of the 19th International Requirements Engineering Conference*, pages 165–174, Trento, Italy. IEEE, IEEE.

[Schneider et al., 2012] Schneider, K., Knauss, E., Houmb, S., Islam, S., and Jurjens, J. (2012). Enhancing security requirements engineering by organizational learning. *Requirements Engineering*, 17(1):35–56.

[Secchi et al., 1999] Secchi, P., Ciaschi, R., and Spence, D. (1999). A concept for an esa lessons learned system. In *Proceedings of Alerts and LL: An Effective Way to Prevent Failures and Problems*, number Technical Report WPP-167, pages 57–61, ESTEC: Noordwijk, The Netherlands.

[Shaker et al., 2012] Shaker, P., Atlee, J. M., and Wang, S. (2012). A feature-oriented requirements modelling language. In *Proceedings of the 20th IEEE International Requirements Engineering Conference*, pages 151–160, Chicago, Illinois. IEEE, IEEE.

[Sharma and Biswas, 2011] Sharma, R. and Biswas, K. (2011). Using courteous logic based representations for requirements specification. In *Proceedings of the 4th International Workshop on Managing Requirements Knowledge*, pages 12–16, Trento, Italy. IEEE.

[Sharma and Biswas, 2012] Sharma, R. and Biswas, K. (2012). Using norm analysis patterns for automated requirements validation. In *Proceedings of the 2nd International Workshop on Requirements Patterns (RePa)*, pages 23–28, Chicago, Illinois. IEEE, IEEE.

[Sim and Alspaugh, 2011] Sim, S. E. and Alspaugh, T. A. (2011). Getting the whole story: An experience report on analyzing data elicited using the war stories procedure. *Empirical Software Engineering*, 16(4):460–486.

[Smialek et al., 2012] Smialek, M., Nowakowski, W., Jarzebowski, N., and Ambroziewicz, A. (2012). From use cases and their relationships to code. In *Proceedings of the 2nd International Workshop on Model-Driven Requirements Engineering (MoDRE)*, pages 9–18, Chicago, Illinois. IEEE, IEEE.

[Sommerville, 2011] Sommerville, I. (2011). *Software Engineering*. Pearson, Boston, MA, 9th edition edition.

[Sommerville and Sawyer, 1997] Sommerville, I. and Sawyer, P. (1997). *Requirement Engineering: A Good Practice Guide*. Wiley, 1 edition.

[Summers, 1987] Summers, D. (1987). *Longman Dictionary of Contemporary English*. Longman, Essex, London, new edition edition.

[Sunindyo et al., 2011] Sunindyo, W., Melik-Merkumians, M., Moser, T., and Biffl, S. (2011). Enforcing safety requirements for industrial automation systems at runtime: Position paper. In *Proceedings of the 2nd Workshop requirements@run.time*, pages 37–42, Trento, Italy. IEEE.

[Sutcliffe et al., 2011] Sutcliffe, A., Thew, S., and Jarvis, P. (2011). Experience with user-centred requirements engineering. *Requirements Engineering*, 16(4):267–280.

[Svensson et al., 2011] Svensson, R. B., Parker, P. L., and Regnell, B. (2011). A prototype tool for quper to support release planning of quality requirements. In *Proceedings of the 5th International Workshop on Software Product Management (IWSPM)*, pages 57–66, Trento, Italy. IEEE.

[Svensson et al., 2012] Svensson, R. B., Sprockel, Y., Regnell, B., and Brinkkemper, S. (2012). Setting quality targets for coming releases with quper: An industrial case study. *Requirements Engineering*, 17(4):283–298.

[Tawhid et al., 2012] Tawhid, R., Braun, E., Cartwright, N., Alhaj, M., Mussbacher, G., Shamsaei, A., Amyot, D., Behnam, S. A., and Richards, G. (2012). Towards outcome-based regulatory compliance in aviation security. In *Proceedings of the 20th IEEE International Requirements Engineering Conference*, pages 267–272, Chicago, Illinois. IEEE, IEEE.

