PRISCILLA MAFRA DE CARVALHO MARQUES

GUIDELINES FOR PREVENTING REQUIREMENTS ENGINEERING PROBLEMS

Dissertation presented to the Computing Graduate program of the Universidade Federal Fluminense in partial fulfilment of the requirements for the degree of Master of Science. Research area: Systems and Information Engineering.

Advisor: Prof. D.Sc. MARCOS KALINOWSKI

Niterói

2017

Ficha Catalográfica elaborada pela Biblioteca da Escola de Engenharia e Instituto de Computação da UFF

M357 Marques, Priscilla Mafra de Carvalho
Guidelines for preventing requirements engineering problems / Priscilla Mafra de Carvalho Marques. – Niterói, RJ : [s.n.], 2017.
149 f.

Dissertação (Mestrado em Computação) - Universidade Federal Fluminense, 2017.
Orientador: Marcos Kalinowski.

1. Engenharia de software. 2. Engenharia de requisitos. 3. Desenvolvimento de software. I. Título.

CDD 005.1

PRISCILLA MAFRA DE CARVALHO MARQUES

GUIDELINES FOR PREVENTING REQUIREMENTS ENGINEERING PROBLEMS

Dissertation presented to the Computing Graduate program of the Universidade Federal Fluminense in partial fulfilment of the requirements for the degree of Master of Science. Research area: Systems and Information Engineering.

Aprovada em Julho de 2017.

Niterói

2017

This work is dedicated to God and my family.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank God, because of His marvellous grace in granting me the opportunity to try getting my master degree, without Him anything that I accomplished until now would not be possible.

I would like to thank my family for all the support, love and incentive.

I would like to thank my advisor Marcos Kalinowski for all the help, patience, wisdom and guidance provided during this time, I could not ask for a better mastermind.

I would like to thank Daniel Méndez Fernández for all the support during the development of my work. I hope we can work together in other research opportunities in the future.

I would like to thank the assessment board members professors José Viterbo and Alessandro Garcia for accepting to review this dissertation.

I would like to thank my bosses Newton Saldanha and Gelson Nascimento for allowing me following my dream and also for all the cheers, without them this would not be possible.

I would like to thank the NaPiRE team especially the ones that helped me answering the survey to validate my work.

Finally, I would like to thank my friends Jorge Thiago, Marcus Souza and Vitor Neves for all their help and valuable advices.

" There is no sense in being precise about something when you do not even know what you are talking about." (Jon von Neumann)

RESUMO

[Contexto] Problemas na Engenharia de Requisitos (ER) podem levar a sérias consequências durante o ciclo de vida de desenvolvimento de um software. [Objetivo] O objetivo dessa dissertação é propor diretrizes que poderão ser utilizadas por diferentes tipos de empresas, de acordo com seu modelo de processo (ágeis e direcionadas a planos), para prevenir problemas da ER. [Método] Para atingir tal objetivo, informações coletadas de um projeto focado em problemas da ER foram utilizadas. Dados referentes à 228 empresas de 10 países diferentes foram analisados para compor as diretrizes. Em seguida, as diretrizes foram avaliadas e refinadas com base no *feedback* de especialistas da área. [Resultados] A partir dos dados coletados, os problemas considerados pelos participantes como mais críticos da ER, suas causas e possíveis ações de mitigação foram identificadas e organizadas em grupos de acordo com o modelo de processo. Finalmente, as causas e ações de mitigação dos problemas considerados mais críticos de cada grupo de modelo de processo foram analisadas para embasar diretrizes que possam ser utilizadas para prevenir tais problemas. A avaliação pelos especialistas permitiu o refinamento das diretrizes. [Conclusões] Esta dissertação disponibiliza as diretrizes resultantes, que podem ser utilizadas, de acordo com o contexto característico de cada empresa, como ponto de partida para apoiar a prevenção de problemas críticos da ER.

Palavras-chave: diretrizes; prevenção de problemas; prevenção de defeitos; engenharia de requisitos

ABSTRACT

[Context] Problems in Requirements Engineering (RE) can lead to serious consequences during the software development lifecycle. [Goal] The goal of this dissertation is to propose empirically-based guidelines that can be used by different types of organisations, according to their process model (agile or plan-driven), to help them preventing RE problems. [Method] To achieve this goal, information collected in a survey on RE problems was used. Data from 228 organisations of 10 different countries was analysed to propose the guidelines. Thereafter, the guidelines were evaluated and refined based on feedback from experts in the field. [Results] From the collected survey data, the RE problems considered by the respondents as the most critical ones, their causes and mitigation actions were identified and organised by clusters of process model. Finally, the causes and mitigation actions of the critical problems of each cluster were analysed to get further insights into guidelines for potentially preventing them. The feedback from the experts allowed refining the guidelines. [Conclusions] This dissertation provides the resulting guidelines, which can be used, according to the characteristic context of the companies, as a starting point to support preventing critical RE problems.

Keywords: guidelines; problem prevention; defect prevention; requirements engineering

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

CHAPTER 1 – INTRODUCTION

1.1 CONTEXT

Requirements Engineering aims at the elicitation, analysis, and specification of requirements that unambiguously reflect the intended purpose of a software system considering and aligning the viewpoints of all relevant stakeholders (Méndez Fernández *et al.* 2016). Precise and consistent requirements directly contribute to appropriateness and cost-effectiveness in the development of a system (Nuseibeh and Easterbrook 2000) whereby RE is a determinant of productivity and (product) quality (Damian and Chisan 2006). However, RE is still characterised by its uncertainty, as many aspects are not clear from the beginning of a project. Hence, it is highly volatile and inherently complex by nature (Méndez Fernández and Wagner 2015).

Brooks (1987) emphasizes this complexity: "The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all the interfaces to people, to machines, and to other software systems. No other part of the work cripples the resulting system if done wrong. No other part is more difficult to rectify later".

Therefore, specialists consider RE as one of the hardest parts of building a software system. Defining precisely what to build when you deal with a large number of variables endorses such difficulty. One important variable that contribute to the complexity of building a software system is the fact that you deal with people all the time. Faults committed by one of such stakeholders can lead to a delivery delay or even to project failure.

Given this complexity and the need to interact with different types of people, organisations may face several problems during the RE process. Many of those problems can be critical, leading to severe implications, including project failure (Brooks 1987) (Méndez Fernández *et al.* 2016). The communication within the project team is an example of a problem itself, in some projects, the team chosen may never have worked together before. Each member of the team has his/her own level of experience, knowledge and concepts, what may hinder the relationship between the project team during the process of building a software system.

Besides the communication within the project team, the communication between the project team and the customer is another even more critical problem to be treated. The fact that the customers may have difficulties specifying what they want and, in many cases, the their limited availability to solve eventual doubts related to the project, increases the difficulty of the requirements elicitation. Situations like these may lead the requirements to be incomplete, inconsistent, hidden and to change constantly, propagating, thus, a RE problem to subsequent parts of the project.

Furthermore, the cost for correcting the RE-related problems increases throughout the software development life cycle (Boehm and Basili 2001), the longer the project team takes to find out what the problem is and when it occurred, the greater will be the cost to solve it. Besides, industry is still struggling in defining and applying high-quality RE and researchers are trying to understand industrial expectations and problems (Méndez Fernández *et al.* 2012). The actual state of empirical evidence in RE is particularly weak and dominated by, if at all, isolated case studies and small-scale studies investigating aspects that hardly can be generalised (Méndez Fernández *et al.* 2016).

1.2 MOTIVATION

In 1994, the Standish Group surveyed over 350 companies about their 8000 software projects to find out how well they were faring (Standish 1994). Thirty-one percent of the software projects were cancelled before they were completed. Moreover, in large companies, only 9 percent of the projects were delivered on time and cost what they were budgeted, and 16 percent met those criteria in small companies. In the Standish Group survey it was possible to observe that some parts of the requirements elicitation, definition and management process are involved in many of the causes leading to project failure. Indeed, incomplete requirements and lack of users involvement were the top factors of causes for project failure cited by the survey respondents (Pfleeger 2001). Naeem *et al.* (2016), on their work claims that the major cause of application failures is poorly defined requirements. According to them, ambiguous requirements lead to confusion, wasted effort and rework.

Despite all the RE-related problems and its severe consequences on the building of a software system, there are no specific guidelines yet, at our disposal, that could be used by different types of organisations to help them on the prevention of the critical RE problems (Mafra *et al.* 2016). Furthermore, the rework, one of the consequences of a RE problem, can consume from 30 to 50 percent of your total development cost (Boehm and Papaccio 1988), and requirements errors account for 70 to 85 percent of the rework cost (Leffingwell 1997) (Wiegers 2003).

Figure 1.1, taken from Wiegers (2003), shows that it costs far more to correct a defect that is found late in the project than to fix it shortly after its creation (Grady 1999).

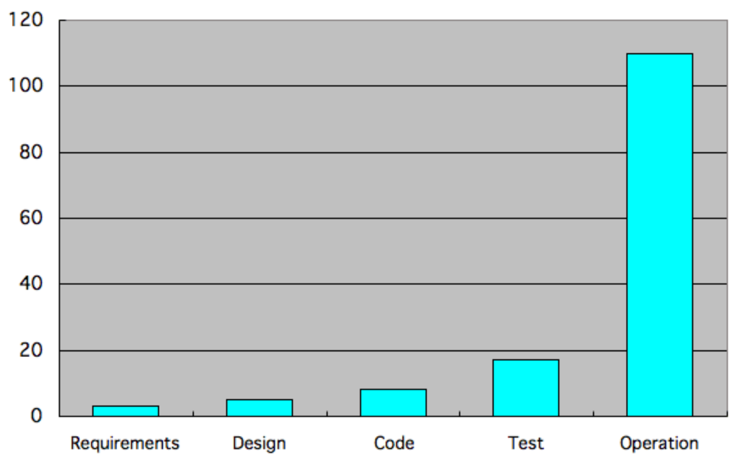


Figure 1.1. Relative cost to correct a requirement defect depending on when it is discovered. Taken from Wiegers (2003).

Boehm and Papaccio (1988) report that for every $1 spent to find and fix a requirements-based problem during the requirements definition process, it could cost up to $5 to repair it during design, $10 during coding, $20 during unit testing, and as much as $200 after delivery of the system (Pfleeger 2001). This scenario reinforces the importance of introducing means to preventing problems right from the beginning of the project.

1.3 GOALS / PROBLEM DEFINITION

In order to better understand the status quo on RE practice and its critical problems, a project named NaPiRE (*Naming the Pain in Requirements Engineering*) was launched in 2012 involving researchers from different countries (Méndez Fernández and Wagner 2015). The NaPiRE project comprises the design of a family of surveys on RE and its goal is to lay an empirical foundation about practical problems and needs of RE to allow directing future research in a problem-driven manner (Méndez Fernández and Wagner 2015). The last survey round (2014-2015) was answered by in total 228 organisations from 10 different countries around the globe, including organisations following different process models (Méndez Fernández *et al.* 2016).

The main goal of this dissertation is to use the data obtained from the NaPiRE initiative to propose empirically-based guidelines that can be used by different types of organisations to support them in the prevention of the critical RE problems.

1.4 DISSERTATION ORGANISATION

This chapter presented the context, the motivation and the goal of this research. The next chapters of this dissertation are organised as follows:

- **Chapter 2 – Theoretical Foundation.** This chapter discusses the basic concepts of the RE and presents related work on RE problems.
- **Chapter 3 – Preliminary Guidelines For Preventing Critical RE Problems.** This chapter presents the preliminary guidelines for preventing the most critical RE problems along with the involved process to produce them. It shows how the raw data from the NaPiRE survey was manipulated to facilitate the analysis and to contribute to the conception of the guidelines.
- **Chapter 4 – Evaluation and Adjusted Guidelines.** This chapter presents the evaluation conducted on the preliminary guidelines with experts in the field and exposes its results. The chapter also presents the updated guidelines generated through the analysis of the evaluation answers.
- **Chapter 5 – Concluding remarks.** This chapter describes the contributions of this research, its limitations and future work.

This dissertation also contains seven appendixes and one annex:

- **APPENDIX A - Rank of The Typical RE Problems:** shows the rank of the criticality for each of the 21 pre-compiled problems practitioners are meant to typically encounter in practice.
- **APPENDIX B - Codes for Causes and Mitigation Actions for the Most Critical RE Problems:** shows the codes made for the raw data of the causes and mitigation actions of all the most critical RE problems.
- **APPENDIX C - Adapted Ishikawa Diagrams for the most Critical RE Problems:** shows the adapted Ishikawa Diagrams for each cluster of the most critical RE problems for agile and plan-driven organisations.
- **APPENDIX D - Preliminary Guidelines for Preventing Critical RE problems on Agile and Plan-Driven Organisations:** shows the preliminary guidelines for each cluster of the top 6 most critical RE problems for agile and plan-driven organisations.
- **APPENDIX E - Evaluation of the Preliminary Guidelines – Survey:** shows the survey used to evaluate the preliminary guidelines for each cluster of the two most critical RE problems (Communication flaws between the project

team and the customer and Incomplete and/or hidden requirements) for agile and plan-driven organisations.

- **APPENDIX F - Results of the Survey on the Guidelines for Preventing the two most Critical RE Problems:** shows the raw expert survey result for the two most critical RE problems.
- **APPENDIX G - Analysis of the Answers of the Survey data on the Guidelines for preventing the two most Critical RE problems:** shows the analysis of the survey data for the two most critical RE problems.
- **ANNEX A - NaPire Survey:** shows the NaPiRE questionnaire. All the research conducted in this dissertation was based on the answers of this survey, which was run in 2014-2015.

CHAPTER 2 – THEORETICAL FOUNDATION

2.1 INTRODUCTION

This chapter will present a theoretical foundation of requirements engineering. Its goal is to describe the overall RE process elements to contribute with a better understanding of the dissertation context. The next subsections provide information about the basic concepts of the requirements engineering, discussing how requirements are managed and developed and presenting characteristics of the requirements engineering practices in agile and plan-driven methods. Thereafter, a discussion about related work on RE problems, along with an overview on the context of the NaPiRE project is presented.

2.2 BASIC CONCEPTS OF REQUIREMENTS ENGINEERING

Requirements engineering is the process where the expectations of the customer for a new system or a modified one are raised and described. Nuseibeh and Easterbrook (2000) defined RE as the process of discovering the purpose of which the software system was intended, by identifying stakeholders and their needs, and documenting these in a form that is amenable to analysis, communication, and subsequent implementation.

Requirements are specifications of what should be implemented by the project team during the development of a software system. They are descriptions of how the system should behave, or of a system property or attribute. They may be a constraint on the development process of the system (Sommerville and Sawyer 1997).

Requirements can be classified as functional and non-functional ones. The functional requirements describe how the system will behave in certain interactions, its functions, inputs, outputs and exceptions, for example. For Wiegers (2003), functional requirements describe the software functionality, what the developer need to implement thereby satisfying the business requirements. An example of functional requirement taken from Pfleeger (2001) says, for a system printing weekly paychecks, the functional requirements must answer questions about when paychecks are issued. What input is necessary for a paycheck to be printed? Under what conditions can the amount of pay be changed? What causes the removal of an employee from the payroll list? Therefore, those requirements describe what the system will do.

The non-functional requirements are not specified as a functionality of the system, although, they are characteristics that ensure the software quality. They are functions and services restrictions of the software system and are applied to the overall system. The project team decides whether to attend the non-functional requirements or not, if they do not attend those requirements, the software system can be ineffective.

The non-functional requirements emerge from users necessity, due to budget restrictions, organisational politics, need of interoperability with other software or hardware systems, or from external factors, as safety regulations or privacy legislations (Sommerville 2011). Figure 2.1, taken from Kotonya and Sommerville (1998) presents a classifications of the non-functional requirements.

This figure shows that non-functional requirements, as said before, can be derived from software characteristics, from the software development organisation or from external fonts. The product requirements specify or restrict software behaviour, Sommerville's point of view is that the product requirements define when the system has to be available and the daily time allowed for its unavailability (Sommerville 2011). The organisational requirements are related to the politics and procedure of the organisation or of the customer, they specify how the users authenticate on the system. The external requirements cover all requirements derived from external fonts that are not related to the development process or the system, they derive from the system necessity of being in accordance with the privacy legislation.

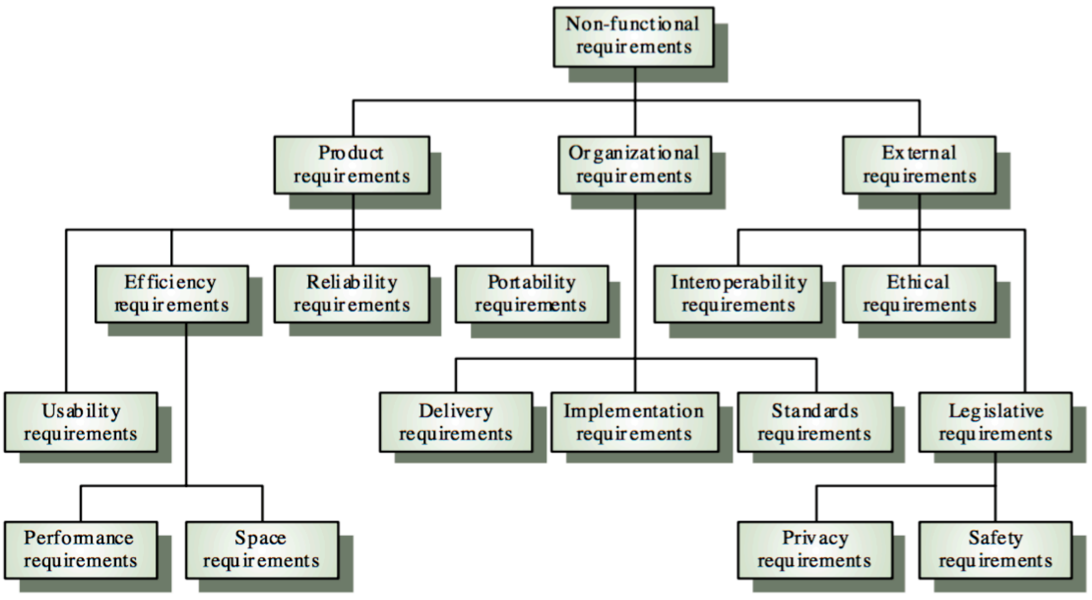


Figure 2.1. Classification of non-functional requirements. Taken from Kotonya and Sommerville (1998).

The paragraphs before presented the characteristics of the functional and the non-functional requirements. In order to summarise the main characteristics of each type of requirements facilitating the comprehension of the difference between them, the

Table 1, adapted from Osta (2013), is presented next.

Table 1. Differences between Functional and Non-Functional Requirements. Adapted from Osta (2013).

Functional Requirements	Non-Functional Requirements
Defines all the services or functions required by the customer that must be provided by the system.	Defines system properties and constraints, e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.
Describes what the software should do.	Does not describe what the software will do, but how the software will do it.
Related to business. For example: calculation of order value by Sales Department or gross pay by the Payroll Department.	Related to improving the performance of the business. For example: checking the level of security. An operator should be allowed to view only my name and personal identification code.
Easy to test.	Difficult to test.
Related to individual system features.	Related to the system as a whole.
Failure to meet the individual functional requirement may degrade the system.	Failure to meet a non-functional requirement may make the whole system unusable.

This section described basic concepts of the requirements (functional and non-functional) and of requirements engineering. The next step is to explain the mechanics of a typical requirements engineering process. This will be explained with more details in the next subsection.

2.2.1 REQUIREMENTS ENGINEERING PROCESS

The Capability Maturity Model Integration (CMMI) contains two process areas related to requirements engineering: requirements management and requirements development, as can be seen in Figure 2.2, adapted from Wiegers (2010).

CMMI can be seen as a collection of best practices to analyse the process maturity of an organisation. Based on CMMI's RE process area configuration, the subcomponents of RE will be explained with more details on the next subsections.

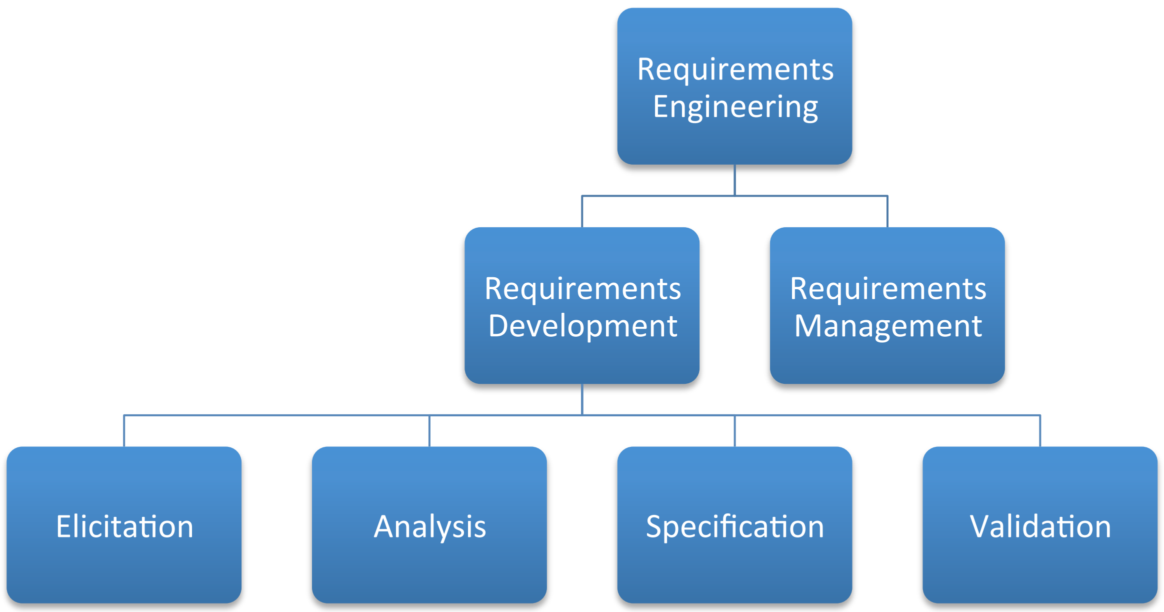


Figure 2.2. Subcomponents of the requirements engineering domain, adapted from Wiegers (2010).

2.2.1.1 REQUIREMENTS MANAGEMENT

Requirements management has a corresponding process area belonging to the maturity level 2 of the CMMI. It concerns managing all requirements received or generated by the project, including both technical and nontechnical requirements as well as requirements levied on the project by the organisation (CMMI PRODUCTION TEAM 2010). The purpose of the Requirements Management process area is to manage requirements of the project's products and product components and to ensure alignment between those requirements and the project's plans and work products (CMMI PRODUCTION TEAM 2010). Requirements management entails establishing and maintaining an agreement with the customer on the requirements for the software project (Paulk *et al.* 1995).

CMMI contains five specific goals related to requirements management (CMMI PRODUCTION TEAM 2010). One of these goals is to understand the requirements. This goal includes the analysis of the requirements to decide on including them into the project. To include requirements into a product it is not only necessary to obtain the customer approval, but of the project team as well. The project team, for instance, will have to evaluate the requirements specification to analyse the feasibility and prevent misunderstandings.

Another CMMI specific goal is to obtain commitment to requirements, through the analysis of the impact of requirements-change on existing commitments or at the start of a new requirement. The importance of this type of commitment is also highlighted by Wiegers

(2010). The changes to existing commitments should be negotiated before project participants commit to a new requirement or requirement change.

It is necessary to manage requirements changes in order to analyse their impact on the project and measure its volatility, for example. The requirements volatility might suggest that the problem is not well understood, the project scope is not well defined, the business is changing rapidly, many requirements were missed during elicitation, or politics are running rampant (Wiegers 2010). To effectively manage their change, it is important to document the requirements and their changes, maintaining a change history.

Maintaining the bidirectional traceability of requirements is another goal of the CMMI requirements management process area. When requirements are managed well, traceability can be established from a source requirement to its lower level requirements and from those lower level requirements back to their source requirements. Such bidirectional traceability helps to determine whether all source requirements have been completely addressed and whether all lower level requirements can be traced to a valid source (CMMI PRODUCTION TEAM 2010). Another important aspect of maintaining the bidirectional traceability of requirements is linking each functional requirement to the design and code elements that implement it and the tests that verify it. This traceability information can also connect functional requirements to the higher-level requirements from which they were derived and to other related requirements (Wiegers 2010).

Finally, it is necessary to ensure the alignment between the project work and the requirements, reviewing project plans, activities, and work products for consistency with requirements and changes made to them (CMMI PRODUCTION TEAM 2010).

2.2.1.2 REQUIREMENTS DEVELOPMENT

Regarding requirements development, CMMI contains a related process area that belongs to maturity level 3 and has the following purpose: “elicit, analyse and establish customer, product and product component requirements” (CMMI PRODUCTION TEAM 2010). As previously shown in Figure 2.2, requirements development is commonly subdivided into elicitation, analysis, specification and validation activities (Abran and Moore 2001). Figure 2.3, taken from Pfleeger (2001), illustrates the process of determining the requirements of a system.