[Teka et al., 2012] Teka, A., Condori-Fernandez, N., Kurtev, I., Quartel, D., and Engelsman, W. (2012). Change impact analysis of indirect goal relations: Comparison of nfr and tropos approaches based on industrial case study. In *Proceedings of the 2nd International Workshop on Model-Driven Requirements Engineering (MoDRE)*, pages 58–67, Chicago, Illinois. IEEE, IEEE.

[Teruel et al., 2011] Teruel, M. A., Navarro, E., Lopez-Jaquero, V., Montero, F., and Gonzales, P. (2011). Assesing the understandability of collaborative systems requirements notations: an empirical study. In *Proceedings of the 1st International Workshop on Empirical Requirements Engineering (EmpiRE)*, pages 85–92, Trento, Italy. IEEE.

[Thomas and Harden, 2008] Thomas, J. and Harden, A. (2008). Methods for the thematic synthesis of qualitative research in systematic reviews. *BMC Medical Research Methodology*, 8(45):45.

[Torres et al., 2012] Torres, R., Bencomo, N., and Astudillo, H. (2012). Mitigating the obsolescence of quality specifications models in service-based systems. In *Proceedings of the 2nd International Workshop on Model-Driven Requirements Engineering (MoDRE)*, pages 68–76, Chicago, Illinois. IEEE, IEEE.

[Uusitalo et al., 2011] Uusitalo, E., Raatikainen, M., Mannisto, T., and Tommila, T. (2011). Structured natural language requirements in nuclear energy domain: Towards improving regulatory guidelines. In *Proceedings of the 4th International Workshop on Requirements Engineering and Law (RELAW)*, pages 67–73, Trento, Italy. IEEE.

[van Tuijl et al., 2011] van Tuijl, G. J., Leenen, W., Shen, Z., van de Weerd, I., and Brinkkemper, S. (2011). Prioritizing requirements: An experiment to test the perceived reliability, usability and time consumption of bubblesort and the analytical hierarchy process. In Fricker, S. and Seyff, N., editors, *Proceedings of the 1st International Requirements Engineering Efficiency Workshop (REEW)*, Essen, Germany.

[Vandeville and Shaikh, 1999] Vandeville, J. V. and Shaikh, M. A. (1999). A structured approximate reasoning-based approach for gathering lessons learned information from system development projects. *Journal of Systems Engineering*, 2(4):242–247.

[Veerappa and Letier, 2011] Veerappa, V. and Letier, E. (2011). Clustering stakeholders for requirements decision making. In *Proceedings of the 17th International Working Conference on Requirements Engineering: Foundation for Software Quality*, LNCS, pages 202–208, Essen, Germany. Springer-Verlag.

[Vogl et al., 2011] Vogl, H., Lehner, K., Grunbacher, P., and Egyed, A. (2011). Reconciling requirements and architectures with the cbsp approach in an iphone app project. In *Proceedings of the 19th International Requirements Engineering Conference*, pages 273–278, Trento, Italy. IEEE, IEEE.

[Waldmann, 2011] Waldmann, B. (2011). There's never enough time: Doing requirements under resource constraints, and what requirements engineering can learn from agile development. In *Proceedings of the 19th International Requirements Engineering Conference*, pages 301–305, Trento, Italy. IEEE, IEEE.

[Wang et al., 2012] Wang, J., Li, J., Wang, Q., Zhang, H., and Wang, H. (2012). A simulation approach for impact analysis of requirement volatility considering dependency change. In Regnell, B. and Damian, D., editors, *Proceedings of the 18th International Working Conference on Requirements Engineering: Foundation for Software Quality*, LNCS, pages 59–76, Essen, Germany. Springer-Verlag.

[Wangenheim et al., 1998] Wangenheim, C. G. v., Ramos, A. M., Althoff, K.-D., Barcia, R. M., Weber, R., and Martins, A. (1998). Case-based reasoning approach to reuse of experiential knowledge in software measurement programs. In Gierl, L., editor, *Proceedings of the 6th German Workshop on Case-Based Reasoning*, Berlin, Germany.

[Weber and Aha, 2002] Weber, R. and Aha, D. (2002). Intelligent delivery of military lessons learnt. *Decision Support Systems*, 34(3):287–304.

[Weber et al., 2001] Weber, R., Aha, D., and Becerra-Fernandez, I. (2001). Intelligent lessons learned systems. *Expert Systems with Applications*, 17:17–34.

[Webster, 1913] Webster (1913). Webster dictionary. http://machaut.uchicago.edu/cgi-bin/WEBSTER.sh?WORD=lesson.

[Webster, 2013] Webster (2013). Webster's new world college dictionary. http://www.yourdictionary.com/lesson.

[Wellman, 2007] Wellman, J. (2007). Lessons learned about lessons learned. *Organization Development Journal*, 25(3):65–72.

[Wever and Maiden, 2011] Wever, A. and Maiden, N. (2011). What are the day-to-day factors that are preventing business analysts from effective business analysis? In *Proceedings of the 19th International Requirements Engineering Conference*, pages 293–298, Trento, Italy. IEEE, IEEE.

[Withall, 2007] Withall, S. (2007). *Software Requirements Patterns (Best Practices)*. Microsoft Press.

[Wnuk et al., 2012] Wnuk, K., Host, M., and Regnell, B. (2012). Replication of an experiment on linguistic tool support for consolidation of requirements from multiple sources. *Empirical Software Engineering*, 17(3):305–344.

[Wu et al., 2012] Wu, Y., Zowghi, D., Peng, X., and Zhao, W. (2012). Towards understanding requirement evolution in a software product line: An industrial case study. In *Proceedings of the 1st International Workshop on the Twin Peaks of Requirements and Architecture (TwinPeaks)*, pages 7–14, Chicago, Illinois. IEEE, IEEE.

[Yang et al., 2012] Yang, H., De Roeck, A., Gervasi, V., Willis, A., and Nuseibeh, B. (2012). Speculative requirements: Automatic detection of uncertainty in natural language requirements. In *Proceedings of the 20th IEEE International Requirements Engineering Conference*, pages 11–20, Chicago, Illinois. IEEE, IEEE.

[Yi et al., 2012] Yi, L., Zhang, W., Zhao, H., Jin, Z., and Mei, H. (2012). Mining binary constraints in the construction of feature models. In *Proceedings of the 20th IEEE International Requirements Engineering Conference*, pages 141–150, Chicago, Illinois. IEEE, IEEE.

[Yin, 2003] Yin, R. K. (2003). *Case Study Research: Design and Methods*. Sage Publications, 3rd edition.

[Yue et al., 2011] Yue, T., Briand, L. C., and Labiche, Y. (2011). A systematic review of transformation approaches between user requirements and analysis models. *Requirements Engineering*, 16(2):75–99.

[Zhu and Herrmann, 2012] Zhu, L. and Herrmann, T. (2012). Design now! – elaborating requirements in situated action. In Daneva, M., Doerr, J., Herrmann, A., and Schneider, K., editors, *Proceedings of the 2nd Workshop on Creativity in Requirements Engineering*, LNCS, pages 93–104, Essen, Germany. Springer-Verlag.

Curriculum Vitae

Name:	Ibtehal Noorwali
Post-Secondary Education and Degrees:	Umm Al-Qura University Makkah, Saudi Arabia 2005-2009 B.Sc., Computer Science
	University of Western Ontario London, Ontario, Canada 2011-2013 M.Sc., Computer Science
Honours and Awards:	Saudi Arabia's Ministry of Higher Education Scholarship for Graduate Studies 2011
Related Work Experience:	Teaching Assistant Umm Al-Qura University 2009-2011

Publications:

Noorwali, I.N., Madhavji, N.H., Maps of Lessons Learnt in Requirements Engineering: A Research Preview, accepted in 19th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 13) in January, 2013.