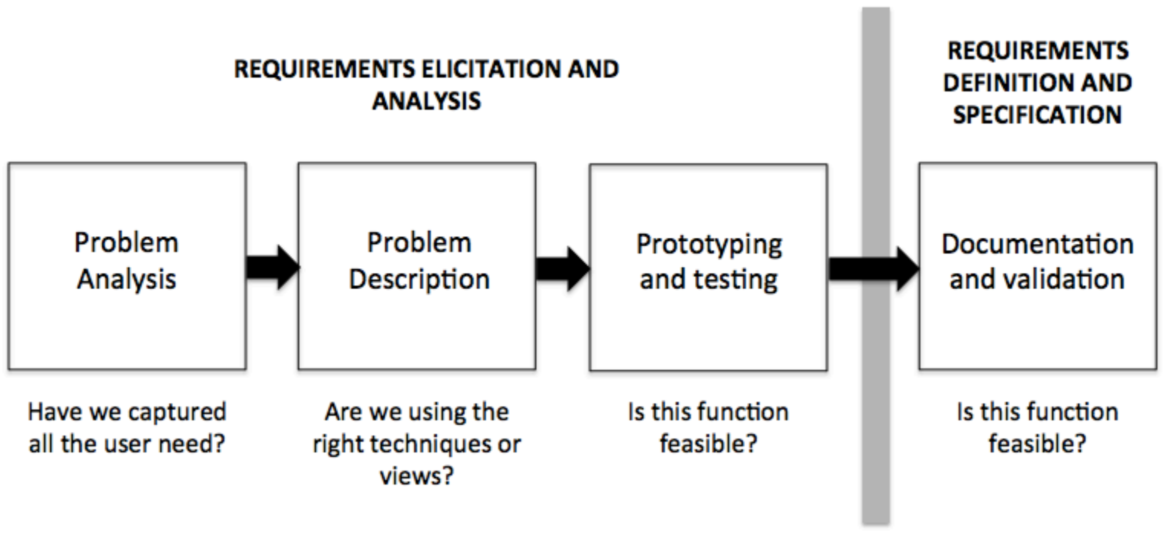

REQUIREMENTS ELICITATION AND ANALYSIS
REQUIREMENTS DEFINITION AND SPECIFICATION
Problem Analysis
Problem Description
Prototyping and testing
Documentation and validation
Have we captured all the user need?
Are we using the right techniques or views?
Is this function feasible?
Is this function feasible?

Figure 2.3. The process of determining requirements. Taken from Pfleeger (2001).

The first activity of the requirements development is to elicit the requirements, to do so it is necessary to work with the customers asking questions that will help to capture what he/she imagined for the system to be. Requirements elicitation enables to write a requirements definition document; written in terms that the customer can understand, the requirements definition is a complete listing of everything the customer expects the proposed system to do (Pfleeger 2001).

Brooks (1987) pointed that it is very difficult for the customers, even those working directly with software engineers, to specify completely, precisely, and correctly the exact requirements of a modern software product before having built and tried some versions of the product they are specifying. Therefore, creating prototypes during the requirements elicitation to help the project team and the customers clarify eventual doubts, analyse a requirement feasibility or validate customers decisions can be a good strategy.

The next requirements development activity is the analysis. The analysis concerns determining whether the requirements raised on the elicitation are clear, complete, consistent and unambiguous, resolving any conflicts, if needed. On the analysis, as the name says, the information received from users, on the elicitation phase, is analysed to distinguish their task goals from functional requirements, non-functional requirements, business rules, suggested solutions, and extraneous information (Wiegers 2010).

After the analysis, the raised requirements are documented in a persistent and well-organised way to facilitate the communication and the change management. During requirements specification, the requirements are also confirmed and the project scope established together with the customers. The requirements specification restates de requirements definition in technical terms appropriate for the development of a system design (Pfleeger 2001). Use cases, user stories, functional requirements and visual analysis models are commonly used during requirements specification (Nicolás and Toval 2009).

Finally, validation and verification activities should be performed, to assure the quality of the produced finding problems with the requirements. Requirements validation aims at assuring that the specification is consistent with the requirements definition; that is, validation makes sure that the requirements will meet the customer's needs (Pfleeger 2001). On the other hand, the requirements verification consists of confirming that the designed and built product fully addresses the documented requirements. On the verification process, inspections, tests and analyses are performed throughout the product lifecycle to ensure that the design, iterations, and the finished product fully address the requirements.

While these process elements and activities help to provide an overview on requirements engineering, it is noteworthy that organisations may apply them differently. Indeed, the software process (and RE practices) may vary significantly according to the process model of the organisation.

Sommerville (2011) states that the Software Process Model is a simplified representation of a software process. Agile and Plan-Driven are two types of software process models that rely on different philosophies. The Plan-Driven methods are known by their formal approach, those methods are characterised by having detailed plans, process control, quality and performance metrics, extensive documentation, predictability, stability, risk management, verification and validation. Furthermore, organisations that adopt the Plan-Driven method have a very formal and contractual relationship with the customer. On the other hand, the Agile methods propose functional software and less documentation, they try to avoid the overhead of planning and documentation of the Plan-Driven methods.

The next subsections provide more details on Plan-Driven and Agile methods and on how the essentials of requirement engineering works in each of those process models.

2.2.2 PLAN-DRIVEN METHODS

Plan-Driven methods are considered a traditional approach for software development. They are characterised by having well-defined development phases offering opportunities for continuous improvement (Boehm and Turner 2004).

According to Svensson (2005), there is a planning aspect inherent in the Plan-Driven methods. Thus, these methods contain detailed descriptions of activities, workflows and roles and so forth in order to facilitate the planning activity. The development activity consists of following a series of carefully specified phases, where the output from one phase is the input to the next phase. Further, there is a focus on thorough documentation in each phase. However, an advantage with plan-driven methods is the repeatability of the process due to the level of detail of its contents.

Boehm and Turner (2004) accounted that Plan-Driven methods work best when the requirements are largely determinable in advance and remain relatively stable. Change rates on order of 1 percent of the requirements per month are acceptable. Unfortunately, it is very difficult to encounter a project this stable in practice, requirements change constantly, increasing the challenges of controlling the changes with Plan-Driven methods. However, the Plan-Driven methods characteristic of working best with stable projects and use extensive documentation leads these methods to have large delays through rework of elaborate plans and specification as one of its risks.

Another characteristic of the Plan-Driven methods, also according to Boehm and Turner (2004), is the fact that the communication tends to be one way. The communication is generally from one entity to another rather than between two entities.

There are several types of process models for Plan-Driven methods: Waterfall, V-Model and RUP are some of the most largely known. The Waterfall model is considered a classical process model. In Figure 2.4, taken from Sommerville (2006), it is possible to observe the fundamental activities of the Waterfall process model organised by its running sequence. Waterfall is a sequential model where each phase of its development must be fully completed before the next phase begins, the phases do not overlap each other. At the end of each phase a review occurs in order to determine if the project is on the right path and whether or not to continue or discard the project. The Waterfall process model is typically used when the requirements are very clear and fixed, the definition of the product is stable, the technology is understood, there are no ambiguous requirements and when working with small projects.

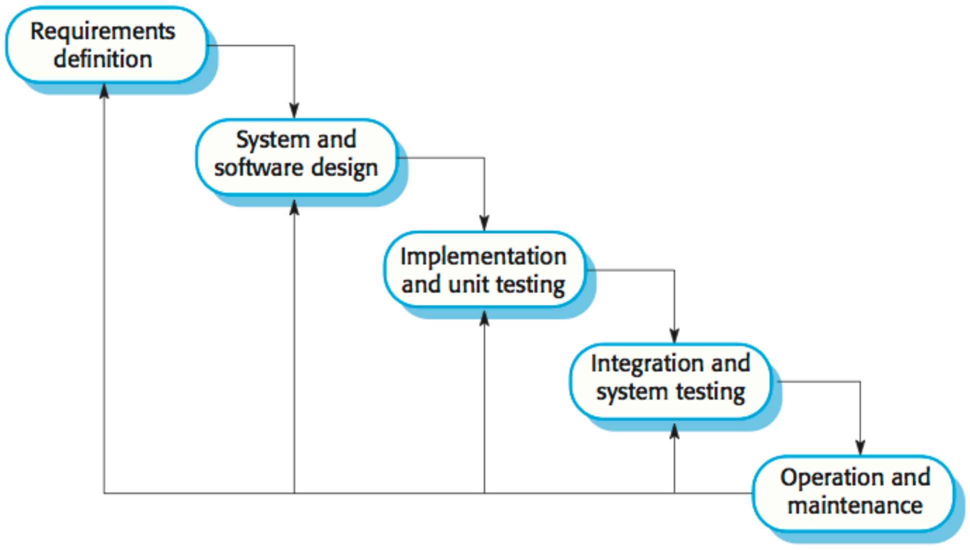

Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

Figure 2.4. Waterfall Model, taken from Sommerville (2006).

Another well-known process model is the V-Model (Figure 2.5). As the Waterfall, this model also follows sequential phases that should occur one at a time, until the project is complete. During each phase, the corresponding tests are designed in parallel to be implemented later during the testing phases. The V-Model should be used for small and medium sized projects where requirements are clear and fixed.

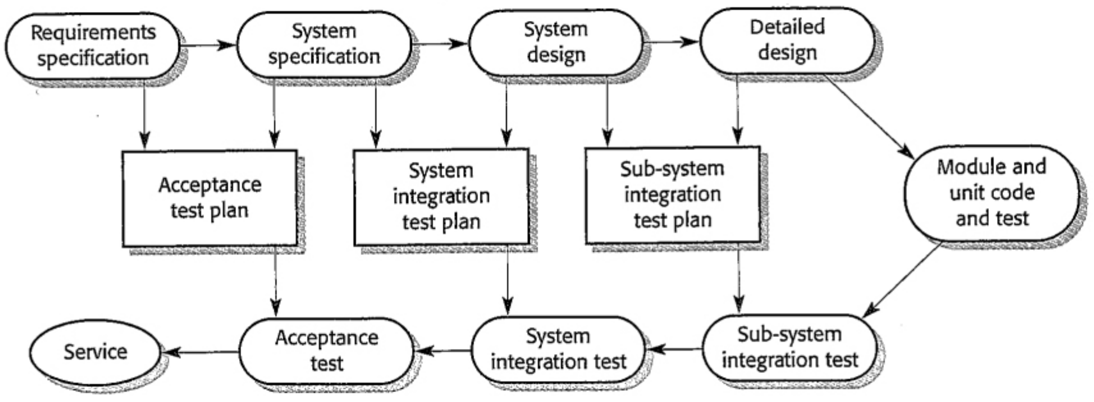

Requirements specification
System specification
System design
Detailed design
Acceptance test plan
System integration test plan
Sub-system integration test plan
Module and unit code and test
Service
Acceptance test
System integration test
Sub-system integration test

Figure 2.5. V-Model, taken from Sommerville (2006).

The last process model quoted before was the Rational Unified Process (RUP). RUP is an iterative software development process model that supports prototyping and incremental deliveries.

According to Sommerville (2011), RUP's phases are: inception, elaboration, construction and transition. Each phase can be executed iteratively and all the set of phases can also be executed incrementally as shown by the arrows in Figure 2.6.

In the inception phase the business case is settled. All entities that will interact with the system are identified and the system contribution to the business is analysed in order to cancel it on the next phase if the contribution is small.

The elaboration phase is when the system requirements model is created, in order to reach this goal, it is important to develop a comprehension of the predominant problem, establish an architecture framework for the system, develop the project plan and identify the biggest risks of the project.

In the construction phase the coding and testing of the project is developed. At the end of this phase the system will be functional and the associated documentation will be ready to be delivered to the customer.

Finally, on the transition phase, the software is delivered to the customer implying that it will be functional on the operational environment and the all the documentation concluded (Sommerville 2011).

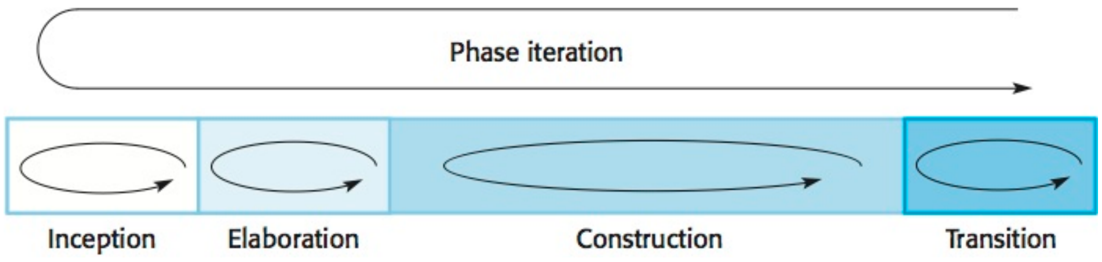


Figure 2.6. RUP model, taken from Sommerville (2006).

Those process models, among others not quoted here, are example process models of Plan-Driven methods and their requirements engineering process differs from the process models of the Agile methods. The next subsection will explain how requirement engineering is typically conducted in the context of Plan-Driven methods.

2.2.2.1 REQUIREMENTS ENGINEERING IN PLAN-DRIVEN METHODS

The Plan-Driven methods, because of their described characteristic and formality, with well-defined processes, basically follow the previously presented requirements engineering processes elements (requirements management and development) and activities.

Organisations that adopt Plan-Driven methods commonly base their processes on practices that are typically also recommended in the context of maturity models, such as following defined processes, having corrective actions managed until their conclusion, and project progress monitored against the project plan.

There are currently more than 27 maturity models like the Project Management Maturity Model (PMMM), the Organisational Project Management Maturity Model (OPM3), the Kerzner Project Management Maturity Model (KPMMM) and the *Modelo de Maturidade em Gerenciamento de Projetos* (MMGP), for instance, but the best-known maturity model related to software construction is the Capability Maturity Model Integration (CMMI) for development.

In such contexts, requirements management and development are typically performed by conducting the activities described in Subsections 2.2.1.1 and 2.2.1.2 following well-defined processes. Moreover, the documentation of requirements is usually done more rigorously, given that there is an inherent need for a detailed specification in order to allow preparing precise estimates and detailed plans in advance.

2.2.3 AGILE METHODS

While the Plan-Driven methods follow well-defined processes and are adherent to the requirements engineering practices presented in the lasts sections, Agile methods propose functional software and less documentation, they try to avoid the overhead on the planning, project and documentations of the Plan-Driven methods.

Agile Methods were proposed in the 1990's with the objective of focusing on the software and not on the documentation. Those methods are intended to quickly deliver the customer a running software and, he/she will be able to propose changes and new requirements inclusions for the systems new interactions (Sommerville 2011). Projects that fail tend to have a great lack of early customer feedback (Eberlein and Leite 2002).

According to the Agile Manifesto (Beck *et al.* 2001), the Agile Methods value individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation and responding to change over following a plan. The focus of the Agile methods rely more on the people than on the process, making it possible to adapt new factors and give faster answers to changes (Prikladnicki *et al.* 2014). Those methods work with small iterations of two or three weeks, called sprints. At the end of each sprint the outcome is presented to the customers that provide their feedback about the requirements evolution.

Currently there are lots of Agile Methods. Extreme programming (XP) and SCRUM are some examples of them. Extreme programming is probably the most well-known and used method of the agile methods recently (Cohen *et al.* 2003). Also, according to Sommerville (2011), in the XP model, the approach of iterative development was used in "extreme" level. For example, many versions of a system can be developed, integrated and tested in one day by different developers.

In XP, the requirements are described as stories and, from those stories, the tasks are directly generated and implemented. This model uses pair programming and the developers create the tests for each task before the implementation. When new code is added to the system, all the created tests have to be successfully executed (Sommerville 2011). Figure 2.7 shows the release cycle of the XP model, the interval between the system releases is small.

There are some characteristics of the XP model that reflect agile principles, they are: XP model uses small and constant releases and, each release includes one functionality presented in the system story; customer is constantly involved with the project, his/her representative participates on the development of the system and is responsible to define the system acceptation tests; use pair programming; sustainable development, which does not involves excessively long working hours; changes are accepted in continuous releases to the customers; use constant refactorings in order to improve code quality (Sommerville 2011).

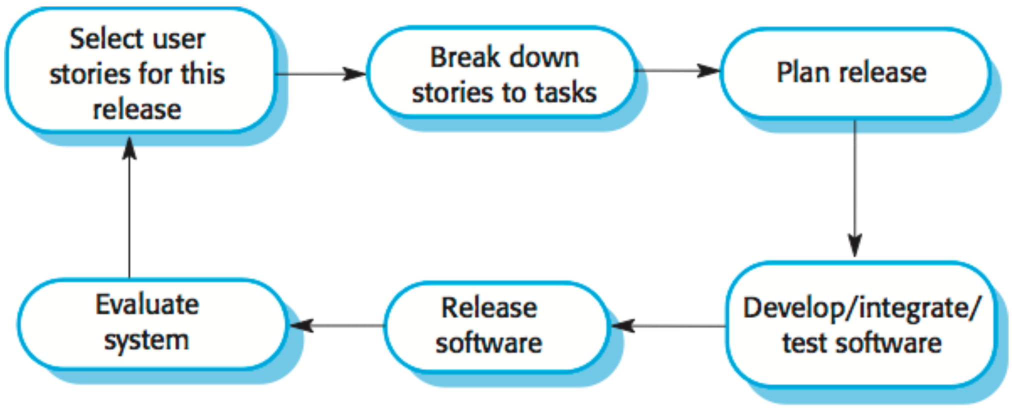

Select user stories for this release
Break down stories to tasks
Plan release
Develop/integrate/ test software
Release software
Evaluate system

Figure 2.7. Release cicle of XP method, taken from Sommerville (2006).

The other example of Agile model quoted before was the SCRUM. According to the International Scrum Institute (2017), SCRUM is a lightweight agile process management framework mainly used in software development and describes an iterative and incremental approach for project work. Since SCRUM does not have agile processes common technical approach (does not suggest the use of pair programming and test-driven development, for instance), it can be used together with other Agile methods like XP, for example, in order to provide a project management framework (Sommerville 2011).

On the Figure 2.8, taken from International Scrum Institute (2017), it is possible to observe the SCRUM management process diagram. It has three phases, the first one concerns the planning, where the projects goals and software architecture are established. On the second phase of SCRUM occurs the sprint cycles, on each sprint a system increment is made.

Finally, the last phase, is the project closure, all the required documentation is completed and the lessons learned on the project are evaluated.

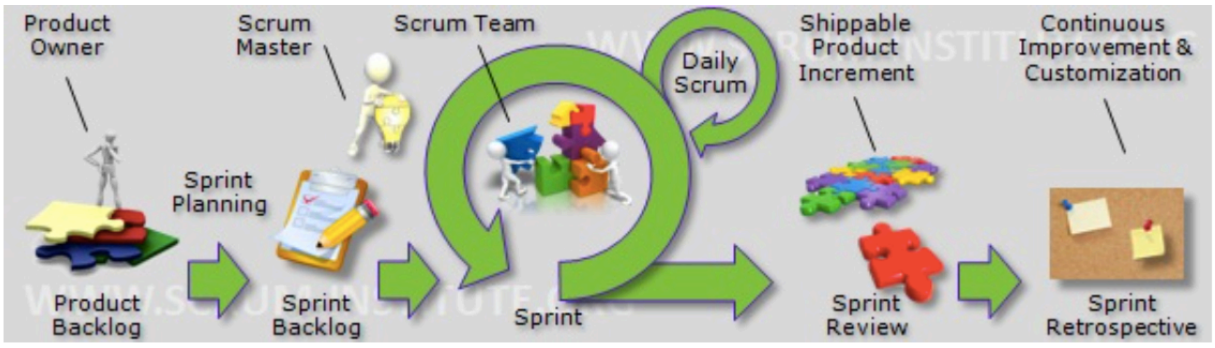


Figure 2.8. Scrum model, taken from International Scrum Institute (2017).

SCRUM has some innovative characteristics like the sprints. Sprints are complete iterative development cycles with fixed time (from two to four weeks) resulting in a deliverable product (Prikladnicki *et al.* 2014). Another innovative characteristic is the fact that all the team has power to make decisions; moreover it has daily meetings, with the overall team, in order to describe the progress since the last meeting, the problems they are facing and what is planned for the next day.

The Scrum Framework in its simple form is best used for smaller, one-team projects. However, it has also been used in bigger multi-teams and/or distributed-team projects (Boehm and Turner 2004).

Therefore, as could be seen on this subsection, Agile organisations typically do not completely rely on the RE process elements and activities described in Section 2.2.1, given that these activities focus on documentation and planning. Requirement engineering in Agile Methods will be explained with more details hereafter.

2.2.3.1 REQUIREMENTS ENGINEERING IN AGILE METHODS

Agile approaches consider the project and the implementation as central activities of the software development. They incorporate other activities like requirements elicitation and tests on the project and the implementation (Sommerville 2011). Specification, design, implementation and testing are inter-leaved and the outputs from the development process are decided through a process of negotiation during the software development process.

On the other hand, the Plan-Driven approach identify different stages of the software process with results related with each stage, the result of one stage is used for the planning of the next stage (Sommerville 2011). The main typical difference between Agile and Plan-Driven RE can be seen in Figure 2.9.

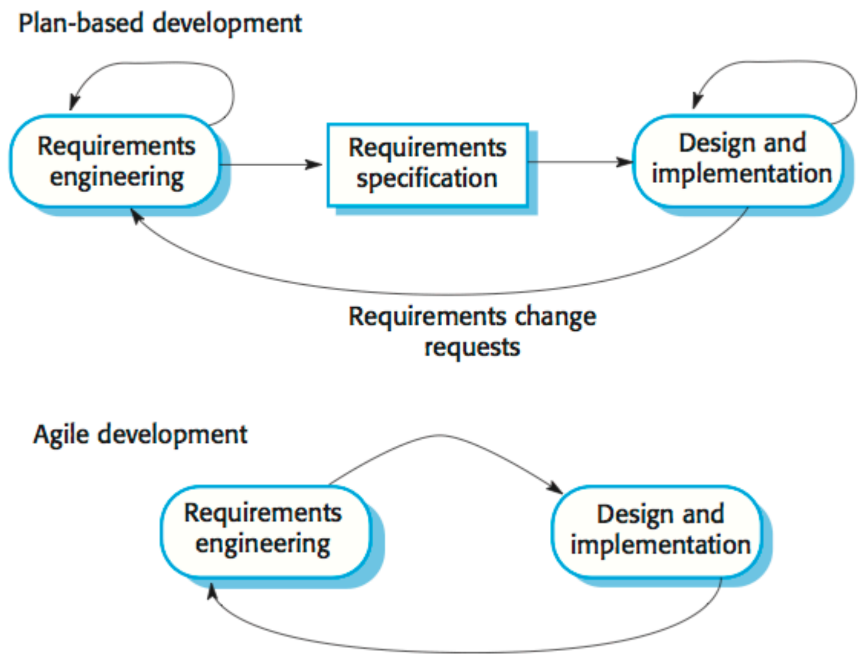

Plan-based development
Requirements engineering
Requirements specification
Design and implementation
Requirements change requests
Agile development
Requirements engineering
Design and implementation

Figure 2.9. Plan-Driven versus Agile development and RE, taken from Sommerville (2006).

According to Aguiar and Borba (2013), at the beginning of the project, in Agile methods, it is necessary to understand the scope and create the high level requirements, requirements analysis occur during all the project, a separated stage of analysis that exists on the Plan-Driven methods does not occur here. Requirements can change anytime, that is why their more formal documentation is not suggested in the scope of this methods but, if it exists, it is important for it to be up to date and useful for all the stakeholders. Besides, it is important for the requirements to be managed and well understood in order to control its changes.

Another characteristic of the agile method is related to the communication between the project team and the customer, this process model focus on the customer interaction. Reviews and feedback from practitioners show that direct and regular interaction with the customer is one of the key factors for project success (Eberlein and Prado Leite 2002).

In Plan-Driven methods all the requirements are specified, however, in agile methods only the sufficient requirements are described. The requirements described in Agile methods are the ones sufficient to determine what will be built and the implemented requirements are shown to the customer in order to verify if the software is on the right path (Smith 2009).

2.3 RELATED WORKS

In the following subsection the work related to survey research on RE problems, published before the introduction of the NaPiRE initiative, is discussed. In sequence, the NaPiRE initiative and its previously published material is presented.

2.3.1 RESEARCH ON REQUIREMENTS ENGINEERING PROBLEMS

There is a large body of research on requirements engineering problems and it is not possible to discuss them all here. However, to lay the foundations of this work, some relevant studies related with requirements engineering problems will be presented.

A well-known survey on causes for project failure is the Chaos Report of the Standish Group on cross-company root causes for project failures (The Standish Group 1994). While most of these causes are related to RE, the survey has serious design flaws and the validity of its results is questionable (Eveleens and Verhoef 2010). Additionally, it exclusively investigated failed projects and general causes at the level of overall software projects. Thus, unfortunately it does not directly support the investigation of RE problems in industry. This study was based on a previous Standish Group's work (The Standish Group 1994) that summarised Standish's research findings and aimed to investigate the general causes of software project failure.

Other studies, such as the (German) Success study (Buschermohle *et al.* 2006), conduct a similar investigation of German companies including a detailed and reproducible study design. Still, both surveys exclusively investigate failed projects and general causes at the level of overall software processes. Thus, the focus of those studies does not support the investigation of contemporary phenomena and problems of RE in industry. Nikula *et al.* (2000) present a survey on RE at the organisational level of small and medium-sized companies in Finland. Based on their findings, they inferred improvement goals, e.g., on optimising knowledge transfer.

Other studies propose techniques that try to avoid ambiguous, incomplete and inappropriate requirements. One of them is the one conducted by Olmos *et al.* (2014), which proposed the Knowledge Management on a Strategy to Requirements Engineering (KMoS-RE strategy) representing a novel RE strategy that aims to minimise those problems, transform and transfer knowledge. According to the authors, this strategy helps to internalise the domain knowledge, to clarify the solution idea, to reduce the ignorance of symmetry, to structure the domain knowledge, and to detect and correct wrong beliefs about the domain. Also, Li *et al.* (2016) developed a decision support framework METRO for the Next Release Problem (NRP) to manage algorithmic uncertainty and requirements uncertainty.

Some surveys have been focusing specifically on RE problems in industry. These surveys include the one conducted by Hall *et al.* (2003) in 12 software organisations. They empirically underpin the problems discussed by Hsia *et al.* (1993) and investigated a set of critical organisational and project-specific problems, such as communication problems, inappropriate skills or vague requirements. Their findings, among others, suggest that most RE problems are mainly organisational rather than technical. Another example of study focused on RE problems on industry is the one by Al-Rawas and Easterbrook (1996), which presents a field study on communication problems in requirements engineering.

Furthermore, in 2003, Beecham *et al.* conducted a study of the problems 12 software companies experienced in software development. On this study, they looked at how different practitioner groups respond to software process improvement problems in order to develop a more holistic understanding of the problems practitioners are experiencing on their attempts to improve their software process. Another study conducted by Li (2016) proposed a decision support framework for analysing uncertainty in requirements selection and optimisation.

Taheri *et al.* (2015) introduces a knowledge audit (KA) model that supports knowledge communication among stakeholders through objectively assessing the knowledge in the requirements elicitation process. The knowledge communication is generally difficult because of the various knowledge background of the stakeholders, this fact leads to different ways of knowledge expression affecting the understand ability and causing ambiguity. Finally, Oran (2016) focused her research on the communication problems within the project team and proposed the creation of a set of artefacts and models to support the communication of requirements.

Moreover, some researchers conducted country-specific investigations on RE problems. Solemon *et al.* (2009) conducted a survey about RE problems in small organisations in Malaysia. Liu *et al.* (2010) also conducted a study about RE problems in Chinese organisations. More recently, Khankaew and Riddle (2014) reported a study involving semi-structured interviews with small and medium-sized organisations from Thailand.

However, each of those studies focused on specific aspects in RE and they mainly reported problems identified for their investigation scenario, without an in-depth discussion on their causes and mitigation actions for their prevention. Additionally, the studies are completely independent and their results are isolated and not generalizable.

To address these issues, the NaPiRE project was launched in a collaboration of researchers from different countries (Méndez Fernández and Wagner 2015). The next subsection will explain with more details about this global survey.

2.3.2 THE NAPIRE PROJECT

The NaPiRE project started in 2012 as a reaction to the lack of a general empirical basis for requirements engineering research (Méndez Fernández *et al.* 2015). The initiative resulted in the design of a global family of surveys to overcome the problem of isolated investigations in RE that are not representative (Méndez Fernández and Wagner 2015). A long-term goal of the project is to establish an empirically sound basis for understanding trends and problems in RE (Méndez Fernández and Wagner 2015). This survey is currently being replicated in several countries around the globe. All up-to-date information on NaPiRE together with links to all publications and the data is available online[1].

The survey questionnaire, that can be found in the ANNEX 1, contains 35 questions gathering the following type of data from the responding organisations: (a) general information, (b) RE status quo, (c) RE improvement status quo, (d) RE problems faced in practice, and (e) RE problem manifestation (e.g., causes, impact, and mitigation actions). Every two years, this survey has been run, the first trial was conducted only in Germany and Netherlands (2012-2013) (Méndez Fernández and Wagner 2015). The second trial was conducted in several countries, including countries from North America, South America, Europe and Asia (2014-2015) (Méndez Fernández *et al.* 2016).

Some research was already conducted based on the data from the second trial of the NaPiRE survey. In (Kalinowski *et al.* 2015) data from 74 Brazilian organisations was used to identify critical RE problems and their main causes. The most critical RE problems, according to those organisations, are related to communication and to incomplete/hidden or underspecified requirements. Furthermore, they provided probabilistic cause-effect diagrams (those diagrams were introduced in (Kalinowski *et al.* 2008) and refined in (Kalinowski *et al.* 2011)) to organise knowledge on common causes of the five most critical identified RE problems.

In (Kalinowski *et al.* 2017), a new Defect Causal Analysis (DCA) approach that uses NaPiRE cross-company data was proposed. They collected data on causes of requirements engineering problems from 74 Brazilian organisations and built a Bayesian network concerning problem manifestation. They evaluated the approach in academia, with industry representatives of the Fraunhofer Project Center at UFBA, and in an industrial case study at the Brazilian National Development Bank (BNDES) and gathered a promising feedback.

[1] www.re-survey.org

In (Méndez Fernández *et al.* 2015), an analysis of the similarities and the differences in the problems experienced in Brazil and Germany was made. In this paper, it was possible to observe that the dominating factors are related to human interactions (e.g., based on the project size and process model) rather than country. Furthermore, a higher inclination to standardised development process models in Brazil and slightly more non-agile, plan-driven RE in Germany was observed. Again incomplete/hidden requirements and poor communication were among the most critical reported RE problems in both countries.

In (Kalinowski *et al.* 2016), a first step was taken into RE problem prevention by further analysing the reported common causes and mitigation actions for the specific problem of incomplete/hidden requirements based on the Austrian and Brazilian data. Finally, in (Mafra *et al.* 2016), already in the context of this dissertation, the causes and mitigation actions of the most critical RE problems were analysed based on each type of organisations (clustered by size and process model) and preliminary guidelines for each critical RE problem were conceived.

In this dissertation the effort done in (Kalinowski *et al.* 2016) and (Mafra *et al.* 2016) based on the findings in (Méndez Fernández *et al.* 2015) is extended, by looking at the complete data set with answers from 228 organisations and by the most critical RE problems, its causes and mitigation actions by clusters of process model, independently of the country. Therefrom, the goal of this work is to derive guidelines for organisations in each of those clusters to help them in preventing those problems. Additional evaluations and refinements of the preliminary guidelines were also performed.

2.4 FINAL REMARKS

This chapter presented the theoretical foundation of requirements engineering, explaining the concepts of requirements management and requirements development, which are considered subcomponents of RE. This chapter also detailed how the RE is typically implemented in organisations that follow the Agile methods and on organisations that follow the Plan-Driven methods.

The previous work related to RE problems were also outlined. However, none of those previous works conducted a real analysis of the most common problems that one can encounter and on how to avoid them. In order to fulfil this miss of a complete study on RE problems, the NaPiRE project, also presented in this chapter, designed a globally distributed family of surveys to lay an empirical and externally valid foundation about the state of the practice in RE. Based on this survey, a deep analysis of its answers was made in order to propose guidelines that will be useful to avoid the critical problems reported by the respondents of the NaPiRE project. The research method used to move from the NaPiRE data towards the guidelines for preventing the critical RE problems will be described on the next chapter.

CHAPTER 3 – PRELIMINARY GUIDELINES FOR PREVENTING CRITICAL RE PROBLEMS

3.1 INTRODUCTION

In the last chapter, typical RE practices were described for Agile and Plan-Driven methods. However, in both methods, critical problems may occur during the RE process. This phase of the software development is recognised as critical, since many software failures originate from inconsistent, incomplete or simply incorrect requirements specifications (Wahono 2003). Many of the serious and common problems associated with software development are related to requirements (Wahono 2003).

Given this scenario, aiming to assist organisations on their RE process improvement and problem prevention efforts, a study on the most critical RE problems, their causes and possible mitigation actions was conducted to propose guidelines to prevent them from occurring and, consequently, diminish the number of project failures.

This chapter describes how preliminary guidelines were built based on the NaPiRE data and presents those preliminary proposed guidelines for preventing the most critical RE problems organisations typically face every day.

3.2 ANALYSING THE NAPIRE DATA

This section describes how the NaPiRE project data was treated and analysed in order to move towards the preliminary guidelines.

ANNEX 1 presents the full questionnaire distributed on the last run of the NaPiRE survey (2014-2015). It constitutes in total 35 questions focusing on the problems practitioners experience in their RE. One of the goals of Méndez Fernández *et al.* (2015) related to RE problems was to use the data to answer the following question: What are observable patterns of problems and context characteristics?

The questionnaire presented the respondents a list of problems practitioners are meant to typically encounter in practice. This list emerged from an analysis of previously conducted studies related to this theme and includes a set of 21 pre-compiled problems shown next in alphabetic order (Méndez Fernández *et al.* 2015):

- Communication flaws within the project team
- Communication flaws between the project team and the customer

- Discrepancy between high degree of innovation and need for formal acceptance of (potentially wrong / incomplete / unknown) requirements
- Gold plating (implementation of features without corresponding requirements)
- Incomplete and / or hidden requirements
- Inconsistent requirements
- Insufficient support by project lead
- Insufficient support by customer
- Missing traceability
- Moving targets (changing goals, business processes and / or requirements)
- Stakeholders with difficulties in separating requirements from previously known solution designs
- Technically unfeasible requirements
- Terminological problems
- Time boxing / Not enough time in general
- Unclear responsibilities
- Unclear / unmeasurable non-functional requirements
- Underspecified requirements that are too abstract and allow for various interpretations
- Volatile customer's business domain regarding, e.g., changing points of contact, business processes or requirements
- Weak access to customer needs and / or (internal) business information
- Weak knowledge of customer's application domain
- Weak relationship to customer

When answering the survey, respondents were asked to report the relevance of the presented problems for their project setting, before being asked to select the 5 most critical ones. Thereafter, subsequent questions, concerning the 5 most critical faced problems, were related to the causes, effects, potential mitigation strategies and the indication of the ones that led to project failure.

In total, 354 organisations spread across 10 different countries agreed to answer the NaPiRE survey. The 354 answers gathered from the NaPiRE survey comprehend mostly data from business analysts, project managers and requirements engineers. Furthermore, approximately 70% of the total amount of respondents can be considered as experts, having more than 3 years of experience in their roles. Out of the 354 answers, 228 (63%) completed the survey by going through all of its questions and successfully reaching its end. Only data of the 228 organisations that fully answered the questions was used in the context of this work.

The guidelines were built focusing on the organisations by cluster of process model (agile and plan-driven) in order to further investigate how the problems manifest within such clusters, aiming at identifying potential RE problem patterns. Furthermore, the type of process model can interfere on the way requirements engineering is practiced. According to Wagner *et al.* (2017) RE is considerably different in agile development than in more traditional development processes. Leffingwell (2011) claims that no matter the specific method, agile's treatment of requirements is fundamentally different.

To achieve such goal some treatment had to be done on the data collected from the questionnaire. The respondents were presented with multiple-choice questions concerning the process model they follow in their projects with the following options: RUP, Scrum, V-Model XT, Waterfall, XP, and Other (in this case informing textually which process model they use) (Méndez Fernández *et al.* 2015). The answers were analysed and those process models were grouped into two clusters: agile (Scrum and XP) and plan-driven (RUP, V-Model XT and Waterfall). Out of the 228 organisations that completed the questionnaire, 196 selected one of the five predefined options as their process model. Table 2 shows the number of organisations per process model group.

Table 2. Number of Organisations Per Process Model

Process Model	Total
Agile	92
Plan-Driven	46
Mixed	58
Total	196

The amount of organisations using mixed process models, i.e. that reported their process fitting into options of both groups is large. However, the mixed ones were excluded from the analyses to remove a potential confounding factor, as, in those cases, we had no information on the extent to which each process model is applied in the organisation. Hence, the total number of answers analysed by each organisation grouped by process model were 138, excluding the mixed ones.

3.3 TOP RE PROBLEMS, CAUSES, AND MITIGATION ACTIONS

Based on the set of 21 pre-defined general RE problems listed in the NaPiRE questionnaire, the respondents were asked to rank the five most critical ones. They were also asked to inform, for each of the critical problems, whether the problem led to project failure or not. Based on this information, the total number of citations of each problem as a critical one and how often the problem was reported as leading to project failure, within each cluster, were compiled. Therefrom, the problems within each cluster were ranked from the most cited to the less cited ones.

To keep the focus of the guidelines on the main problems within each cluster of process model it was decided that the guidelines should address the 5 most frequently cited (critical) problems within each cluster. The top 5 most critical RE problems, as informed by the 138 respondents, can be visualised in Table 3.

Table 3. Most critical RE problems within each cluster of process model.

Process Model	
Agile	**Plan-Driven**
1. Incomplete and/or hidden requirements	1. Incomplete and/or hidden requirements
2. Communication flaws between the project team and the customer	2. Communication flaws between the project team and the customer
3. Moving targets (changing goals, business processes and / or requirements)	3. Underspecified requirements that are too abstract and allow for various interpretations
4. Time boxing / Not enough time in general	4. Moving targets (changing goals, business processes and / or requirements)
5. Underspecified requirements that are too abstract and allow for various interpretations	5. Communication flaws within the project team

Further details on the frequency in which the top 5 most critical RE problems are meant to lead to project failure and the percentage of its citation are shown on Table 4. Because of different points of view between respondents from Plan-Driven and Agile organisations concerning the problems criticality, to achieve a unique list of problems, the top 5 rank of the most critical RE problems needed to be converted into a top 6. This change can be explained through the analysis of the Table 4 where it is possible to observe that the *Time boxing* problem was considered the 4th most critical RE problem for Agile organisations and the 6th for Plan-Driven organisations, on the other hand, the *Communication flaws within the project team* problem was considered the 6th most critical for Agile and the 5th for Plan-Driven organisations. The complete table with the rank of criticality for all the 21 RE problems can be seen in APPENDIX A.

Table 4. Rank of the 6 most critical RE Problems

		Process Model	
Problems		**Agile** (92 citations)	**Plan-Driven** (46 citations)
Incomplete and/or hidden requirements	Ranking Position	1	1
	Citations	45 (49%)	24 (52%)
	Project Failure	22 (49%)	8 (33%)
Communication flaws between the project team and the customer	Ranking Position	2	2
	Citations	42 (46%)	19 (41%)
	Project Failure	22 (52%)	12 (63%)
Moving targets (changing goals, business processes and / or requirements)	Ranking Position	3	4
	Citations	30 (33%)	17 (37%)
	Project Failure	16 (53%)	6 (35%)
Time boxing / Not enough time in general	Ranking Position	4	6
	Citations	30 (33%)	13 (28%)
	Project Failure	12 (40%)	3 (23%)
Underspecified requirements that are too abstract and allow for various interpretations	Ranking Position	5	3
	Citations	30 (33%)	17 (37%)
	Project Failure	8 (27%)	9 (53%)
Communication flaws within the project team	Ranking Position	6	5
	Citations	28 (30%)	16 (35%)
	Project Failure	12 (43%)	5 (31%)

Table 4 shows, for instance, that the problem *Incomplete and/or hidden requirements* was the one most often cited as a critical problem for both Agile and Plan-Driven organisations. It was the #1 in the rank being cited as critical by 49% (45 out of 92 Agile organisations) and by 52% (24 out of 46 Plan-Driven organisations). 49% (22 out of 45 Agile organisations) and 33% (8 out of 24 Plan-Driven organisations) cited that the problem reported might lead to project failure.

Also from the table, it is possible to observe that the most critical RE problems are also the most leading to project failure, concerning to agile organisations respondents. *Incomplete and/or hidden requirements* and *Communication flaws between the project team and the customer* are the problems that, according to the respondents, are more likely to lead to project failure with 22 citations each. However, according to plan-driven organisations respondents, the problems that often lead to project failure are *Communication flaws between the project team and the customer* and *Underspecified requirements that are too abstract and allow for various interpretations* with 12 and 9 citations, respectively, being closely followed by *Incomplete and/or hidden requirements* with 8 citations. Indeed, under specification and incompleteness of requirements are closely related.

Concerning the occurrence of the problems within the clusters, shown in Table 4, it is important to highlight the *Incomplete and/or hidden requirements* and *Communication flaws between the project team and the customer* problems. Those problems were classified as the two most cited problems in both clusters. One explanation for this behaviour is that plan-driven organisations might tend to have concrete and approved communication plans, which might not be easy to follow in all organisational and project contexts. Organisations in which the respondents follow agile process models on the other hand could rely on much more informal communication with the customer. If this communication is not achieved in practice, it constitutes a relevant problem for projects conducted in this manner (Mafra *et al.* 2016).

The free-text answers show the reason for the occurrence of the *Incomplete and/or hidden requirements* problem, reporting that it happens in agile organisations because of its agility characteristic that leads to a lack of pattern and formal rules, beyond the harm of the requirements detailed elaboration. On the other hand, for plan-driven organisations, the respondents reported that their problem relies mostly on its formality ("lack in review process", "inflexible planning" and the fact that "a Product Requirements Document is generated up front but it is not always formally updated as changes come through") as some causes for this problem. Some effects of this problem, according to the respondents are "rework", "problems in live environment" and "product different from the expected by the customer", for instance.

For the *Communication flaws between the project team and the customer* problem, respondents said that organisations tend to split the work in several teams of which some work directly with customers, while others don't. One respondent describes that their "sales or account teams, product managers [...] act as proxies for the end user" (Méndez Fernández *et al.* 2015). Some agile companies seem to suffer especially from customers not willing to participate with a considerable amount of time ("Not enough customers willing to help out and also time constraints", "Customer is busy and skips meetings." or "Customers have no time to explain what they actually need"). The plan-driven organisations that rated this problem as important did not show a consistent pattern of reasons in their free-text answers (Méndez Fernández *et al.* 2015).

Concerning the *Moving targets* problem, plan-driven organisations cite this problem often in comparison to agile ones that supports the basic premise of agile software development, which helps to quickly adapt to changing needs. The respondents from plan-driven organisations mention as reasons the "Lack of change management on the customer side", the "Unclear business vision and understanding by stakeholders" and overall "badly written requirements". The negative effects on their projects are manifold including "project delays; extended engagement of resources beyond original plan; customer dissatisfaction" and "expensive projects, time consuming implementation, bad quality" (Méndez Fernández *et al.* 2015).

Another critical problem is *Time boxing*. In both agile and plan-driven organisations it was possible to find three (related) reasons for this prevalence of time boxing problems: bad estimations, unrealistic release dates and scope changes. The respondents often mentioned that estimations were not accurate: "A combination of bad planning and bad estimation of time for development" or "Bad estimates, unrealistic expectations". Especially sales and marketing is blamed for promising unrealistic dates: "Sales shouldn't give wishful promises" or "Release dates are sometimes arbitrary and often released early to customers creating a hard deadline". At last, frequent scope changes seem to contribute to this problem: "Last minute changes; change of priority; Business urgency" (Méndez Fernández *et al.* 2015).

Following, another critical problem, according to the respondents is *Underspecified requirements that are too abstract and allow for various interpretations.* Some of the reasons for this problem to be placed on the top 6 most critical RE problems lay on the fact that in most of the cases the communication between the team and the customer is not sufficient, besides the requirements are poorly written in both, agile and plan-driven organisations.

“Succinct and not clear specifications” and “lack of a pattern and expertise” were some of the answers given by the respondents on the survey as causes for triggering this problem.

Finally, the last critical problem from the top 6 list is the *Communication flaws within the project team*. Respondents from both, Agile and Plan-Driven organisations reported that the team distribution in many sectors or by functionality is one of the reasons of this problem’s occurrence. They also quoted communication mismatch and information loss on tight schedule as some difficulties related to Plan-Driven organisations. For Agile organisations, “Lack of internal process pattern” and “Business analysts and developers don’t share the same language” are some example of reasons for this communication problem.

After determining the 6 most critical RE problems according to the NaPiRE answers, the next step was to analyse the causes and mitigation actions for each of the 6 problems that appear as top 5 problems in the different clusters. It is noteworthy that when analysing the causes and mitigation actions for a problem in a given cluster, we only considered the answers on causes and mitigation actions provided by respondents within this same cluster. This decision was taken considering that the causes and mitigation actions could be considerably different for organisations in the different clusters.

The respondents informed both, causes and mitigation actions, in plain text as answers to open questions. Therefore, to analyse the answers given to the open questions on causes and mitigation actions, were applied textual coding techniques as recommended by (classic) Grounded Theory (Adolph, Hall and Kruchten 2011), generating and peer-reviewing codes (representing key characteristics) for each of the open text answers. Hereafter the codes for causes and mitigation actions for the 2 most critical RE problems shown on Table 4 are presented. The codes for the causes and mitigation actions of the other problems can be seen in APPENDIX B. Table 5 and

Table 6 show the causes and mitigation actions for agile and plan-driven organisations. The coded causes and mitigation actions are shown in alphabetic order, together with the number of times each code was cited by the respondents of the survey. For the number of citation, the total amount of organisations per cluster, that fully responded the survey, was considered. There were also empty answers and they could mean that they are not aware of any specific cause.

Table 5. Causes and Mitigation Actions for Agile organisations on the 2 most critical RE problems

	Agile	
Problems	**Causes**	**Mitigation Actions**
Communication flaws between the project team and the customer	• Communication flaws between team and customer (2) • Complexity of domain (1) • Complexity of RE (1) • Conflicting stakeholder viewpoints (1) • Insufficient agility (1) • Insufficient resources (1) • Lack of experience of RE team members (2) • Lack of time (1) • Language barriers (3) • Missing completeness check of requirements (2) • Missing customer involvement (1) • Missing direct communication to customer (4) • Missing engagement by customer (4) • Missing IT project experience at customer side (1) • Missing RE awareness at customer side (1) • Missing requirements specification template (1) • Strict time schedule by customer (1) • Stakeholders lack business vision	• Conduct daily meetings (1) • Create a requirements specification template (1) • Define a common structure to describe and explain requirements (1) • Improve customer commitment (5) • Improve Requirements Specification (1) • Increase the communication with customer (3) • Introduce an agile methodology (1) • Introduce an early feedback loop with customer (4) • Negotiate more time with the customer if necessary (1) • Plan and execute regular communication events/ meetings (6) • Plan and execute trainings (in order to improve skill and performance) (2) • Use mock-ups (1) • Use prototyping (2)

	and understanding (1) • Subjective interpretations (1) • Too high team distribution (3) • Weak management at customer side (1)	
Incomplete and/or hidden requirements	• Communication flaws between team and customer (2) • Complexity of RE (1) • Customer does not know what he wants (1) • Insufficient agility (2) • Insufficient analysis at the beginning of the project (1) • Insufficient information (1) • Insufficient resources (1) • Lack of a well-defined RE process (2) • Lack of experience of RE team members (3) • Missing company wide standard (1) • Missing completeness check of requirements (1) • Missing domain knowledge (2) • Missing IT project experience at customer side (1) • Missing knowledge about development framework (1) • Missing RE awareness at customer side (1) • Missing requirements specification template (1) • Missing of a global view of the	• Acquire (external) requirements experts (2) • Create a Definition of Ready to the team (1) • Create a Process Model (1) • Create a requirements specification template (3) • Explore use cases and scenarios during the RE (2) • Improve customer commitment (1) • Implement a change management process (1) • Increase requirements analysis (1) • Increase the communication with customer (1) • Integrate Testing and RE (1) • Introduce and use checklists for monitoring requirements along their life cycles (1) • Introduce an early feedback loop with customer (3) • Spend more time in analysis before commitment (1) • Not to chose incomplete requirements to implement (1)

	system (1) • Poor requirements elicitation techniques (1) • Requirements remain too abstract (1) • Stakeholders lack business vision and understanding (1) • Subjective interpretations (1) • Unavailability of requirements engineer (1) • Unclear business needs (1) • Unclear project scope (1) • Unclear roles and responsibilities at customer side (2) • Volatile requirements (1) • Weak qualification of RE team members (3) • Weak qualification of stakeholders (1)	• Plan and execute regular communication events/ meetings (1) • Plan and execute trainings (in order to improve skill and performance) (2) • Use prototyping (3) • Spend more time on requirements specifications (1) • Spend more time on requirements validation (1)

Table 6. Causes and Mitigation Actions for Plan-Driven organisations on the 2 most critical RE problems

	Plan-Driven	
Problems	**Causes**	**Mitigation Actions**
Communication flaws between the project team and the customer	• Communication flaws between team and customer (4) • Conflicting stakeholder viewpoints (1) • Lack of a well-defined RE process (1) • Lack of time (1) • Language barriers (2)	• Avoid using technical terms during meetings with customers (1) • Centralise communication not allowing extra official communications (2) • Improve customer commitment (2)

	• Missing involvement of developers (1) • Missing tool support (1) • Not following the communication plan (1) • Poor project management (1) • Requirements remain too abstract (1) • Subjective interpretations (1) • Too high team distribution (2)	• Increase documentation quality (1) • Increase Requirements Elicitation methods (1) • Increase the communication with customer (2) • Introduce a leader / manager for the delivery team (1) • Introduce an agile methodology (1) • Introduce communications tools (1) • Involve the production team (1) • Plan and execute regular communication events/ meetings (3) • Plan and execute trainings (in order to improve skill and performance) (1) • Promote a knowledge transfer within the project team (1)
Incomplete and/or hidden requirements	• Complexity of project (1) • Conflict of interests at customer side (1) • Customer does not know what he wants (1) • High workload (1) • Insufficient planning of RE (1) • Lack of discipline (1) • Lack of experience of RE team members (2) • Lack of time (4)	• Acquire (external) requirements experts (1) • Create a documentation of models and solutions (1) • Evaluate and introduce tools (1) • Explore use cases and scenarios during the RE (1) • Implement a pair developers review (1) • Increase documentation quality (1)

	• Missing completeness check of requirements (1) • Requirements remain too abstract (1) • Strict time schedule by customer (1) • Thinking in legacy systems (1) • Unclear business needs (1) • Unclear roles and responsibilities at customer side (1) • Unclear terminology (1) • Unexpected changes in requirements (1) • Weak qualification of RE team members (3)	• Increase Requirements Elicitation methods (1) • Introduce software inspections (1) • Perform cross checks with solution designs (1) • Plan and execute regular communication events/ meetings (2) • Plan and execute trainings (in order to improve skill and performance) (4) • Promote a knowledge transfer within project team (1) • Use prototyping (1) • Use sign-offs before implementation (1) • Use stronger formal reviews (2)

From Table 5 and

Table 6 it is possible to observe that, for Incomplete and/or hidden requirements problem, the main reported causes are Lack of time, Lack of experience of RE team members, and Weak qualification of RE team members, but this figure changes within the specific clusters, where some causes were commonly reported. This characteristic repeated itself on the Communication flaws between the project team and the customer problem. There, Language barriers and Too high team distribution were the main causes reported by the respondents and their citation number also changed within the clusters. What can also be seen, even implicitly, are the cycles in the causes and the problems, i.e. some of the causes are, in fact, problems; for instance, Communications flaws between the project team and the customer is given as one problem, but also named by our respondents as a cause.

As could be seen in Table 5 and

Table 6 and confirmed on the APPENDIX B, some of the most cited causes are more related to some specific problems than to others. Typical examples that can be seen are Lack of time leading mainly to the *Time boxing* problem, Lack of experience of RE team members leading mainly to *Incomplete and / or hidden requirements* and *Underspecified requirements*, or too high team distribution leading mainly to *Communication flaws within the project team*.

To provide a more complete view on the causes reported for some of the most critical RE problems, in particular, the two most cited ones (which are also the most cited ones for project failure), *Communication flaws between the project team and the customer* and *Incomplete and / or hidden requirements*, an adapted Ishikawa diagram (Ishikawa 1982) (Kalinowski *et al.* 2011) was built using the cause categories suggested in guidelines for defect causal analysis in (Kalinowski *et al.* 2012): *input, method, organisation, people,* and *tools*. It is noteworthy to mention that, the diagrams were also used to represent relative frequencies, i.e. how often each cause was cited out of the total citations.

Figure 3.1 and Figure 3.2, respectively, show the diagrams for the causes of the *Communication flaws between the project team and the customer* and *Incomplete and/or hidden requirements* problems in Agile organisations. For instance, in Figure 3.1, we can see that the most frequently cited causes were related to the categories Input (~38%, i.e. 16 out of 42 reported causes were from that category) and Method (~29%). The most frequent reported causes for this problem are the Missing engagement by customer (~12%), Missing direct communication to customer (~12%), Too high team distribution (~9%) and Language Barriers (~9%).

Concerning the *Incomplete and/or hidden requirements*, also for Agile organisations, shown in Figure 3.2, the main affected categories were Method (~32%, i.e. 14 out of 45 reported causes were from that category), Input (~30%) and People (~30%). The most frequently cited causes were Lack of experience of RE team members (~8%), Weak qualification of RE team members (~8%) and Missing domain knowledge (~5%), Unclear roles and responsibilities at customer side (~5%), Communication flaws between team and customer (~5%), Lack of a well-defined RE process (~5%) and Insufficient agility (~5%).

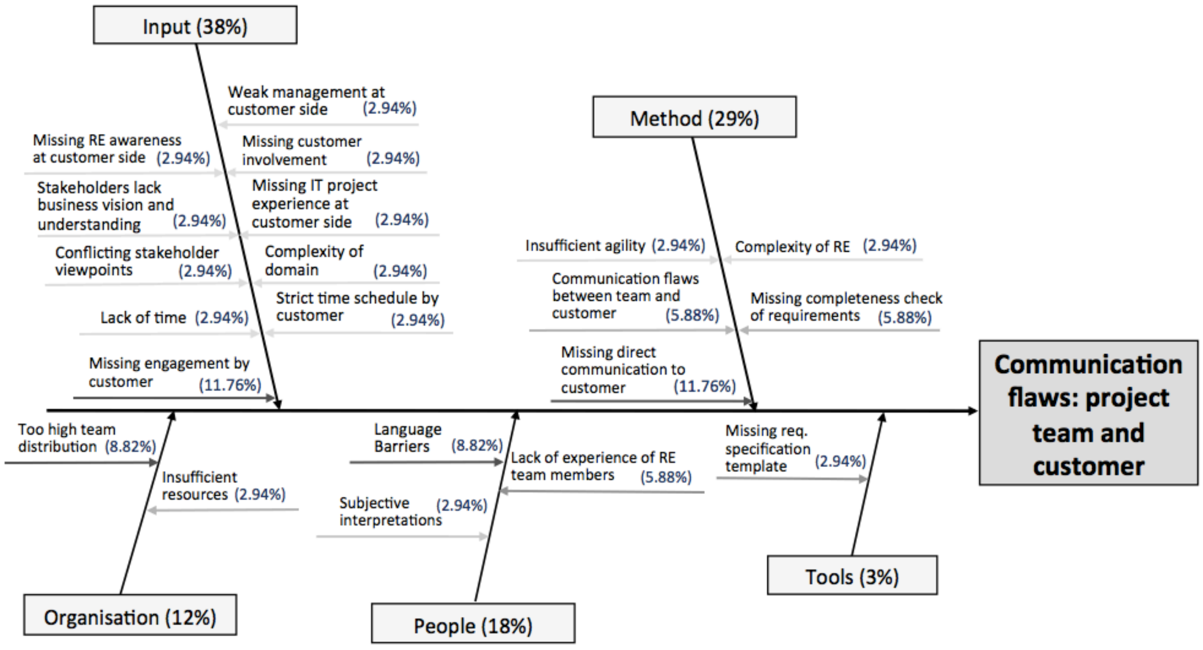

Input (38%)
Weak management at customer side (2.94%)
Missing RE awareness at customer side (2.94%)
Missing customer involvement (2.94%)
Stakeholders lack business vision and understanding (2.94%)
Missing IT project experience at customer side (2.94%)
Conflicting stakeholder viewpoints (2.94%)
Complexity of domain (2.94%)
Strict time schedule by customer (2.94%)
Lack of time (2.94%)
Missing engagement by customer (11.76%)
Method (29%)
Insufficient agility (2.94%)
Complexity of RE (2.94%)
Communication flaws between team and customer (5.88%)
Missing completeness check of requirements (5.88%)
Missing direct communication to customer (11.76%)
Communication flaws: project team and customer
Too high team distribution (8.82%)
Insufficient resources (2.94%)
Language Barriers (8.82%)
Lack of experience of RE team members (5.88%)
Subjective interpretations (2.94%)
Missing req. specification template (2.94%)
Organisation (12%)
People (18%)
Tools (3%)

Figure 3.1. Adapted Ishikawa Diagram for *Communication flaws between the project team and the customer* problem on Agile Organisations

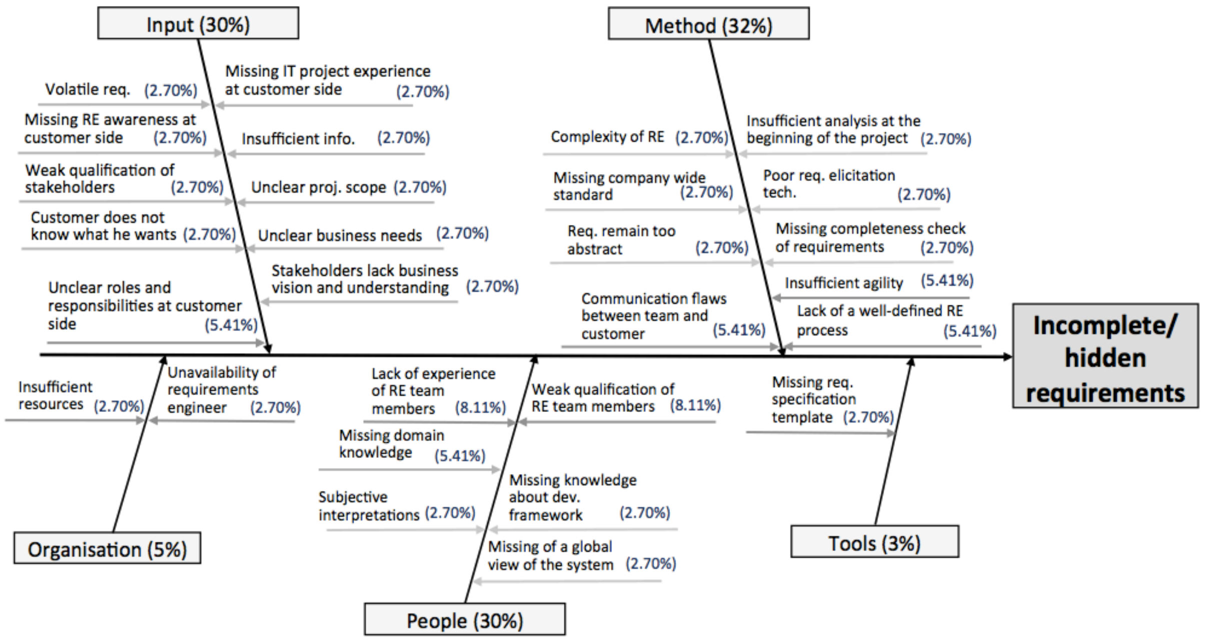

Input (30%)
Volatile req. (2.70%)
Missing IT project experience at customer side (2.70%)
Missing RE awareness at customer side (2.70%)
Insufficient info. (2.70%)
Weak qualification of stakeholders (2.70%)
Unclear proj. scope (2.70%)
Customer does not know what he wants (2.70%)
Unclear business needs (2.70%)
Stakeholders lack business vision and understanding (2.70%)
Unclear roles and responsibilities at customer side (5.41%)
Method (32%)
Complexity of RE (2.70%)
Insufficient analysis at the beginning of the project (2.70%)
Missing company wide standard (2.70%)
Poor req. elicitation tech. (2.70%)
Req. remain too abstract (2.70%)
Missing completeness check of requirements (2.70%)
Insufficient agility (5.41%)
Communication flaws between team and customer (5.41%)
Lack of a well-defined RE process (5.41%)
Incomplete/ hidden requirements
Insufficient resources (2.70%)
Unavailability of requirements engineer (2.70%)
Lack of experience of RE team members (8.11%)
Weak qualification of RE team members (8.11%)
Missing req. specification template (2.70%)
Missing domain knowledge (5.41%)
Subjective interpretations (2.70%)
Missing knowledge about dev. framework (2.70%)
Missing of a global view of the system (2.70%)
Organisation (5%)
People (30%)
Tools (3%)

Figure 3.2. Adapted Ishikawa Diagram for *Incomplete/hidden requirements* problem on Agile Organisations

Concerning the Plan-Driven organisations, Figure 3.3 and Figure 3.4, respectively, show the diagrams for the causes of the *Communication flaws between the project team and the customer* and *Incomplete and/or hidden requirements* problems of them. The Ishikawa Diagrams for all the most critical RE problems can be seen in APPENDIX C.

In Figure 3.3, we can see that the most frequently cited causes were related to the category Method (~47%, i.e. 9 out of 19 reported causes were from that category). The most frequent reported causes for this problem are the Communication flaws between team and customer (~21%), Language Barriers (~11%) and Too high team distribution (~11%).

Concerning the *Incomplete and/or hidden requirements*, also for Plan-Driven organisations, shown in Figure 3.4, the Input was the main affected category with ~42%, i.e. 10 out of 24 reported causes were from that category. The most frequently cited causes were Lack of time (~17%) and Weak qualification of RE team members (~13%).

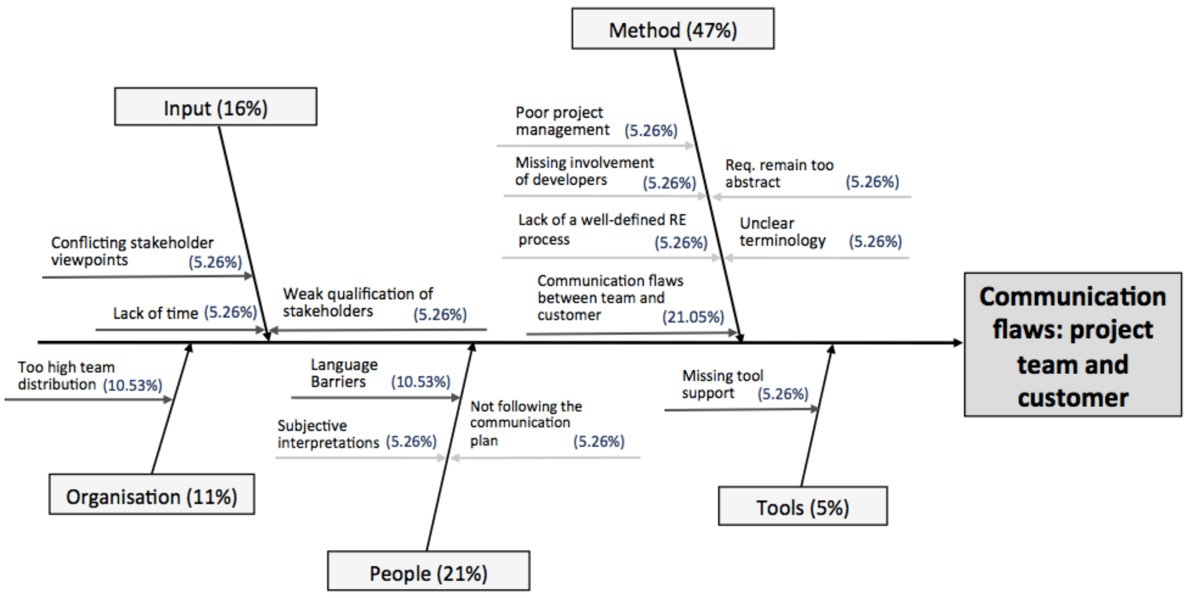

Method (47%)
Input (16%)
Poor project management (5.26%)
Missing involvement of developers (5.26%)
Req. remain too abstract (5.26%)
Lack of a well-defined RE process (5.26%)
Unclear terminology (5.26%)
Conflicting stakeholder viewpoints (5.26%)
Communication flaws between team and customer (21.05%)
Weak qualification of stakeholders (5.26%)
Lack of time (5.26%)
Communication flaws: project team and customer
Too high team distribution (10.53%)
Language Barriers (10.53%)
Missing tool support (5.26%)
Not following the communication plan (5.26%)
Subjective interpretations (5.26%)
Organisation (11%)
Tools (5%)
People (21%)

Figure 3.3. Adapted Ishikawa Diagram for *Communication flaws between the project team and the customer* problem on Plan-Driven Organisations

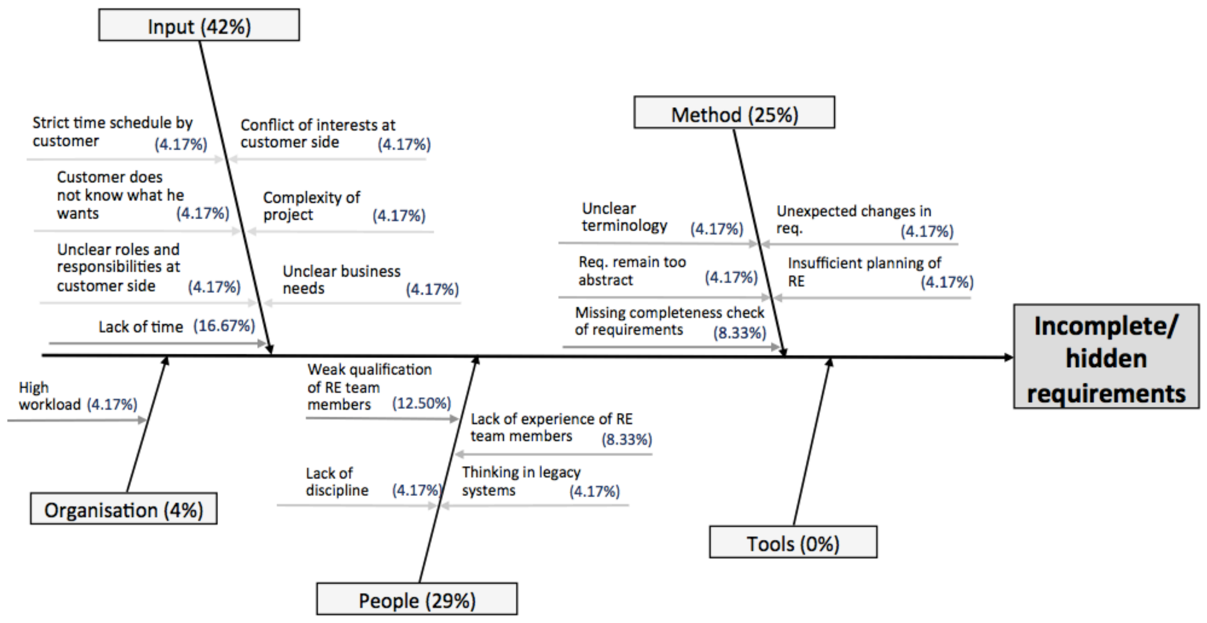

Input (42%)
Strict time schedule by customer (4.17%)
Conflict of interests at customer side (4.17%)
Customer does not know what he wants (4.17%)
Complexity of project (4.17%)
Unclear roles and responsibilities at customer side (4.17%)
Unclear business needs (4.17%)
Lack of time (16.67%)
Method (25%)
Unclear terminology (4.17%)
Unexpected changes in req. (4.17%)
Req. remain too abstract (4.17%)
Insufficient planning of RE (4.17%)
Missing completeness check of requirements (8.33%)
Incomplete/ hidden requirements
High workload (4.17%)
Weak qualification of RE team members (12.50%)
Lack of experience of RE team members (8.33%)
Lack of discipline (4.17%)
Thinking in legacy systems (4.17%)
Organisation (4%)
Tools (0%)
People (29%)

Figure 3.4. Adapted Ishikawa Diagram for *Incomplete/hidden requirements* problem on Plan-Driven Organisations

After organising the information on causes and mitigation actions for the different clusters, the next step was to analyse this information to gather further insights into the prevention of the RE problems within each cluster, assembling the results of this analysis into candidate guidelines. The outcome of this analysis will be presented in the next subsection.

3.4 PRELIMINARY GUIDELINES FOR PREVENTING CRITICAL RE PROBLEMS

The candidate guidelines for preventing the most critical problems within each cluster were obtained through analysis based on the organised information from the NaPiRE survey on causes and mitigation actions for RE problems, witch can be seen on Table 16 and Table 17, and focused on how to prevent the problems.

The mitigation actions suggested on the NaPiRE survey were used as a starting point for the analysis on how to address the causes and prevent the problems. Of course, during this analysis only suggested mitigation actions that were aligned with the proposed process model were considered, as the purpose of the guideline is not to change the process model, but to prevent its main problems in tune with the process model followed.

To illustrate how the guidelines were produced, hereafter will be described the rationale used for the *Communication flaws between the project team and the customer* problem on Plan-Driven Organisations and for the *Incomplete and/or hidden requirements* problem on Agile Organisations depicted in Figure 3.3 and Figure 3.2 respectively. This selection was made to approach both main problems and process models to expose how the guidelines were generated. Hence, not all the advices for each of the categories will be quotes next, the goal is just to show how they were produced. The overall advices included on the proposed preliminarily guidelines for each category, cluster and RE problem will be shown afterwards.

For the *Communication flaws between the project team and the customer* problem, was decided to include some advices to address each cause of the **Input** category, for instance, for the cause:

- "Weak qualification of stakeholders" the advice to "allocate a project manager to the project with large experience and expertise" and to "provide training (if needed) to improve team skills";
- "Lack of time" the advice to "improve the management of the project schedule";
- "Conflicting stakeholder viewpoints" the advice to "use prototyping".

Concerning the category **Organisation** the advices to deal with the cause "too high team distribution" were to "promote knowledge transfer within the project team" and to "work with small teams".

In the **Method** category to address the cause:

- "Lack of a well-defined RE process" were included the advices to "conduct regular meetings with the customer" and to "explain customers the importance of their contribution";
- "Poor project management" were inferred the advices to "allocate a project manager to the project with large experience and expertise" and to "improve project management practices";
- "Requirements remain to abstract" the advices to "spend more effort in elicitation, analysis and modelling";
- "Missing involvement of developers" and "unclear terminology" were included the advices of "promote the integration of the project team" and to "create a project glossary with domain concepts" and "improve the documentation of the requirements specifications", respectively.

As could be seen, the causes on the Method category were related to the documentation, the communication between the project team and the customer and the management of the project.

For causes of the category **People**, to address the causes “language barriers”, “subjective interpretations” and “not following the communication plan” inferred the advices to “avoid using technical terms during meetings with customers” and to “centralise communication not allowing extra official communications”. This category was mainly related to the barriers on the communication. Finally, for the cause of the category **Tools** the advice to “introduce communication tools (e-mails, instant messages, telephone/video conference, intranet)” was inferred.

For the *Incomplete and/or hidden requirements* problem, was decided to include, to address the **Input** category, in addition to other orientations, some advices to deal with the cause:

- “Unclear roles and responsibilities at customer side” was included the advice to “educate user representatives and managers about software requirements”;
- “Volatile requirements” the advices to “measure requirements volatility (e.g., the amount of changes)” and to “renegotiate project commitments when requirements change” were included.
- “Customer does not know what he wants” the advice to “use prototyping” was included.

Concerning the causes of the category **Organisation** the advice to “allocate a requirements engineer to the project with large experience and expertise” was included. This advice should deal with the cause “unavailability of requirements engineer”.

In the **Method** category the causes were mainly related to well-documented and analysed requirements and a good communication. To address the cause:

- “Lack of a well-defined RE process” were inferred, among others, the advices to “create a well defined RE process” and to “create a DoR (Definition of Ready) for the team, i.e., a checklist for accepting a requirement in a sprint” were inferred, for instance. Such DoR is commonly used in agile projects to avoid the beginning of work on features that do not comply with clearly defined completion criteria, which usually translates into costly rework. Nevertheless, it is noteworthy to mention that explicitly defining a rigorous RE process is not well aligned to the agile development paradigm.
- “Communication flaws between the project team and customer” the advices to “conduct regular meetings with the customer” and to “explain customers the importance of their contribution” were included;

- “Insufficient agility” the respondents suggested a bigger control and management of the schedule and time of the project in order to avoid delays. The advice suggested was to “improve the management of the project schedule”.

For causes of category **People**, to address the cause:

- “weak qualification of RE team members” and “lack of experience of RE team members” were inferred the advices to “allocate a requirements engineer to the project with large experience and expertise” and to “provide training (if needed) to improve team skills”.
- “Missing knowledge about the development framework”, the advice of “provide training (if needed) to improve team skills on the development framework”.

This category was mainly related to the lack of experience of the team and lack of knowledge about the domain. Finally, for the causes of the category **Tools** were inferred the advices to “create a requirements specification template” and to “spend more effort in requirements specification”.

The preliminary guidelines compiled with all the advices for *Communication flaws between the project team and the customer* and *Incomplete and/or hidden requirements* problems faced by agile and plan-driven organisations can be seen in Table 8 and Table 7, respectively. The preliminary guidelines for all the most critical RE problems are can be seen in APPENDIX D.

Table 7. Preliminary guidelines for preventing *Communication flaws between the project team and the customer* on Agile and/or Plan-Driven organisations

Communication Flaws between the project team and the customer	
Agile	**Plan-Driven**
• Allocate a project manager to the project with large experience and expertise • Allocate a requirements engineer to the project with large experience and expertise • Avoid using technical terms during meetings with customers	• Allocate a project manager to the project with large experience and expertise • Allocate a requirements engineer to the project with large experience and expertise • Avoid using technical terms during meetings with customers • Centralise communication not allowing

• Conduct daily stand-up meetings with the project team • Conduct regular meetings with the customer • Create a requirements specification template • Educate user representatives and managers about software requirements • Explain customers the importance of their contribution • Focus on business needs, assuring that requirements are sufficient and really necessary • Improve the documentation of the requirements specifications • Improve the management of the project schedule • Introduce an early feedback loop with customer (e.g., conducting regular demos of new aggregated value) • Negotiate more time with the customer if necessary • Promote the integration of the project team • Provide training (if needed) to improve team skills • Spend more effort in requirements specification • Use an agile method • Use prototyping	extra official communications • Conduct regular meetings with the customer • Create a project glossary with domain concepts • Create a well-defined RE process • Explain customers the importance of their contribution • Improve project management practices • Improve the documentation of the requirements specifications • Improve the management of the project schedule • Introduce communication tools (e-mails, instant messages, telephone/video conference, intranet) • Promote the integration of the project team • Promote knowledge transfer within project team • Provide training (if needed) to improve team skills • Spend more effort in elicitation • Spend more effort in analysis and modelling • Use an agile method • Use prototyping • Work with small teams

Table 8. Preliminary guidelines for preventing *Incomplete and/or hidden requirements* on Agile and/or Plan-Driven organisations

Incomplete and/or hidden requirements	
Agile	**Plan-Driven**
• Allocate a requirements engineer to the project with large experience and expertise • Analyse technical feasibility of the requirements • Conduct daily stand-up meetings with the project team • Conduct regular meetings with the customer • Create a DoR (Definition of Ready) for the team, i.e., a checklist for accepting a requirement in a sprint • Create a requirements specification template • Create a requirements traceability matrix • Create a well-defined RE process • Educate user representatives and managers about software requirements • Explain customers the importance of their contribution • Explore use cases and scenarios during requirements engineering • Implement a requirements change management process • Improve the management of the project schedule • Integrate Testing and RE • Introduce an early feedback loop with	• Allocate a requirements engineer to the project with large experience and expertise • Allocate the team members to work no more than 40 hours a week to maintain the productivity • Avoid using technical terms during meetings with customers • Conduct regular meetings with the customer • Create a project glossary with domain concepts • Educate user representatives and managers about software requirements • Establish a baseline and control versions of requirements documents • Explore use cases and scenarios during requirements engineering • Focus on business needs, assuring that requirements are sufficient and really necessary • Improve the documentation of the requirements specifications • Improve the management of the project schedule • Perform requirements reviews (e.g., peer reviews, technical reviews, inspections) • Promote knowledge transfer within project team • Provide training (if needed) on the

customer (e.g., conducting regular demos of new aggregated value) • Measure requirements volatility (e.g., the amount of changes) • Provide training (if needed) on the business domain and overall system scope • Provide training (if needed) to improve team skills • Renegotiate project commitments when requirements change • Spend more effort in analysis and modelling • Spend more effort in elicitation and analysis • Spend more effort in requirements specification • Spend more effort in requirements validation • Track the status of each requirement • Use a requirements management tool • Use prototyping	business domain and overall system scope • Provide training (if needed) to improve team skills • Reuse requirements across projects • Review lessons learned regarding requirements on other projects • Review the checklist of problems that may occur from similar system built before or if the checklist doesn't exists, create one to help future projects • Spend more effort in analysis and modelling • Spend more effort in elicitation • Spend more effort in requirements validation • Use prototyping • Write a vision and scope document

The presented guidelines for preventing the most critical RE problems (*Communication flaws between the project team and the customer* and *Incomplete and/or hidden requirements*), as mentioned before, are preliminary ones. To evaluate their adequacy to address the RE problems in the proposed contexts an additional evaluation was conducted involving experts from the academy to refine the guidelines. This evaluation and the adjusted guidelines will be presented in the next chapter.

3.5 FINAL REMARKS

This chapter presented the preliminary guidelines for preventing the most critical RE problems along with the involved process to produce them.

As could be seen, the first step towards the guidelines generation was to select only the answers of the respondents that fully answered the NaPiRE survey and divide them in clusters of process model (Agile and Plan-Driven). Next, their answers were analysed in order to determine which of the 21 RE problems were considered the most critical RE problems by the respondents in each cluster of process model. With the list of the most critical problems, the six most critical ones were selected and their causes and mitigation actions were analysed in order to propose the advices that would be included on the preliminary guidelines. The rationale used to produce these guidelines was provided for the two most critical problems

However, the result concerns preliminary guidelines that need further evaluation. In the context of this dissertation we conducted an initial evaluation and the refinement of the guidelines for the two most critical RE problems (Incomplete and/or hidden RE problems and Communication flaws between the project team and the customer). Therefore, a survey involving experts from the academy from different countries was conducted and will be presented in details in the next chapter. The selection of the two most critical RE problems was done considering the effort required by the experts to evaluate the proposed action items of the guidelines. Evaluating the guidelines for more problems would require additional survey rounds, which would not be reasonable to conduct within the timeframe of a dissertation.

CHAPTER 4 – EVALUATION AND ADJUSTED GUIDELINES

4.1 INTRODUCTION

In the previous chapter, the preliminary guidelines for the most critical RE problems for Agile and Plan-Driven organisations were presented. However, those guidelines needed to be evaluated in order to assess their adequacy to address the RE problems in the proposed (agile or plan-driven) contexts.

There are three main types of evaluation strategies: surveys, case studies and formal experiments. According to Pfleeger (2001), surveys are an empirical strategy for collecting information from people, where usually the person applying the survey have the control over the situation at hand. Case studies provide the person with a good understanding of what actually happens in the real world, these results would be difficult to achieve with any other research method. In case studies, some phenomenon is chosen to be studied within its real-life setting (Runeson *et al.* 2012). Finally, formal experiments are the most controlled type of study. In this type of evaluation several methods are used to reduce bias and eliminate confounding factors so cause and effect can be evaluated with some confidence (Wohlin *et al.* 2012).

In order to initiate the evaluation of the preliminary guidelines, all the evaluation strategies were analysed in an attempt to decide which of them was the most suitable for the scenario at hand. Referring to the formal experiment, during the discussions was analysed how such experiment would have to be prepared to evaluate the guidelines. As the guidelines are advices to avoid critical RE problems that organisations face constantly, the experiment would need to simulate a real-world project development with its common problems and difficulties. Creating such scenario to simulate a real project development process is very difficult since, in real projects, several unpredictable situations may occur during the development lifecycle. The hardness of simulating this kind of environment confirms the fact that a formal experiment, in this case, would lead to serious threats of validity. The construct and external validity, for instance, would be threatened since the environment would be very different from the ideal. More details about the validity of an experiment can be found on (Wohlin *et al.* 2012).

Concerning the case study strategy, it was concluded that this would be the ideal evaluation strategy for the presented scenario since it would be possible to evaluate the guidelines in a real-world environment. However, case studies demand time to be applied, and the available time for the guidelines to be evaluated was not long enough to enable the usage of this evaluation strategy. Moreover, it would not allow evaluating the different advices included in the guidelines, given that only some of them would be selected in a real project context. Furthermore, at least two case studies would have to be conducted to cover agile and plan-driven contexts.

The chosen evaluation strategy for the evaluation of the preliminary guidelines concerned surveying requirements engineering experts. Main reasons were allowing to obtain initial feedback on all the advice items in the pretended contexts and surpassing the difficulty of simulating a real-world environment.

This chapter will present a survey evaluation conducted to assess the preliminary guidelines for preventing the two most critical RE problems (*Communication problems between the project team and the customer* and *Incomplete and/or hidden requirements*). The updated guidelines resulting from the evaluation and the analysis of its results will also be presented.

4.2 PLANNING (SURVEY)

In order to briefly and clearly define the goal of this evaluation, the GQM (Basili *et al.* 1994) method was used and will be presented next:

“**Analyse the** use of the guidelines during the RE process development
with the purpose of characterize
with respect to the appropriateness of the guideline advices for preventing the most critical RE problems
from the point of view of RE experts
in the context of a real-world organisation facing the most critical RE problems during the development of its projects”.

Based on this goal, the idea of the survey was to present all the proposed guideline advices to help avoiding the most critical RE problems for each type of organisation (agile and plan-driven) to experts, so that they could judge their appropriateness based on their personal knowledge and experience. Thus, each advice included in the guidelines would have a likert scale (options: *disagree, partially disagree, partially agree, disagree* and *not sure*) and the respondents would be able to select the option that best suited their opinion on how much they agreed that the referred advice would help avoiding a particular RE problem. It is noteworthy that, to force participants to position themselves when judging the advices, the option *neutral* was not included. Additionally, they would also be able to suggest new advices to help avoiding the RE problem.

While the overall goal concerns all the RE problems, there would be to many guideline advices to be evaluated by the experts and answering the survey would require significant effort. Indeed, all the guideline advices proposed for each process model of each critical RE problem would have to be analysed. Therefore, during the survey planning, a discussion was conducted and the decision taken was to apply the survey, to evaluate only the two most critical RE problems (*Communication flaws between the project team and the customer* and *Incomplete and/or hidden requirements*). The resulting planned survey can be seen in APPENDIX E.

Concerning the survey population, we needed RE experts that would be able to judge the guideline advice items. Several RE and empirical software engineering experts with large experience supported the NaPiRE initiative. They were chosen as the survey population as we believed that they had appropriate experience related to the topic and would be able to answer the survey in an unbiased way judging the options according to what they, based on their backgrounds, believed would be best to prevent the problems. We limited the population to the experts involved in the 2014-2015 NaPiRE trial.

4.3 OPERATION

For operation purposes the survey, shown on APPENDIX E, was created on Google Forms and the link shared by e-mail to 26 experts involved in the 2014-2015 NaPiRE trial. The deadline for answering the survey was of two weeks. After an additional two week extension data collection was considered concluded.

In total, 19 of the involved experts answered the survey registering their opinion on the compiled advice items. Namely:

- From Austria: Dietmar Winkler (Vienna University of Technology), Michael Felderer (University of Innsbruck) and Stefan Biffl (Vienna University of Technology);
- From Brazil: Marcos Kalinowski (Fluminense Federal University), Rodrigo Spínola (UNIFACS/Fraunhofer Project Center) and Tayana Conte (Federal University of Amazonas);
- From Canada: Günther Ruhe (University of Calgary) and Maleknaz Nayebi (University of Calgary);
- From Estonia: Dietmar Pfahl (University of Tartu);

- From Germany: André Schekelmann (University Hochschule Niederrhein), Daniel Méndez Fernández (Technical University of Munich) and Stefan Wagner (University of Stuttgart);
- From Italy: Antonio Vetrò (Politecnico di Torino);
- From Netherlands: Roel Wieringa (University of Twente);
- From Spain: Jose Luis de la Vara (Carlos III University of Madrid);
- From Sweden: Marie-Therese Christiansson (Karlstads Universitet);
- From United Kingdom: Desmond Greer (Queen's University Belfast);
- From United States: Birgit Penzenstadler (California State University) and Jeffrey C. Carver (University of Alabama).

It is noteworthy that all participants are active in relevant software engineering and/or requirements engineering program committees and journal editorial boards.

4.4 RESULTS

In the next subsections the results of the evaluation survey for the two most critical RE problems (*Communication flaws between the project team and the customer* and *Incomplete and/or hidden requirements)* for each process model will be presented.

4.4.1 COMMUNICATION FLAWS IN AGILE ORGANISATIONS

The next figures show the answers given by the 19 respondents on how much do they agree that a specific guideline advice can help avoiding a critical RE problem. Figure 4.1, Figure 4.2 and Figure 4.3 present the answers given by the experts for the *Communication flaws between the project team and the customer* problem for Agile organisations.

The vertical axe shows the number of experts that voted whether they agree that each guideline (shown on the horizontal axe) can help avoiding a specific critical RE problem.

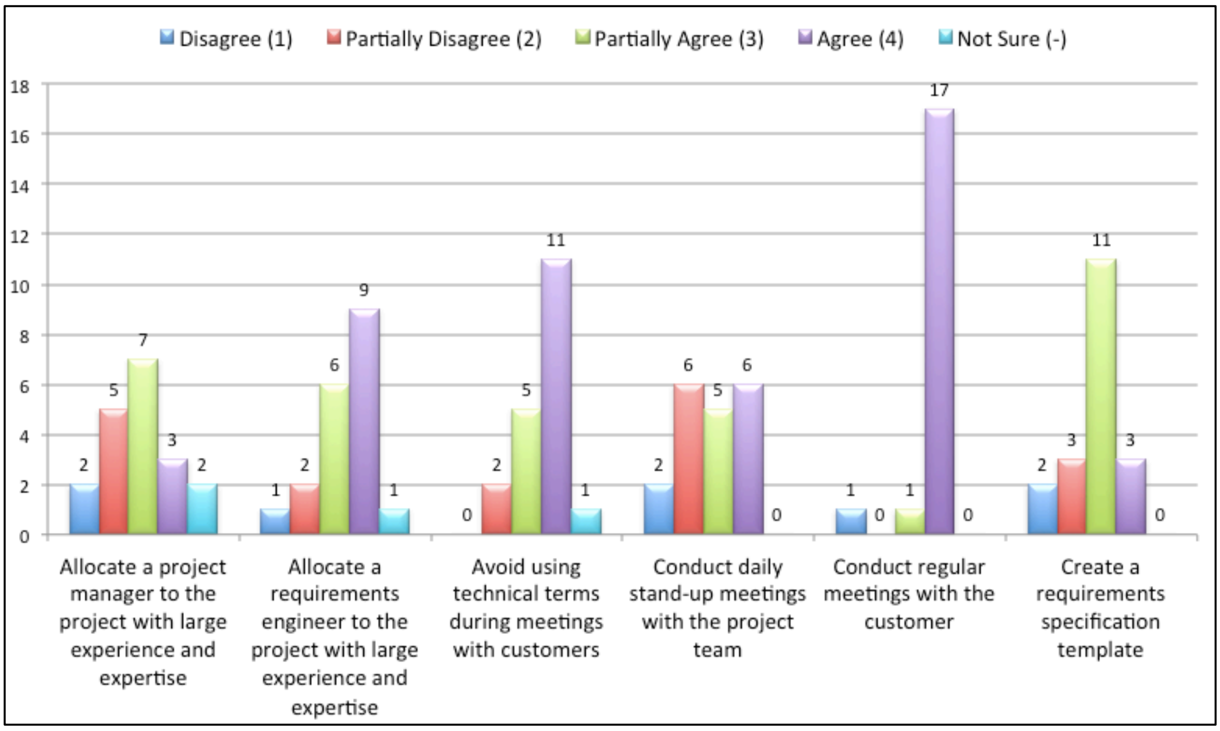


Figure 4.1. Results for *Communication flaws between the project team and the customer* for Agile Organisations (Part 1)

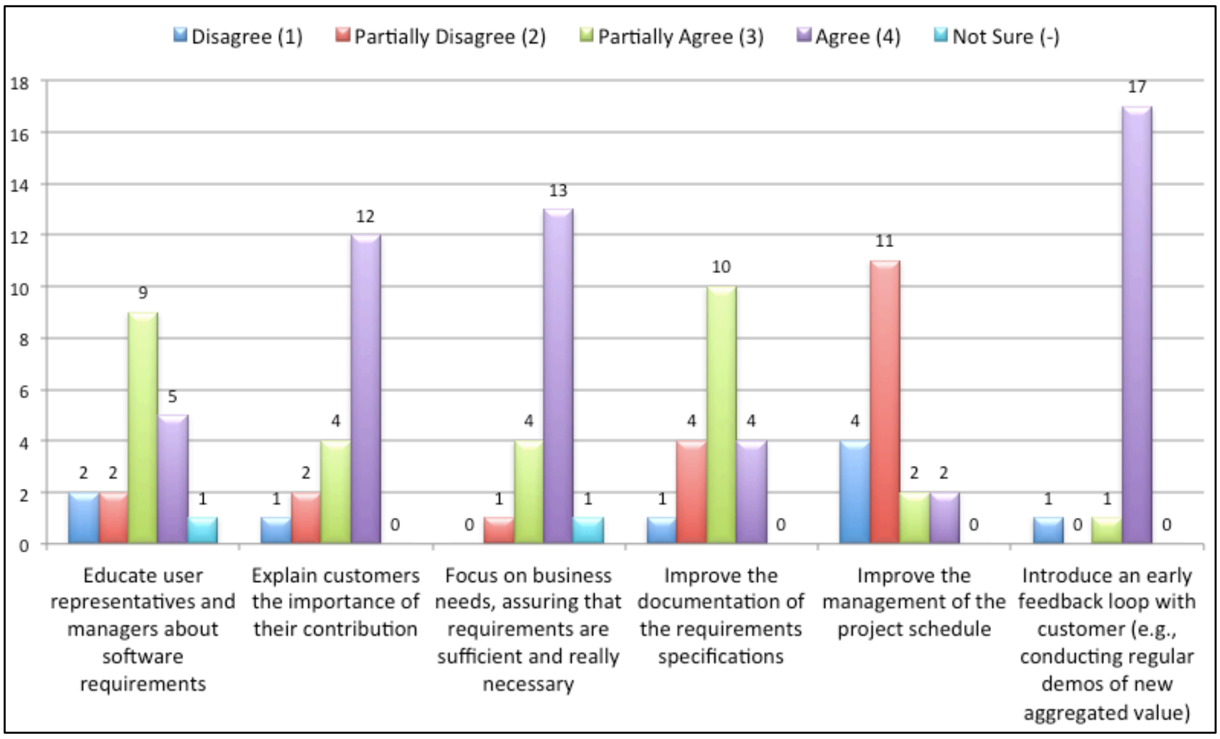


Figure 4.2. Results for *Communication flaws between the project team and the customer* for Agile Organisations (Part 2)

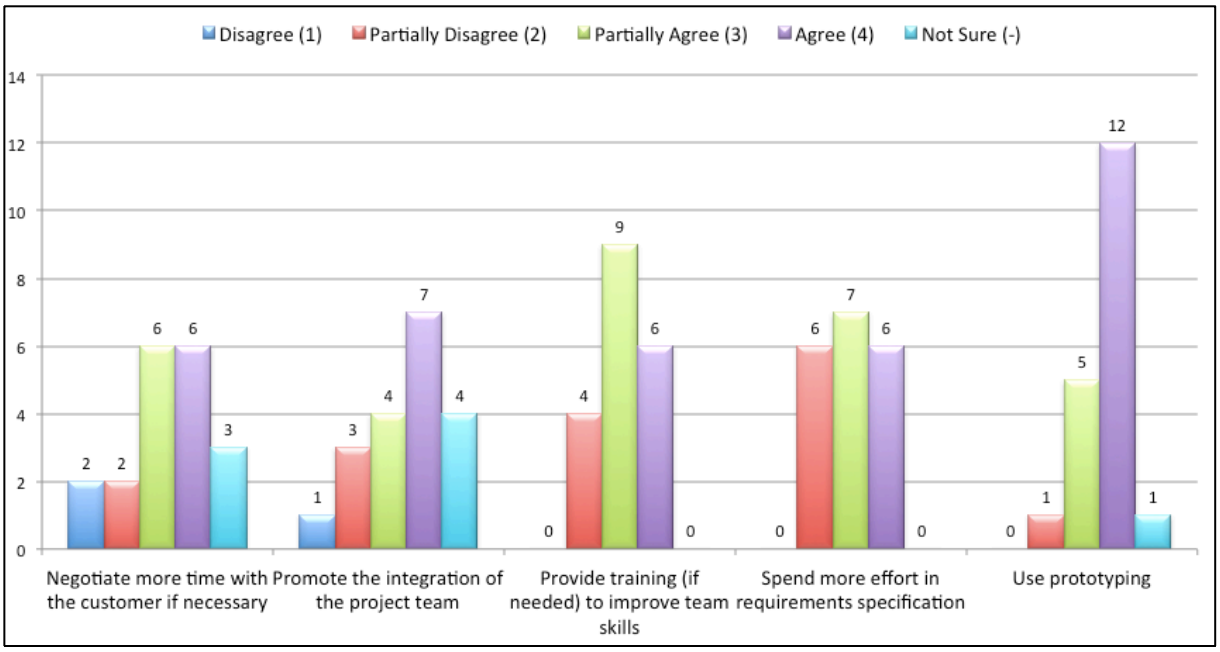


Figure 4.3. Results for *Communication flaws between the project team and the customer* for Agile Organisations (Part 3)

The figures show that some advices had a great acceptance by the experts. Most of the respondents of the evaluation survey agreed that the advices "Conduct regular meetings with the customer", "Introduce an early feedback loop with customer" and "Use prototyping" can help avoiding the *Communication flaws* problem on Agile organisations.

However, it was also possible to observe that, for some advices, the opinions of the experts varied substantially as happened with the advices: "Allocate a project manager to the project with large experience and expertise", "Conduct daily stand-up meetings with the project team", "Negotiate more time with the customer if necessary", "Promote the integration of the project team" and "Spend more effort in requirements specification". For those advices a more deep analysis was necessary in order to determine if they could be considered as an accepted or rejected one by the point of view of the RE experts, what will be discussed on the subsection 4.5.

Besides answering the survey with their level of agreement with the guidelines, the respondents also suggested some advices that they think could help avoiding each problem exposed on the evaluation. The following additional advices were suggested by them to help avoiding the *Communication flaws between the project team and the customer* problem on Agile organisations:

- Allocate team members with large expertise in the line of business;

- Avoid project situation where the project language is not the mother's tongue of both communication partners;
- Increase the expertise of the team in the line of business (e.g. training);
- Invest on activities to build up or increase trust;
- Keep all communication at a single "point of truth" - it is critical if important project communication is stored in personal mail inboxes of team members;
- Mandate (contract) on-site customer;
- Use Agile Hothouse (2-3 day joint workshops where design decisions are made after some prototyping/re-prototyping);
- Use a glossary with central (domain-specific) terms.

Thus, the additional advices given by the experts are mostly focused on the project team, with advices related to a greater interaction within them in order to facilitate their communication and knowledge exchange and an investment on trainings in order to increase the team skills in the business area. Also, an interesting advice was about the interaction between the team and the customer, with a special care about the used language and an allocation of a team member on the customer environment.

4.4.2 COMMUNICATION FLAWS IN PLAN-DRIVEN ORGANISATIONS

The next figures (Figure 4.4, Figure 4.5 and Figure 4.6) show the answers given by the experts for the *Communication flaws between the project team and the customer* problem for Plan-Driven organisations.

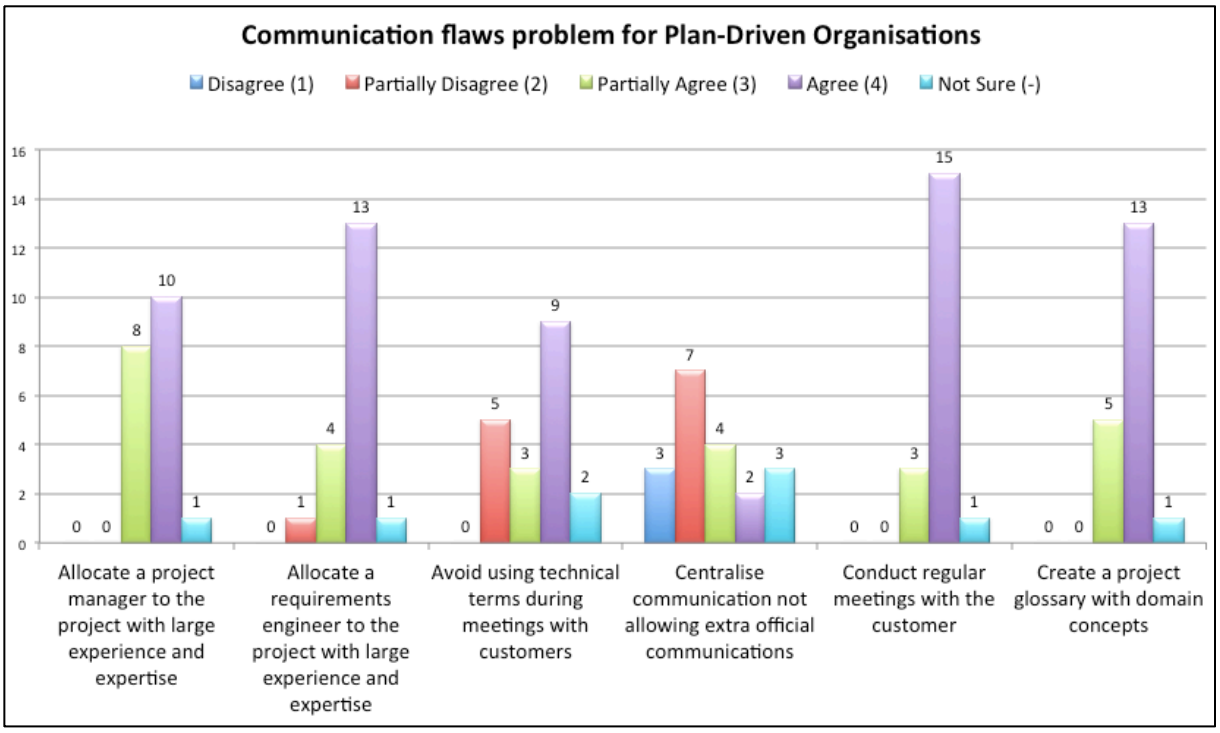


Figure 4.4. Results for *Communication flaws between the project team and the customer* problem for Plan-Driven Organisations (Part 1)

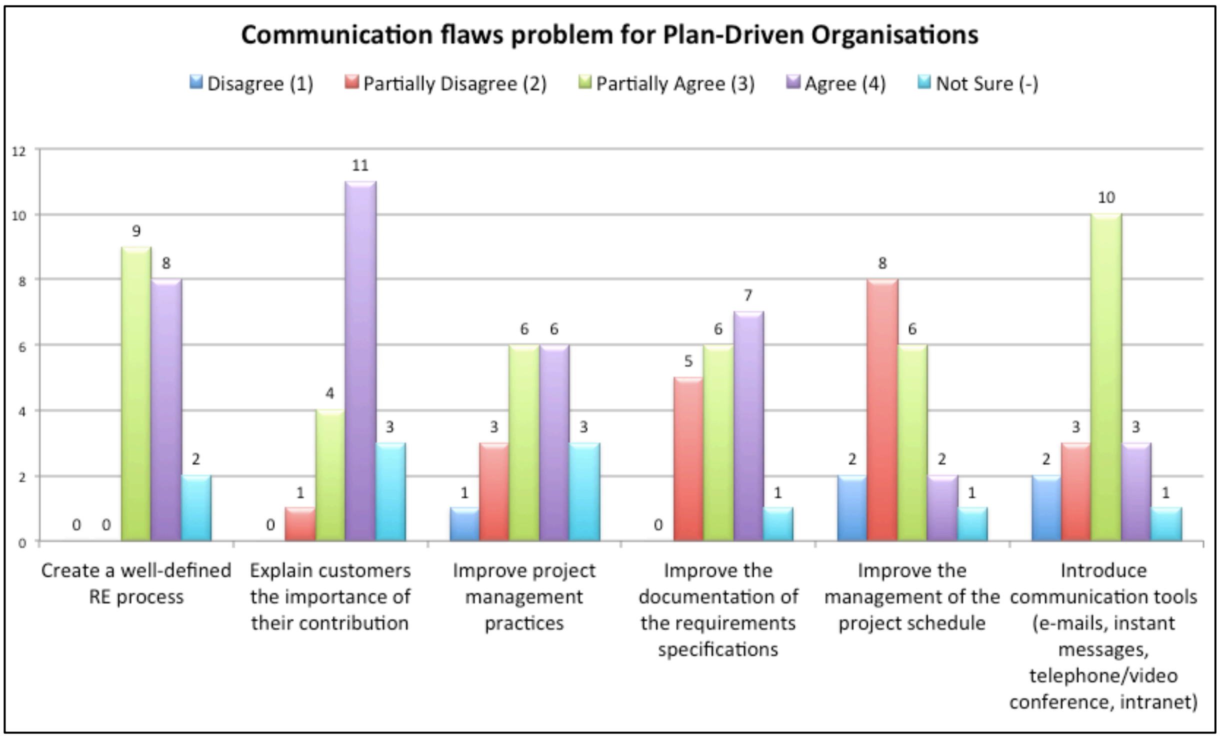


Figure 4.5. Results for *Communication flaws between the project team and the customer* problem for Plan-Driven Organisations (Part 2)

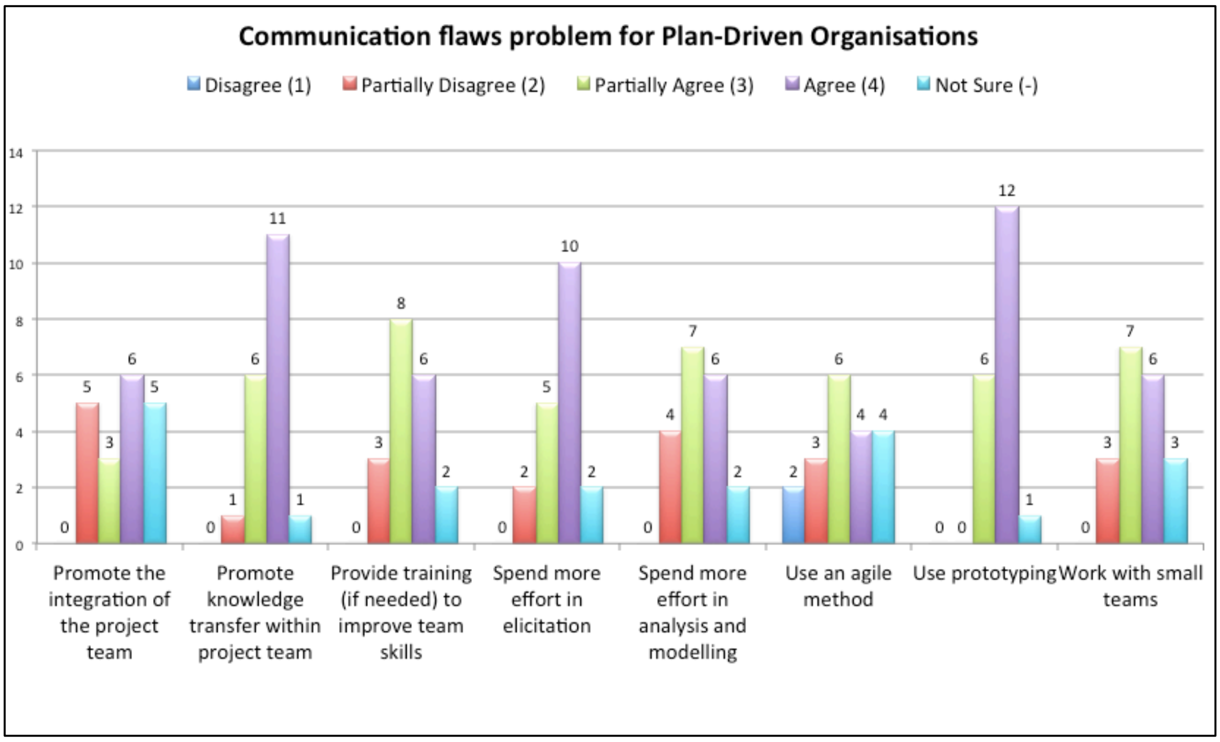


Figure 4.6. Results for *Communication flaws between the project team and the customer* problem for Plan-Driven Organisations (Part 3)

From the figures, it is possible to see that some advices had a great acceptance by the experts. Most of the respondents of the evaluation survey agreed that the advices "Allocate a project manager to the project with large experience and expertise", "Allocate a requirements engineer to the project with large experience and expertise", "Conduct regular meetings with the customer", "Create a project glossary with domain concepts", "Create a well-defined RE process", "Explain customers the importance of their contribution", "Promote knowledge transfer within project team" and "Use prototyping" can help avoiding the *Communication flaws* problem on plan-driven organisations.

However, it was also possible to observe that, for some advices, the opinions of the experts varied substantially as happened with the advices: "Centralise communication not allowing extra official communications", "Improve the documentation of the requirements specifications", "Improve the management of the project schedule" and "Promote the integration of the project team", for example.

Concerning the Communication problem on Plan-Driven organisations the following advices were suggested:

- Allocate team members with large expertise in the line of business;

- Avoid project situation where the project language is not the mother's tongue of both communication partners;
- Document the oral communication;
- Increase the expertise of the team in the line of business (e.g. training);
- Invest on activities to build up or increase trust;
- Keep all communication at a single "point of truth" - it is critical if important project communication is stored in personal mail inboxes of team members only;
- Provide early solution pieces (spikes, gui prototypes) to increase the confidence of the customer in the skills of the team.

4.4.3 INCOMPLETE REQUIREMENTS IN AGILE ORGANISATIONS

The next figures (Figure 4.7, Figure 4.8, Figure 4.9 and Figure 4.10) show the answers given by the experts for the *Incomplete and/or hidden requirements* problem for Agile organisations.

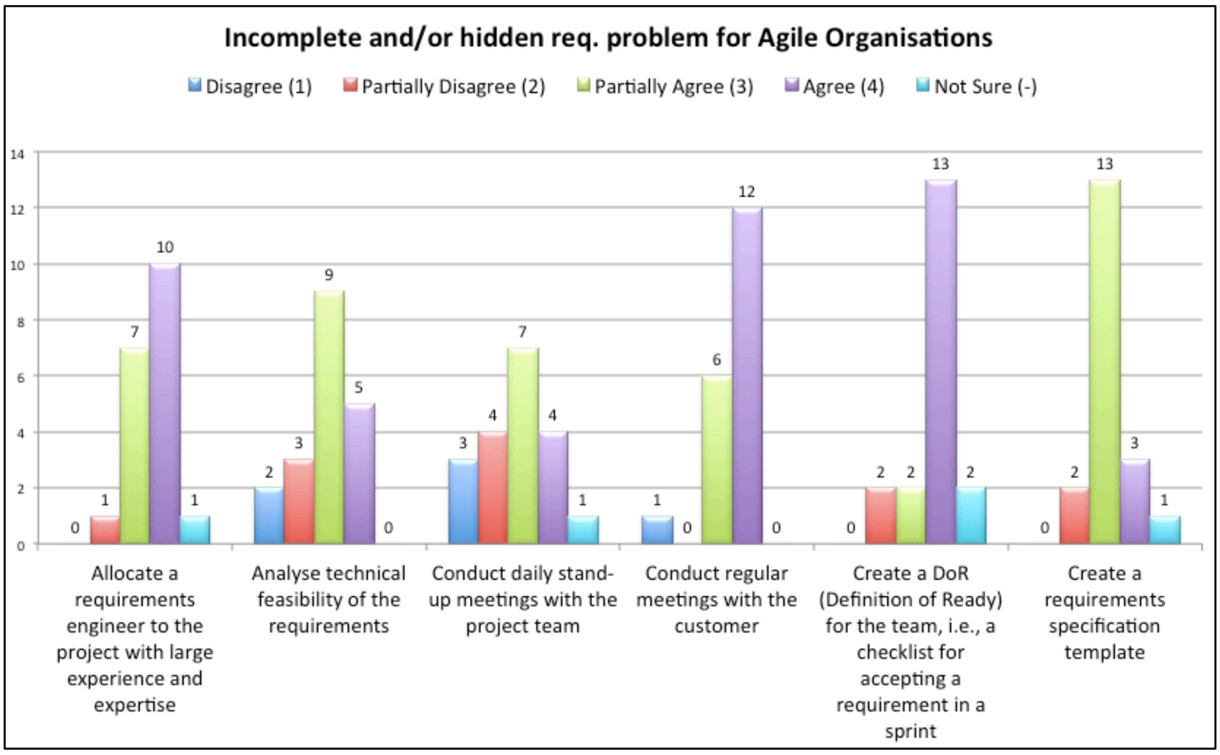


Figure 4.7. Results for *Incomplete and/or hidden requirements* problem for Agile Organisations (Part 1)

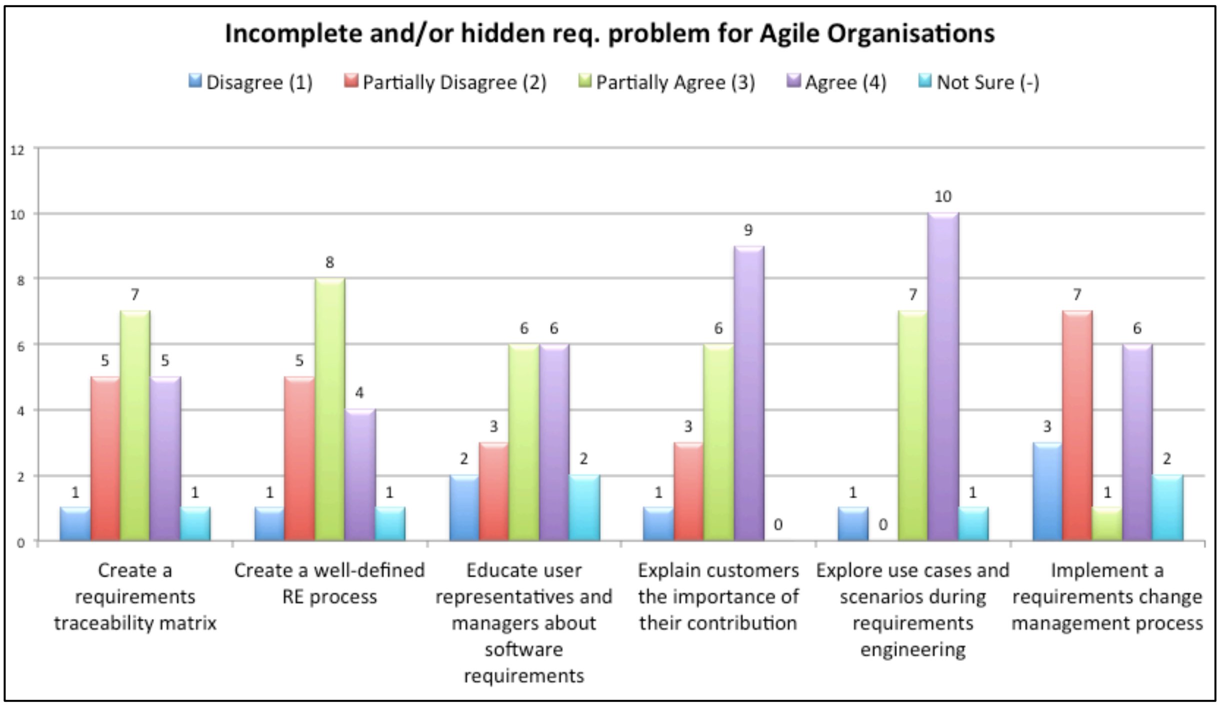


Figure 4.8. Results for *Incomplete and/or hidden requirements* problem for Agile Organisations (Part 2)

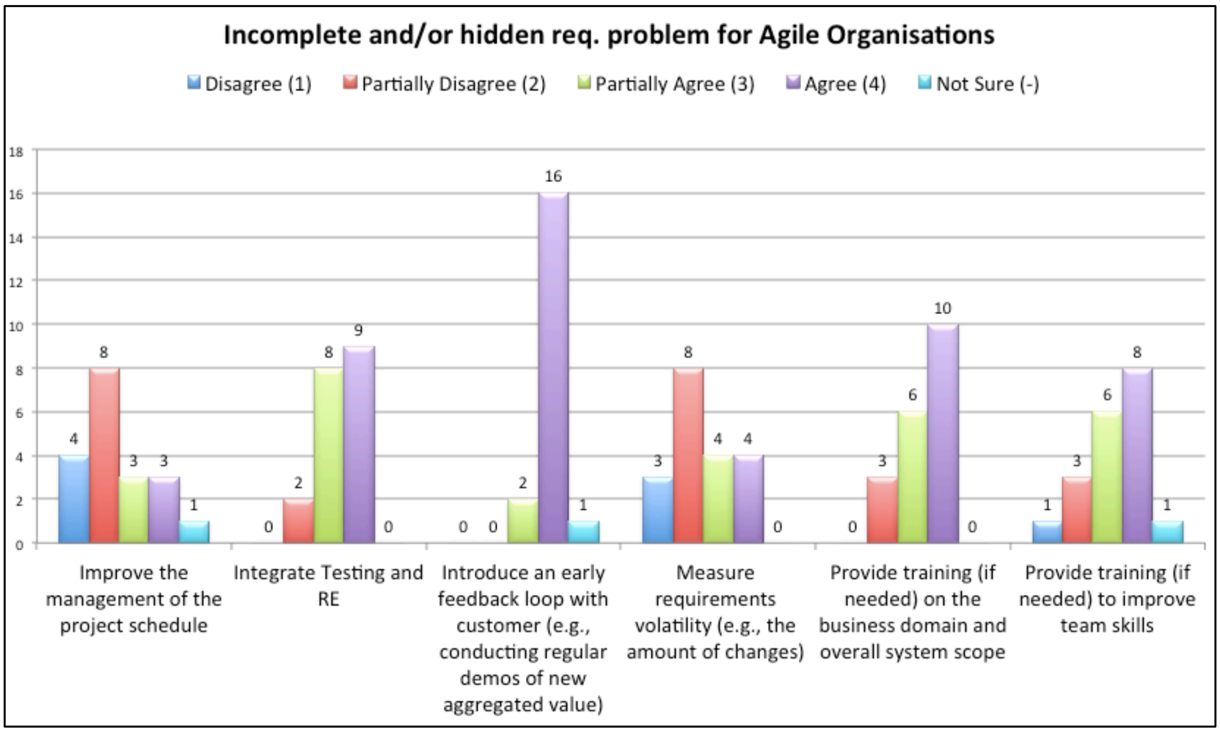
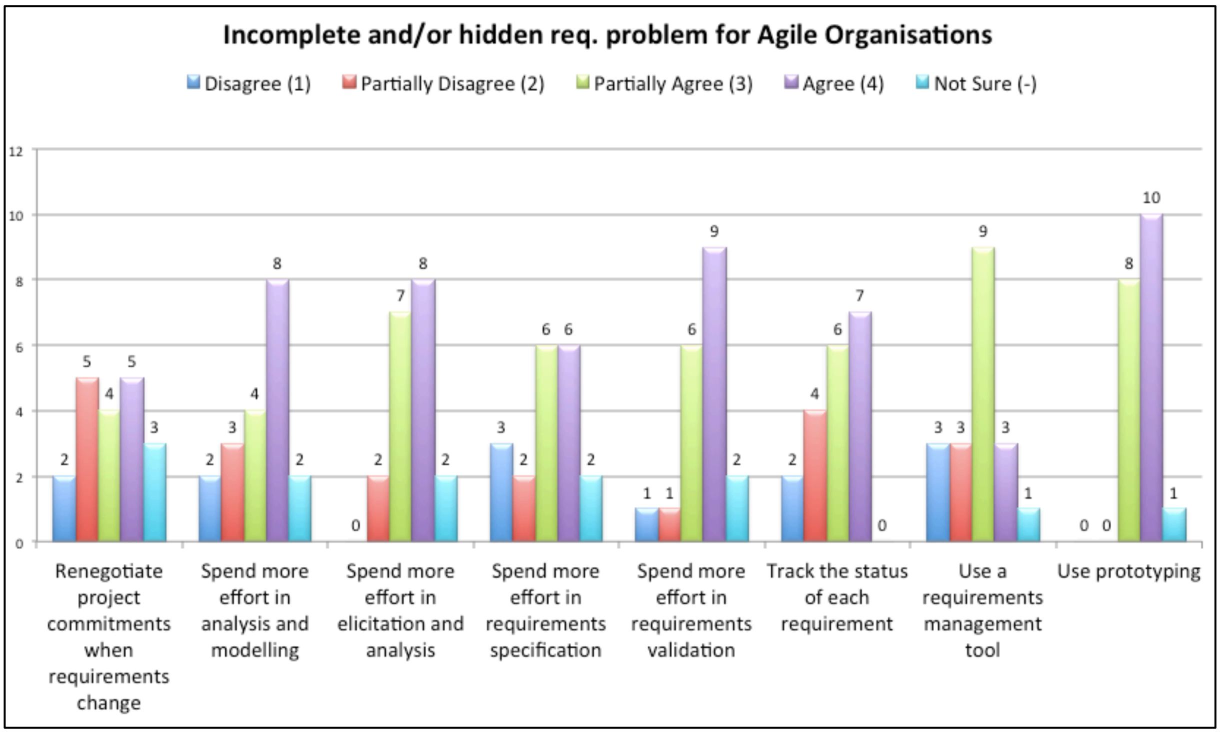


Figure 4.9. Results for *Incomplete and/or hidden requirements* problem for Agile Organisations (Part 3)



	Disagree (1)	Partially Disagree (2)	Partially Agree (3)	Agree (4)	Not Sure (-)
Renegotiate project commitments when requirements change	2	5	4	5	3
Spend more effort in analysis and modelling	2	3	4	8	2
Spend more effort in elicitation and analysis	0	2	7	8	2
Spend more effort in requirements specification	3	2	6	6	2
Spend more effort in requirements validation	1	1	6	9	2
Track the status of each requirement	2	4	6	7	0
Use a requirements management tool	3	3	9	3	1
Use prototyping	0	0	8	10	1

Figure 4.10. Results for *Incomplete and/or hidden requirements* problem for Agile Organisations (Part 4)

From the figures, it is possible to see that some advices had a great acceptance by the experts. Most of the respondents of the evaluation survey agreed that the advices “Allocate a requirements engineer to the project with large experience and expertise”, “Conduct regular meetings with the customer”, “Create a DOR (Definition of Ready) for the team, i.e., a checklist for accepting a requirement in a sprint”, “Create a requirements specification template”, “Explore use cases and scenarios during the requirements engineering”, “Integrate Testing and RE”, “Introduce an early feedback loop with the customer” and “Use prototyping” can help avoiding the *Incomplete and/or hidden requirements* problem on agile organisations.

However, it was also possible to observe that, for some advices, the respondents showed no clear consensus: “Conduct daily stand-up meetings with the project team”, “Create a requirements traceability matrix”, “Implement a requirements change management process”, “Implement the management of the project schedule”, “Measure requirements volatility” and “Renegotiate project commitments when requirements change”, for example.

Concerning the Incomplete requirements problem on Agile organisations the following advices were suggested:

- Adopt others techniques, like Personas, to help understand and empathise;

- Conduct regular (at the start of the project: daily) review session within the RE team to align the way requirements are documented;
- Have three-amigos meetings;
- Perform stakeholder analysis;
- Write a "business overview" document that summarises central business processes, solution ideas, assumptions, etc. - and keep it up to date throughout the project;
- Write a vision and scope document at the start of the product.

The three-amigos meeting is a meeting where the business analyst (BA), the developer ant the quality assurance (QA) discuss the new feature and review the specification in order to create a common understanding and shared vocabulary across them. On this meeting, the QA and the developer also identify missing requirements cases and what needs to be defined before a feature can be assigned into a sprint.

4.4.4 INCOMPLETE REQUIREMENTS IN PLAN-DRIVEN ORGANISATIONS

The next figures (Figure 4.11, Figure 4.12, Figure 4.13 and Figure 4.14) show the answers given by the experts for the *Incomplete and/or hidden requirements* problem for Plan-Driven organisations.

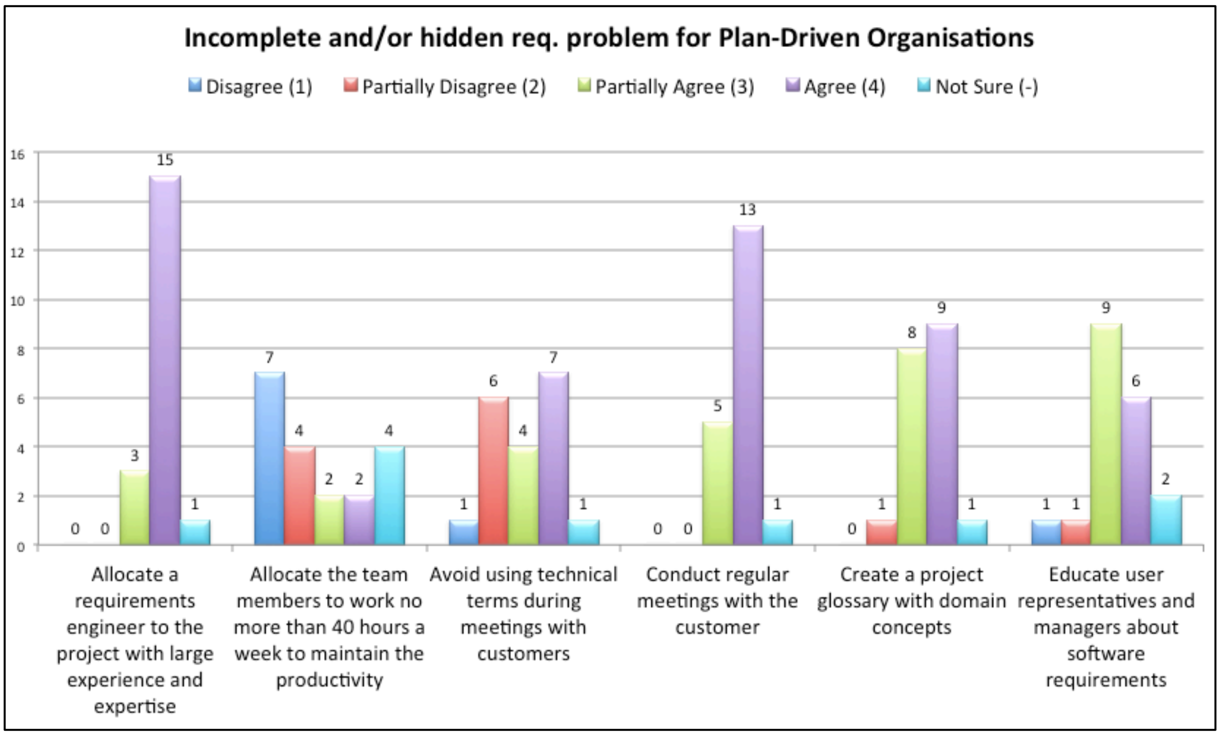


Figure 4.11. Results for *Incomplete and/or hidden requirements* problem for Plan-Driven Organisations (Part 1)

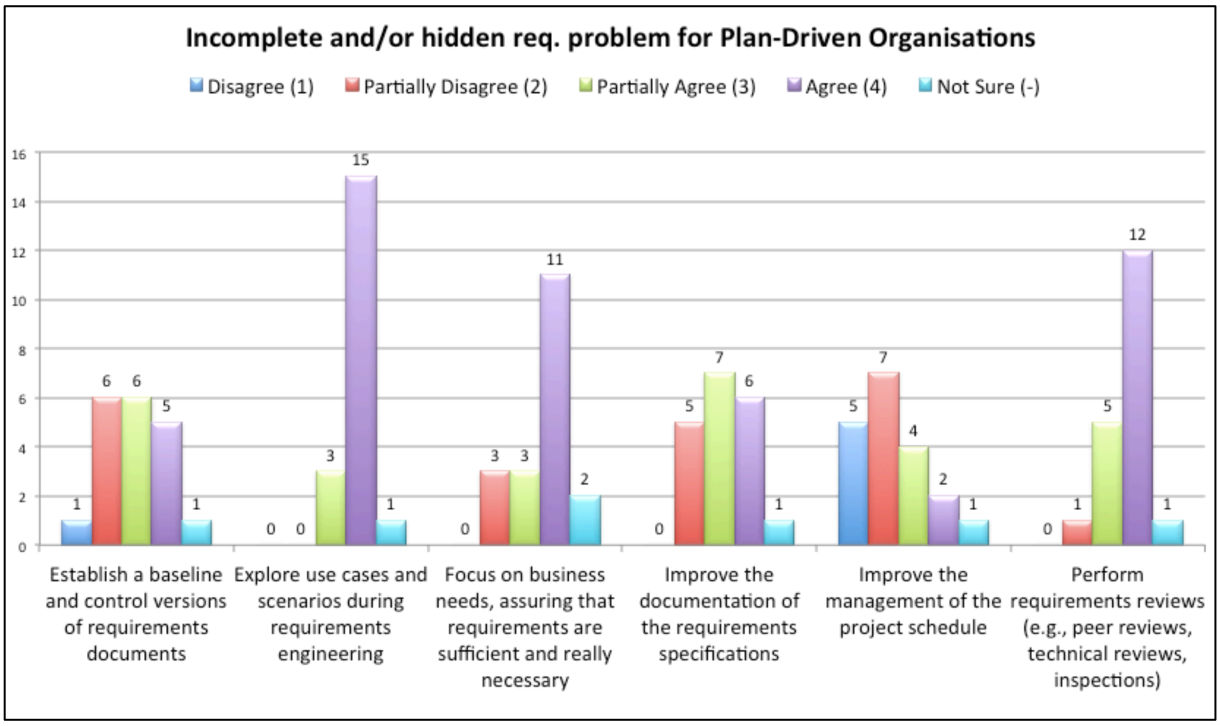


Figure 4.12. Results for *Incomplete and/or hidden requirements* problem for Plan-Driven Organisations (Part 2)

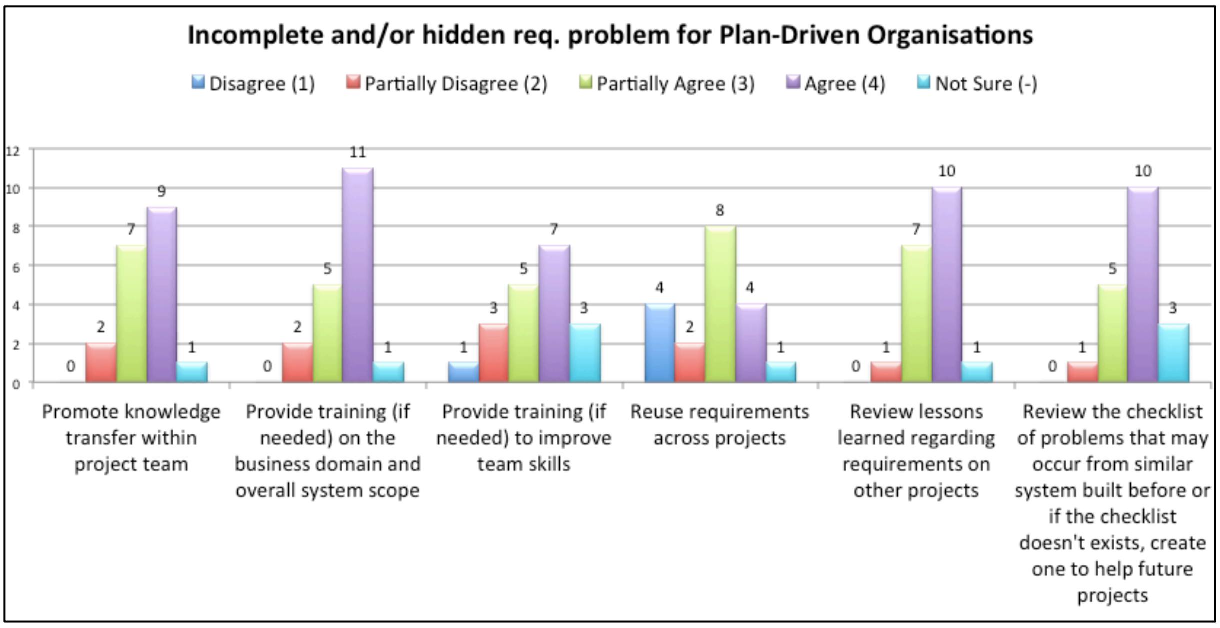


Figure 4.13. Results for *Incomplete and/or hidden requirements* problem for Plan-Driven Organisations (Part 3)

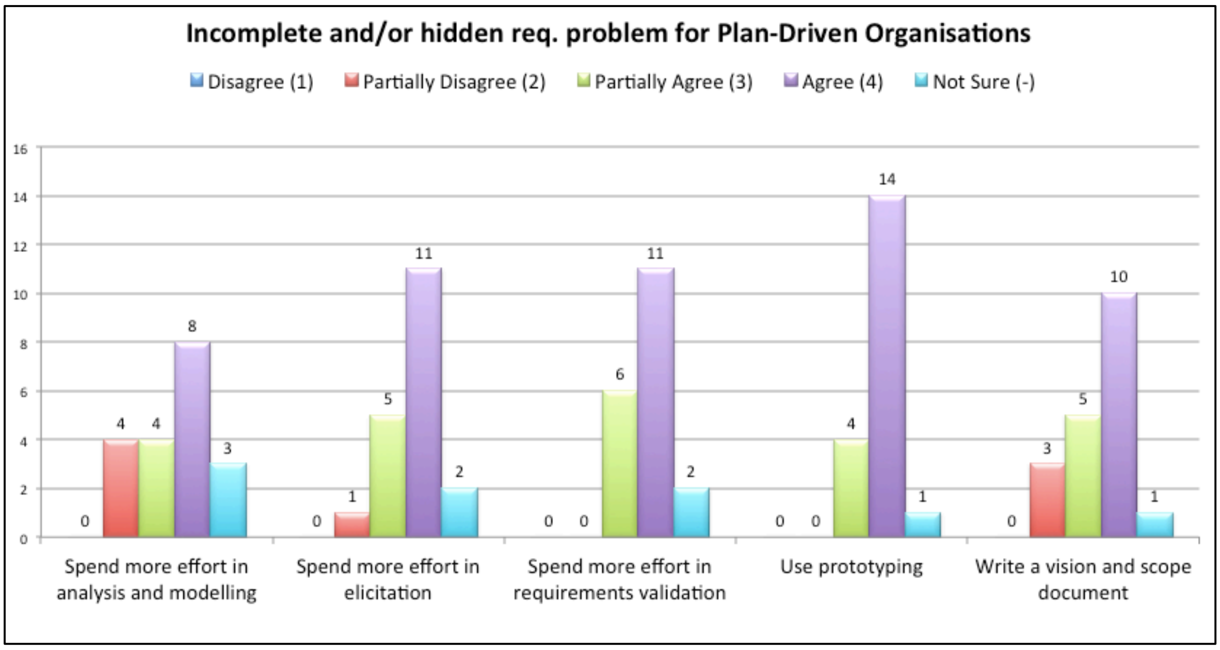


Figure 4.14. Results for *Incomplete and/or hidden requirements* problem for Plan-Driven Organisations (Part 4)

From the figures, it is possible to see that some advices had a great acceptance by the experts. Most of the respondents of the evaluation survey agreed that the advices “Allocate a requirements engineer to the project with large experience and expertise”, “Conduct regular meetings with the customer”, “Explore use cases and scenarios during the requirements engineering", "Perform requirements reviews", "Review lessons learned regarding requirements on other projects", "Spend more effort in requirements validation" and "Use prototyping" can help avoiding the *Incomplete and/or hidden requirements* problem on plan-driven organisations.

However, it was also possible to observe that, for some advices, the respondents showed no clear consensus: "Allocate the team members to work no more than 40 hours a week to maintain the productivity", "Avoid using technical terms during meetings with customers", "Establish a baseline and control versions of requirements documents", "Improve the documentation of the requirements specification" and "Improve the management of the project schedule", for example.

Regarding the *Incomplete and/or hidden requirements* problem in Plan-Driven organisations, the additional advices suggested by the respondents to help, together with the proposed guideline advices, were:

- Adopt other techniques, like Personas, to help understand and empathise;
- Conduct regular (at the start of the project: daily) review session within the RE team to align the way requirements are documented;
- Perform stakeholder analysis;
- Write a "business overview" document that summarises central business processes, solution ideas, assumptions, etc. - and keep it up to date throughout the project.

As could be seen in the Figure 4.1 to Figure 4.14 the RE experts selected, for each guideline how much do they agree that it could help avoiding a specific critical problem. However, not all the guidelines were completely accepted by the respondents as useful ones. Next, the analysis of the respondents' answers together with the final updated guidelines will be presented.

4.5 UPDATED GUIDELINES FOR PREVENTING CRITICAL RE PROBLEMS

After getting the experts responses about their level of agreement with the efficiency of the guidelines, their answers were analysed in order to update the preliminary guidelines.

In the survey, the respondents marked from 1: Disagree to 4: Agree in each guideline. With those answers at hand, in order to analyse the likert scale data the median and the MAD (median absolute deviation) were used. For each advice, the median of the answers was calculated and this information allowed to filter the guidelines maintaining in the updated ones only the advices with median equal or greater than 3 (partially agree) and discarding the other ones (disagree and partially disagree). The "Not Sure" answers were not considered in this analysis, since it does not bring substantial information to take conclusions. Along with the median, the MAD was also generated for each guideline in order to verify how much the answers varied around the median.

Table 9 and Table 10show the guidelines for preventing *Communication flaws between the project team and the customer* problem in Agile and Plan-Driven organisations, respectively, together with the calculated median and MAD values ordered from top to bottom according to the median value. Additionally, Table 11 and Table 12 show the same information referring to the *Incomplete and/or hidden requirements* problem in Agile and Plan-Driven organisations, respectively, ordered from top to bottom according to the median value.

Table 9. Survey result and guidelines analysis for preventing *Communication flaws between the project team and the customer* problem for Agile organisations

Proposed Guidelines	Median	MAD
• Allocate a requirements engineer to the project with large experience and expertise	4	1
• Avoid using technical terms during meetings with customers	4	1
• Conduct regular meetings with the customer	4	1
• Explain customers the importance of their contribution	4	1
• Focus on business needs, assuring that requirements are sufficient and really necessary	4	1
• Introduce an early feedback loop with customer (e.g., conducting regular demos of new aggregated value)	4	1
• Promote the integration of the project team	4	1
• Use prototyping	4	1
• Allocate a project manager to the project with large experience and expertise	3	2
• Conduct daily stand-up meetings with the project team	3	3
• Create a requirements specification template	3	1
• Educate user representatives and managers about software	3	2

requirements		
• Improve the documentation of the requirements specifications	3	1
• Negotiate more time with the customer if necessary	3	1
• Provide training (if needed) to improve team skills	3	1
• Spend more effort in requirements specification	3	1
• Improve the management of the project schedule	2	3

Table 10. Survey result and guidelines analysis for preventing *Communication flaws between the project team and the customer* problem for Plan-Driven organisations

Proposed Guidelines	Median	MAD
• Allocate a project manager to the project with large experience and expertise	4	1
• Allocate a requirements engineer to the project with large experience and expertise	4	1
• Avoid using technical terms during meetings with customers	4	2
• Conduct regular meetings with the customer	4	1
• Create a project glossary with domain concepts	4	1
• Create a well-defined RE process	4	1
• Explain customers the importance of their contribution	4	1
• Promote the integration of the project team	4	2
• Promote knowledge transfer within project team	4	1
• Spend more effort in elicitation	4	1
• Use prototyping	4	1
• Centralise communication not allowing extra official communications	3	1
• Improve project management practices	3	1
• Improve the documentation of the requirements specifications	3	1
• Introduce communication tools (e-mails, instant messages, telephone/video conference, intranet)	3	1
• Provide training (if needed) to improve team skills	3	1
• Spend more effort in analysis and modelling	3	1

• Use an agile method	3	1
• Work with small teams	3	1
• Improve the management of the project schedule	2	1

Table 11. Survey result and guidelines analysis for preventing *Incomplete and/or hidden requirements* problem for Agile organisations

Proposed Guidelines	Median	MAD
• Allocate a requirements engineer to the project with large experience and expertise	4	1
• Conduct regular meetings with the customer	4	1
• Create a DoR (Definition of Ready) for the team, i.e., a checklist for accepting a requirement in a sprint	4	2
• Explore use cases and scenarios during requirements engineering	4	1
• Introduce an early feedback loop with customer (e.g., conducting regular demos of new aggregated value)	4	1
• Provide training (if needed) on the business domain and overall system scope	4	1
• Spend more effort in analysis and modelling	4	1
• Spend more effort in elicitation and analysis	4	1
• Spend more effort in requirements validation	4	1
• Use prototyping	4	1
• Analyse technical feasibility of the requirements	3	1
• Conduct daily stand-up meetings with the project team	3	1
• Create a requirements specification template	3	1
• Create a requirements traceability matrix	3	1
• Create a well-defined RE process	3	1
• Educate user representatives and managers about software requirements	3	1
• Explain customers the importance of their contribution	3	1
• Integrate Testing and RE	3	1

• Provide training (if needed) to improve team skills	3	1
• Renegotiate project commitments when requirements change	3	1
• Spend more effort in requirements specification	3	1
• Track the status of each requirement	3	1
• Use a requirements management tool	3	1
• Implement a requirements change management process	2	2
• Improve the management of the project schedule	2	2
• Measure requirements volatility (e.g., the amount of changes)	2	2

Table 12. Survey result and guidelines analysis for preventing *Incomplete and/or hidden requirements* problem for Plan-Driven organisations

Proposed Guidelines	Median	MAD
• Allocate a requirements engineer to the project with large experience and expertise	4	1
• Conduct regular meetings with the customer	4	1
• Create a project glossary with domain concepts	4	1
• Explore use cases and scenarios during requirements engineering	4	1
• Focus on business needs, assuring that requirements are sufficient and really necessary	4	2
• Perform requirements reviews (e.g., peer reviews, technical reviews, inspections)	4	1
• Promote knowledge transfer within project team	4	1
• Provide training (if needed) on the business domain and overall system scope	4	1
• Provide training (if needed) to improve team skills	4	2
• Review lessons learned regarding requirements on other projects	4	1
• Review the checklist of problems that may occur from similar system built before or if the checklist doesn't exists, create one to help future projects	4	1
• Spend more effort in analysis and modelling	4	2
• Spend more effort in elicitation	4	1

• Spend more effort in requirements validation	4	1
• Use prototyping	4	1
• Write a vision and scope document	4	2
• Avoid using technical terms during meetings with customers	3	1
• Educate user representatives and managers about software requirements	3	1
• Establish a baseline and control versions of requirements documents	3	1
• Improve the documentation of the requirements specifications	3	1
• Reuse requirements across projects	3	2
• Allocate the team members to work no more than 40 hours a week to maintain the productivity	2	2
• Improve the management of the project schedule	2	1

It was possible to observe, from the Table 9, that two advices had a MAD equal to three, they were: "Conduct daily stand-up meetings with the project team" and "Improve the management of the project schedule". This means that the answers of the experts varied significantly, in other words, not all the experts agreed that those advices could really help avoiding the *Communication* problem in agile organisations. However, this disagreement did not occur on the other scenarios in such a strong way.

In order to generate the updated guidelines the preliminary ones were filtered according to the results of the survey, the advices with median less than or equal to 2 (partially disagree) were excluded from the final updated list. Also, the suggestions given by the experts on the survey as possible guidelines to prevent the critical problem at hand were included in the guidelines.

Once the advices selection criteria was based on the median, the advices that had a high value of MAD could be included on the final updated guidelines depending on their median value. The "Conduct daily stand-up meetings with the project team" advice, for instance, had a median three, so it was included on the final updated guidelines, since it attend the selection criteria.

Table 13 shows the updated guidelines for *Communication flaws between the project team and the customer* and the Table 14 shows updated guidelines for *Incomplete and/or hidden requirements* problems disposed on alphabetical order.

Table 13. Updated Guidelines for preventing *Communication flaws between the project team and the customer* problem

Communication flaws between the project team and the customer	
Agile	**Plan-Driven**
• Allocate a project manager to the project with large experience and expertise • Allocate a requirements engineer to the project with large experience and expertise • Allocate team members with large expertise in the line of business • Avoid project situation where the project language is not the mother's tongue of both communication partners • Avoid using technical terms during meetings with customers • Conduct daily stand-up meetings with the project team • Conduct regular meetings with the customer • Create a requirements specification template • Educate user representatives and managers about software requirements • Explain customers the importance of their contribution • Focus on business needs, assuring that requirements are sufficient and really necessary • Improve the documentation of the	• Allocate a project manager to the project with large experience and expertise • Allocate a requirements engineer to the project with large experience and expertise • Allocate team members with large expertise in the line of business • Avoid project situation where the project language is not the mother's tongue of both communication partners • Avoid using technical terms during meetings with customers • Conduct regular meetings with the customer • Create a project glossary with domain concepts • Create a well-defined RE process • Document the oral communication • Explain customers the importance of their contribution • Increase the expertise of the team in the line of business (e.g. training) • Keep all communication at a single "point of truth" - it is critical if

requirements specifications • Increase the expertise of the team in the line of business (e.g. training) • Introduce an early feedback loop with customer (e.g., conducting regular demos of new aggregated value) • Invest on activities to build up or increase trust • Keep all communication at a single "point of truth" - it is critical if important project communication is stored in personal mail inboxes of team members • Mandate (contract) on-site customer • Negotiate more time with the customer if necessary • Promote the integration of the project team • Provide training (if needed) to improve team skills • Spend more effort in requirements specification • Use Agile Hothouse (2-3 day joint workshops where design decisions are made after some prototyping/re-prototyping) • Use a glossary with central (domain-specific) terms • Use prototyping	important project communication is stored in personal mail inboxes of team members • Promote the integration of the project team • Promote knowledge transfer within project team • Provide early solution pieces (spikes, gui prototypes) to increase the confidence of the customer in the skills of the team • Spend more effort in elicitation • Use prototyping • Centralise communication not allowing extra official communications • Improve project management practices • Improve the documentation of the requirements specifications • Introduce communication tools (e-mails, instant messages, telephone/video conference, intranet) • Invest on activities to build up or increase trust • Provide training (if needed) to improve team skills • Spend more effort in analysis and modelling • Use an agile method • Work with small teams

Table 14. Updated Guidelines for preventing *Incomplete and/or hidden requirements* problem

Incomplete and/or hidden requirements	
Agile	**Plan-Driven**
• Adopt others techniques, like Personas, to help understand and empathise • Allocate a requirements engineer to the project with large experience and expertise • Analyse technical feasibility of the requirements • Conduct daily stand-up meetings with the project team • Conduct regular (at the start of the project: daily) review session within the RE team to align the way requirements are documented • Conduct regular meetings with the customer • Create a DoR (Definition of Ready) for the team, i.e., a checklist for accepting a requirement in a sprint • Create a requirements specification template • Create a requirements traceability matrix • Create a well-defined RE process • Educate user representatives and managers about software requirements • Explain customers the importance of their contribution • Explore use cases and scenarios during requirements engineering • Have three-amigos meetings	• Adopt others techniques, like Personas, to help understand and empathise • Allocate a requirements engineer to the project with large experience and expertise • Avoid using technical terms during meetings with customers • Conduct regular (at the start of the project: daily) review session within the RE team to align the way requirements are documented • Conduct regular meetings with the customer • Create a project glossary with domain concepts • Educate user representatives and managers about software requirements • Establish a baseline and control versions of requirements documents • Explore use cases and scenarios during requirements engineering • Focus on business needs, assuring that requirements are sufficient and really necessary • Improve the documentation of the requirements specifications • Perform requirements reviews (e.g.,

• Integrate Testing and RE • Introduce an early feedback loop with customer (e.g., conducting regular demos of new aggregated value) • Perform stakeholder analysis • Provide training (if needed) on the business domain and overall system scope • Provide training (if needed) to improve team skills • Renegotiate project commitments when requirements change • Spend more effort in analysis and modelling • Spend more effort in elicitation and analysis • Spend more effort in requirements specification • Spend more effort in requirements validation • Track the status of each requirement • Use a requirements management tool • Use prototyping • Write a "business overview" document that summarises central business processes, solution ideas, assumptions, etc. - and keep it up to date throughout the project • Write a vision and scope document at the start of the product	peer reviews, technical reviews, inspections) • Perform stakeholder analysis • Promote knowledge transfer within project team • Provide training (if needed) on the business domain and overall system scope • Provide training (if needed) to improve team skills • Reuse requirements across projects • Review lessons learned regarding requirements on other projects • Review the checklist of problems that may occur from similar system built before or if the checklist doesn't exists, create one to help future projects • Spend more effort in analysis and modelling • Spend more effort in elicitation • Spend more effort in requirements validation • Use prototyping • Write a "business overview" document that summarises central business processes, solution ideas, assumptions, etc. - and keep it up to date throughout the project • Write a vision and scope document

4.6 FINAL REMARKS

This chapter presented the survey evaluation strategy used to validate the preliminary guidelines shown on Chapter 3 and the results of its application. The survey was applied with RE experts from several countries, which informed their level of agreement with the guideline advice items on the prevention of a specific critical RE problem in both contexts (agile and plan-driven).

Finally, the chapter presented the updated guidelines generated through the analysis of the median value of the survey answers. Only the advices marked by the experts with level 3 or more (partially agree and agree) were included in the filtered list discarding the other ones. The survey results and analysis reinforce our confidence the updated guidelines as containing representative and helpful advice that can be followed by the organisations to help avoiding the most critical RE problems in practice.

CHAPTER 5 – CONCLUDING REMARKS

5.1 CONTRIBUTIONS

Requirements Engineering (RE) is an important area of software engineering. Organisations that implement effective requirements engineering processes can reap multiple benefits. A great reward comes from reducing unnecessary rework during the late development stages and throughout the commonly lengthy maintenance period (Wiegers 2003). On the other hand, RE problems can be critical, leading to severe implications, including project failure (Brooks 1987) (Méndez Fernández 2016).

Having the severity of RE problems in mind, this dissertation proposed guidelines that can be used by different types of organisations, according to their process model (agile or plan-driven), to help them preventing RE problems. To achieve this goal, information collected in a survey on RE problems was used. Data from 228 organisations of 10 different countries was analysed to propose the guidelines. Thereafter, the guidelines were evaluated and refined based on answers of a survey applied to RE experts involved on the NaPiRE initiative.

As main contributions of this dissertation we highlight the following:

- Producing the preliminary guidelines for preventing RE problems based on the NaPiRE data. This result was published at the EuroMicro Conference on Software Engineering and Advanced Applications (Mafra *et al.* 2016);
- Evaluating and refining the preliminary guidelines for the two identified most critical RE problems: *Communication flaws between the project team and the customer* and *Incomplete and/or hidden requirements*. This result is targeted at a journal publication that is currently under production.
- Supporting analyses on RE problem manifestation in the context of the NaPiRE project (Méndez Fernández *et al.* 2016).

The overall process to propose and evaluate the guidelines provides us some initial confidence that the resulting guidelines can be useful for the organisations to avoid critical RE problems in practice. The guidelines were proposed based on the causes and mitigation actions provided by the respondents of 228 organisations based on their real experiences in industry and also evaluated together with RE experts from the academy.

5.2 LIMITATIONS

On the research at hand, some limitations occurred and can be described. One of them is the threat to validity of the coding process used to analyse the answers to the open questions related to the causes and the mitigation actions provided by the respondents of the NaPiRE survey, which were used to derive the guidelines. Coding is essentially a creative task with subjective facets of coders like experience, expertise and expectations. This threat was controlled during the research by peer-reviewing the coding process.

Concerning the NaPiRE survey itself, it went through several validation cycles to reduce threats to validity (Méndez Fernández and Wagner 2015), the survey was built on the basis of a theory induced from available studies, internally and externally reviewed a few times, and piloted in an industrial context.

There are also limitations concerning the evaluation strategy. The survey was only applied to evaluate the guidelines for the two most critical RE problems. Mainly because of the size of the survey and the effort required to answer it if adding more RE problems. Also, the proposed guidelines for preventing the two most critical RE problems were only evaluated by experts from the academy using a survey, i.e., they were not evaluated (nor refined) in real industrial settings.

5.3 FUTURE WORK

Future work mainly comprises the further evaluation and refinement of the guidelines and integrating them into more holistic RE process improvement approaches. As a first step, the survey can be performed for the other four critical RE problems since, for this round of evaluation, only the “Communication flaws between the project team and the customer” and “Incomplete and/or hidden requirements” problems were evaluated.

Also, to complement the performed evaluation strategy, a case study would be a natural next step to allow capturing, in a real world environment, how much the guidelines can really help avoiding the most critical RE problems and to increase the confidence in them.

Furthermore, investigations could be performed to assess whether the guidelines should be refined using other types of context variables, besides the process model.

Finally, other options concern integrating the guidelines into RE process improvement approaches, for instance, building context-specific sets of best practices (e.g., maturity models) or integrating problem prevention into RE risk management approaches.

REFERENCES

ABRAN, A., MOORE, J.W. (2001). Guide to the Software Engineering Body of Knowledge, Trial Version. Los Alamitos, CA: *IEEE Computer Society Press.*

ADOLPH, S., HALL, W., KRUCHTEN, P. (2011). Using Grounded Theory to study the Experience of Software Development. *Empirical Software Engineering*, (Vol. 16, Issue 4).

AGUIAR, P.G., BORBA, G.L. (2013). Como especificar requisitos para desenvolvimento de software em metodologias ageis? *Centro Universitário UNA.*

AL-RAWAS, A., EASTERBROOK, S. (1996). Communication problems in requirements engineering: a field study. *National Aeronautics and Space Administration.*

ALVES, S. DE R., ALVES, A. L. (2009). Engenharia de Requisitos em Metodologias Ágeis. *Goiânia: Universidade Católica de Goiás (PUC – Goiás).*

BASILI, V.R., CALDERA, C., ROMBACH, D. (1994). Goal Question Metric Paradigm. *Encyclopedia of Software Engineering (Marciniak J. editor), John Wiley & Sons,* (Vol. 1, pp. 528-532).

BECK, K., GRENNING, J., MARTIN, R.C., BEEDLE, M., HIGHSMITH, J., MELLOR, S., VAN BENNEKUM, A., HUNT, A., SCHWABER, K., COCKBURN, A., JEFFRIES, R., SUTHERLAND, J., CUNNINGHAM, W., KERN, J., THOMAS, D., FOWLER, M., MARICK, B. (2001). Manifesto for Agile Software Development. *Agile Alliance*.

BOEHM, B., BASILI, V. R. (2001). Software Defect Reduction Top 10 List. *IEEE Computer*, vol. 34, pp. 135-137.

BOEHM, B., PAPACCIO, P. N. (1988). Understanding and Controlling Software Costs. *IEEE Transactions on Software Engineering* (Vol. 14, Issue 10, pp. 1462-1476).

BOEHM, B., TURNER, R. (2004). Balancing Agility and Discipline. *Addison-Wesley Professional*.

BROOKS, F. (1987). No Silver Bullet: Essence and Accidents of Software Engineering. *IEEE Computer* (Vol. 20, pp. 10-19).

BUSCHERMOHLE, R., EEKHOFF, H., JOSKO, B. (2006). Success – Erfolgs- und Misserfolgsfaktoren bei der Durchführung von Hard- und Softwareentwicklungsprojekten in Deutschland. ISBN-13 978-3-8142-2035-2, *BIS-Verlag der Carl von Ossietzky Universität Oldenburg*

CHAKRABORTY, A., KANTI BAOWALY, M., AREFIN, A., NEWAZ BAHAR, A. (2012). The Role of Requirements Engineering in Software Development Life Cycle. *Journal of Emerging Trends in Computing and Information Sciences (CIS Journal)* (Vol. 3 (5)).

CMMI PRODUCTION TEAM (2010). CMMI for Development, Version 1.3. Improving processes for developing better products and services. *Software Engineering Process Management Program, Software Engineering Institute.*

COHEN, D., LINDWALL, M., COSTA, P. (2003). Agile Software Development. *Fraunhofer Center Maryland.*

DAMIAN, D., CHISAN, J. (2006). An Empirical Study of the Complex Relationships between Requirements Engineering Processes and other Processes that lead to Payoffs in Productivity, Quality, and Risk Management. *IEEE Transactions on Software Engineering* (Vol. 32 (7), pp. 433-453).

EBERLEIN, A., LEITE, J.C.S.P. (2002). Agile Requirements Definition: A View from Requirements Engineering. *Proceedings of the International Workshop on Time-Constrained Requirements Engineering (TCRE'02), Germany.*

EVELEENS, J., VERHOEF, T. (2010). The Rise and Fall of the Chaos Report Figures. IEEE Software (Vol. 27, pp. 30-36).

GRADY, R. B. (1999). An Economic Release Decision Model: Insights into Software Project Management. *In: Proceedings of the Applications of Software Measurement Conference, Orange Park, FL: Software Quality Engineering* (pp. 227-239).

HALL, T., BEECHAM, S., RAINER, A. (2003). Requirements Problems in Twelve Software Companies: an Empirical Analysis. *Empirical Software Engineering,* (Vol. 8, pp. 7-42).

HSIA, P., DAVIS, A., KUNG, D. (1993). Status report: Requirements engineering. *IEEE Software,* (Vol. 10, Issue 6, pp. 75–79).

INTERNATIONAL SCRUM INSTITUTE (2017). Scrum Revealed. 2nd Edition. *International Scrum Institute.*

ISHIKAWA, K. (1982). Guide to Quality Control. 2nd Edition. *Asian Productivity Org.*

KALINOWSKI, M., CARD, D.N., TRAVASSOS, G.H. (2012). Evidence-Based Guidelines to Defect Causal Analysis. IEEE Software (Vol. 29, Issue 4, pp. 16-18).

KALINOWSKI, M., CURTY, P., PAES, A., FERREIRA, A., SPÍNOLA, R., MÉNDEZ FERNÁNDEZ, D., FELDERER, M., WAGNER, S. (2017). Supporting Defect Causal Analysis in Practice with Cross-Company Data on Causes of Requirements Engineering Problems. *Proc. of the 39th International Conference on Software Engineering (ICSE '17, SEIP track), IEEE, Buenos Aires, Argentina.*

KALINOWSKI, M., FELDERER, M., CONTE, T., SPÍNOLA, R., PRIKLADNICKI, R., WINKLER, D., MÉNDEZ FERNÁNDEZ, D., WAGNER, S. (2016). Preventing incomplete/hidden requirements: Reflections on survey data from Austria and Brazil. *Software Quality Days (SWQD), Lecture Notes in Business Information Processing,* (Vol. 238, pp. 63-78).

KALINOWSKI, M., SPÍNOLA, R., CONTE, T. *et al.* (2017). Towards building knowledge on causes of critical requirements engineering problems. *International Conference on Software Engineering and Knowledge Engineering (SEKE), Pittsburgh, USA,* (pp. 1-6).

KALINOWSKI, M., TRAVASSOS, G.H., CARD, D.N. (2008). Towards a Defect Prevention Based Process Improvement Approach. *In: Euromicro Conference on Software Engineering and Advanced Applications (SEAA),* (pp. 199-206).

KALINOWSKI, M., MENDES, E., TRAVASSOS, G.H., (2011). Automating and Evaluating the Use of Probabilistic Cause-Effect Diagrams to Improve Defect Causal Analysis. *In: International Conference on Product Focused Software Development and Process Improvement (PROFES),* (pp. 232-246).

KHANKAEW, S., RIDDLE, S. (2014) A Review of Practice and Problems in Requirements Engineering in Small and Medium Software Enterprises in Thailand. *International Workshop on Empirical Requirements Engineering (EmpiRE)* (pp.1-8).

KOTONYA, G., SOMMERVILLE, I. (1998). Requirements Engineering: Processes and Techniques. Wiley Publishing.

LEFFINGWELL, D. (1997). Calculating the Return on Investment from More Effective Requirements Management. *American Programmer* (Vol. 10, Issue 4, pp. 13-16).

LEFFINGWELL, D. (2011). Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise. *Addison-Wesley.*

LI, L. (2016). Exact Analysis for Next Release Problem. *IEEE 24th International Requirements Engineering Conference, RE 2016, Beijing, China* (pp.438-443).

LI, L., HARMAN, M., WU, F., ZHANG, Y. (2016). The Value of Exact Analysis in Requirements Selection. *IEEE Transactions on Software Engineering* (Vol. PP (99)).

LIMA, V.C.M., SECA NETO, A.G.S., EMER, M.C.F.P. (2014). Investigação Experimental e Práticas Ágeis: Ameaças à Validade de Experimentos Envolvendo a Prática Ágil Programação em Par. *Revista Eletrônica de Sistemas de Informação* (Vol. 13 (1), pp.1-16).

LIU, L., LI, T., PENG, F. (2010). Why requirements engineering fails: A survey report from china. *International Conference on Requirements Engineering (RE)* (pp. 317-322).

MAFRA, P., KALINOWSKI, M., MÉNDEZ FERNÁNDEZ, D., FELDERER, M., WAGNER, S. (2016). Towards Guidelines for Preventing Critical Requirements Engineering Problems. *In: Proc. of the 42nd EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA '16), IEEE, Limassol, Cyprus.*

MÉNDEZ FERNÁNDEZ, D., WAGNER, S. (2015). Naming the Pain in Requirements Engineering: A Design for a Global Family of Surveys and First Results from Germany. *Information and Software Technology* (Vol. 57, pp. 616-643).

MÉNDEZ FERNÁNDEZ, D., WAGNER, S., KALINOWSKI, M., SCHEKELMANN, A., TUZCU, A., CONTE, T., SPÍNOLA, R., PRIKLADNICKI, R. (2015). Naming the Pain in Requirements Engineering: Comparing Practices in Brazil and Germany. *IEEE Software*, (Vol. 32 (5), pp. 16-23).

MÉNDEZ FERNÁNDEZ, D., WAGNER, S., KALINOWSKI, M., FELDERER, M., MAFRA, P., VETRÒ, A., CONTE, T., CHRISTIANSSON, M.-T., GREER, D., LASSENIUS, C., MANNISTO, T., NAYABI, M., OIVO, M., PENZENSTADLER, B., PFAHL, D., PRIKLADNICKI, R., RUHE, G., SCHEKELMANN, A., SEN, S., SPÍNOLA, R., TUZCU, A., DE LA VARA, J. L., WIERINGA, R. (2016). Naming the Pain in Requirements Engineering – Contemporary Problems, Causes and Effects in Practice. *Empirical Software Engineering* (doi:10.1007/s10664-016-9451-7).

MÉNDEZ FERNÁNDEZ, D., WAGNER, S., LOCHMANN, K., BAUMANN, A., DE CARNE, H. (2012). Field Study on Requirements Engineering: Investigation of Artefacts, Project Parameters, and Execution Strategies. *Information and Software Technology* (Vol. 54, pp. 162-178).

MPS.BR PRODUCTION TEAM (2012). MPS.BR – Melhoria de Processo do Software Brasileiro. Guia Geral do MPS de Software. *SOFTEX*

NAEEM, M.A., WAHEED, E.U., ALI RAZA, S.F. (2016). Requirement Correctness Problems and Strategies for Web Applications. *Pakistan Journal of Engineering, Technology and Science, PJETS* (Vol. 6 (1)).

NICOLÁS, J., TOVAL, A. (2009). On the generation of requirements specifications from software engineering models: A systematic literature review. *Information and Software Technology* (Vol. 51 (9), pp. 1291-1307).

KULA, U., SAJANIEMI, J., KALVIAINEN, H. (2000). A State-of-the-practice Survey on Requirements Engineering in Small-and Medium-sized Enterprises. *Telecom Business Research Center Lappeenranta.*

NIKULA, U., SAJANIEMI, J., KALVIAINEN, H. (2000). A State-of-the-practice Survey on Requirements Engineering in Small-and Medium-sized Enterprises. *Telecom Business Research Center Lappeenranta.*

NUSEIBEH, B., EASTERBROOK, S. (2000). Requirements Engineering: A Roadmap. *In: Proceedings of the Conference on the Future of Software Engineering, ACM, New York, USA* (pp. 35-46).

OLMOS, K., RODAS, J. (2014). KMoS-RE: knowledge management on a strategy to requirements engineering. *Requirements Engineering Journal* (Vol. 19 (4), pp. 421-440).

ORAN, A.C. (2016). A Set of Artefacts and Models to Support Requirements Communication Based on Perspectives. *In: ACM SIGSOFT Software Engineering Notes, New York, USA* (Vol. 41 (6), pp. 1-5).

OSTA, H. (2013). Software Development Module Code: CST 240.

PAULK, M., *et al.* (1995). The Capability Maturity Model: Guidelines for Improving the Software Process. Reading, MA: Addison-Wesley.

PFLEEGER, S. L. (2001). Software Engineering: Theory and Practice. 2ª. *Prentice Hall, Inc. Pearson Education, Upper Saddle River, New Jersey,* 659p.

PRADO LEITE, J.C.S., FIORINI, S.T., DURÁN, A., BERNÁRDEZ, B., DÍAS, J.S., PELOZO, E.I. (2001). Requirements Processes: An Experience Report. *Workshop em Engenharia de Requisitos, Buenos Aires, Argentina,* pp.74-86.

PRIKLADNICKI, R., WILLI, R., MILANI, F. (2014). Métodos Ágeis para Desenvolvimento de Software. *Bookman.* ISBN 978-85-8260-208-9. PP (311).

RUNESON, P., HOST, M., RAINER, A., REGNELL, B. (2012). Case Study Research in Software Engineering: Guidelines and Examples. *John Wiley & Sons.*

SMITH, G., SIDKI, A. (2009). Becoming Agile: …in an imperfect world. 1st Edition. *Greenwich: Manning Publications Co.* 408p.

SOLEMON, B., SAHIBUDDIN, S., ABD GHANI, A. A. (2009). Requirements Engineering Problems and Practices in Software Companies: An Industrial Survey. *Advances in Software Engineering* (Vol. 59, pp.70- 77).

SOMMERVILLE, I., SAWYER, P. (1997). Requirements Engineering: A good practice guide. Chichester, England: John Wiley & Sons.

SOMMERVILLE, I. (2011). Software Engineering. 9th Edition. *Pearson Education.* 529p.

SOMMERVILLE, I. (2006). Software Engineering. 8th Edition. *Pearson Addison Wesley.* 568p.

SVENSSON, H. (2005). Developing Support for Agile and Plan-Driven Methods. Ph.D. Thesis from Royal Institute of Technology, Department of Computer and Systems Sciences.

TAHERI, L., CHE PA, N., ABDULLAH, R., ABDULLAH, S. (2015). A knowledge audit model to assess the knowledge in requirement elicitation process. 9th Malaysian Software Engineering Conference, MySEC, Kuala Lumpur, Malasia (pp. 106-111).

THE STANDISH GROUP (1994). The CHAOS Report. *Dennis, MA: The Standish Group.*

WAGNER, S., MÉNDEZ FERNÁNDEZ, D., FELDERER, M., KALINOWSKI, M. (2017). Requirements Engineering Practice and Problems in Agile Projects: Results from an International Survey. *Ibero-American Conference on Software Engineering (CIBSE) - Requirements Engineering Track, Buenos Aires, Argentina.*

WAHONO, R.S. (2003). Analysing requirements engineering problems. *Proceedings of the IECI Japan Workshop*, *Chofu Bunka Kaikan Tazukuri, Japan* (pp. 55-58).

WIEGERS, K. E. (2003). Software Requirements: Practical techniques for gathering and managing requirements throughout the product development cycle. 2ª. *Microsoft Press*, 516p.

WOHLIN, C., RUNESON, P., HOST, M. OHLSSON, M.C., REGNELL, B., WESSLÉN, A. (2012). Experimentation in Software Engineering. *Springer, Verlag Berlin Heidelberg,* ISBN 3642290434.

ZAVE, P., JACKSON, M. (1997). Four Dark Corners of Requirements Engineering. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, (Vol. 6 (1), pp. 1-30).

APPENDIX A – RANK OF THE TYPICAL RE PROBLEMS

Next it is shown the rank of the criticality for each of the 21 pre-compiled problems practitioners are meant to typically encounter in practice. The Table 15 shows the typical RE problems, together with the frequency in which they are meant to lead to project failure and the percentage of its citation. They are organised in alphabetical order.

Table 15. Rank of the Typical RE Problems

		Process Model	
Problems		**Agile** (92 citations)	**Plan-Driven** (46 citations)
Communication flaws between the project team and the customer	Ranking Position	2	2
	Citations	42 (46%)	19 (41%)
	Project Failure	22 (52%)	12 (63%)
Communication flaws within the project team	Ranking Position	6	5
	Citations	28 (30%)	16 (35%)
	Project Failure	12 (43%)	5 (31%)
Discrepancy between high degree of innovation and need for formal acceptance of (potentially wrong / incomplete / unknown) requirements	Ranking Position	15	13
	Citations	13 (14%)	17 (37%)
	Project Failure	5 (39%)	0
"Gold plating" (implementation of features without corresponding requirements)	Ranking Position	11	14
	Citations	16 (16%)	8 (17%)
	Project Failure	4 (25%)	0
Incomplete and / or hidden requirements	Ranking Position	1	1
	Citations	45 (49%)	24 (52%)
	Project Failure	22 (49%)	8 (33%)
Inconsistent requirements	Ranking Position	8	10
	Citations	22 (24%)	8 (17%)
	Project Failure	7 (32%)	3 (38%)
Insufficient support by customer	Ranking Position	12	8
	Citations	15 (16%)	10 (22%)
	Project Failure	10 (67%)	5 (50%)

Insufficient support by project lead	Ranking Position	21	20
	Citations	2 (2%)	3 (7%)
	Project Failure	1 (50%)	2 (67%)
Missing traceability	Ranking Position	17	9
	Citations	12 (13%)	10 (22%)
	Project Failure	1 (8%)	3 (30%)
Moving targets (changing goals, business processes and / or requirements)	Ranking Position	3	4
	Citations	30 (33%)	17 (37%)
	Project Failure	16 (53%)	6 (35%)
Stakeholders with difficulties in separating requirements from previously known solution designs	Ranking Position	7	7
	Citations	23 (25%)	11 (24%)
	Project Failure	5 (22%)	3 (28%)
Technically unfeasible requirements	Ranking Position	20	16
	Citations	7 (8%)	5 (11%)
	Project Failure	3 (43%)	0
Terminological problems	Ranking Position	16	11
	Citations	13 (14%)	8 (17%)
	Project Failure	2 (15%)	1 (13%)
Time boxing / Not enough time in general	Ranking Position	4	6
	Citations	30 (33%)	13 (28%)
	Project Failure	12 (40%)	3 (23%)
Unclear responsibilities	Ranking Position	10	18
	Citations	16 (17%)	4 (9%)
	Project Failure	7 (44%)	2 (50%)
Unclear / unmeasurable non-functional requirements	Ranking Position	19	12
	Citations	8 (9%)	8 (17%)
	Project Failure	1 (13%)	1 (13%)
Underspecified requirements that are too abstract and allow for various interpretations	Ranking Position	5	3
	Citations	30 (33%)	17 (37%)
	Project Failure	8 (27%)	9 (53%)

Volatile customer's business domain regarding, e.g., changing points of contact, business processes or requirements	Ranking Position	18	19
	Citations	11 (12%)	4 (9%)
	Project Failure	4 (36%)	1 (25%)
Weak access to customer needs and / or (internal) business information	Ranking Position	9	17
	Citations	18 (20%)	4 (9%)
	Project Failure	6 (33%)	2 (50%)
Weak knowledge of customer's application domain	Ranking Position	13	15
	Citations	14 (15%)	6 (13%)
	Project Failure	4 (29%)	4 (67%)
Weak relationship to customer	Ranking Position	14	21
	Citations	13 (14%)	0
	Project Failure	5 (39%)	0

APPENDIX B – CODES FOR CAUSES AND MITIGATION ACTIONS FOR THE MOST CRITICAL RE PROBLEMS

Next it is shown the codes for causes and mitigation actions of all the most critical RE problems. Table 16 and Table 17 show the causes and mitigation actions for agile and plan-driven organisations respectively. The coded causes and mitigation actions are shown in alphabetic order, together with the number of times each code was cited by the respondents of the survey.

Table 16. Causes and Mitigation Actions for Agile organisations on the most critical RE problems

	Agile	
Problems	**Causes**	**Mitigation Actions**
Communication flaws between the project team and the customer	• Communication flaws between team and customer (2) • Complexity of domain (1) • Complexity of RE (1) • Conflicting stakeholder viewpoints (1) • Insufficient agility (1) • Insufficient resources (1) • Lack of experience of RE team members (2) • Lack of time (1) • Language barriers (3) • Missing completeness check of requirements (2) • Missing customer involvement (1) • Missing direct communication to customer (4) • Missing engagement by customer (4)	• Conduct daily meetings (1) • Create a requirements specification template (1) • Define a common structure to describe and explain requirements (1) • Improve customer commitment (5) • Improve Requirements Specification (1) • Increase the communication with customer (3) • Introduce an agile methodology (1) • Introduce an early feedback loop with customer (4) • Negotiate more time with the customer if necessary (1) • Plan and execute regular communication events/ meetings

	• Missing IT project experience at customer side (1) • Missing RE awareness at customer side (1) • Missing requirements specification template (1) • Strict time schedule by customer (1) • Stakeholders lack business vision and understanding (1) • Subjective interpretations (1) • Too high team distribution (3) • Weak management at customer side (1)	(6) • Plan and execute trainings (in order to improve skill and performance) (2) • Use mock-ups (1) • Use prototyping (2)
Communication flaws within the project team	• Communication flaws between team and customer (1) • Complexity of RE (1) • Conflicting stakeholder viewpoints (1) • Insufficient agility (1) • Insufficient collaboration in process (2) • Lack of a well-defined RE process (1) • Lack of experience of RE team members (1) • Lack of time (2) • Missing company wide standard (1) • Missing direct communication to customer (1) • Requirements remain too	• Conduct daily meetings (1) • Create a documentation of models and solutions (1) • Create a team with people that has great empathy with each other (1) • Define a common structure to describe and explain requirements (1) • Evaluate and optimise business processes (1) • Improve time management / planning (1) • Increase the communication with customer (1) • Introduce a leader / manager for the delivery team (1) • Introduce an early feedback loop

	abstract (1) • Too high team distribution (1) • Weak qualification of RE team members (1)	with customer (1) • Introduce a leader / manager for the delivery team (1) • Introduce communications tools (1) • Plan and execute regular communication events/ meetings (1) • Plan and execute trainings (in order to improve skill and performance) (1) • Prioritise activities / goals (1) • Promote a knowledge transfer within the project team (1)
Incomplete and/or hidden requirements	• Communication flaws between team and customer (2) • Complexity of RE (1) • Customer does not know what he wants (1) • Insufficient agility (2) • Insufficient analysis at the beginning of the project (1) • Insufficient information (1) • Insufficient resources (1) • Lack of a well-defined RE process (2) • Lack of experience of RE team members (3) • Missing company wide standard (1) • Missing completeness check of requirements (1)	• Acquire (external) requirements experts (2) • Create a Definition of Ready to the team (1) • Create a Process Model (1) • Create a requirements specification template (3) • Explore use cases and scenarios during the RE (2) • Improve customer commitment (1) • Implement a change management process (1) • Increase requirements analysis (1) • Increase the communication with customer (1) • Integrate Testing and RE (1)

	• Missing domain knowledge (2) • Missing IT project experience at customer side (1) • Missing knowledge about development framework (1) • Missing RE awareness at customer side (1) • Missing requirements specification template (1) • Missing of a global view of the system (1) • Poor requirements elicitation techniques (1) • Requirements remain too abstract (1) • Stakeholders lack business vision and understanding (1) • Subjective interpretations (1) • Unavailability of requirements engineer (1) • Unclear business needs (1) • Unclear project scope (1) • Unclear roles and responsibilities at customer side (2) • Volatile requirements (1) • Weak qualification of RE team members (3) • Weak qualification of stakeholders (1)	• Introduce and use checklists for monitoring requirements along their life cycles (1) • Introduce an early feedback loop with customer (3) • Spend more time in analysis before commitment (1) • Not to chose incomplete requirements to implement (1) • Plan and execute regular communication events/ meetings (1) • Plan and execute trainings (in order to improve skill and performance) (2) • Use prototyping (3) • Spend more time on requirements specifications (1) • Spend more time on requirements validation (1)
Moving targets (changing goals,	• Changing business needs (5) • Complexity of domain (1)	• Acquire (external) requirements experts (1)

business processes and/or requirements)	• Customer does not know what he wants (3) • Insufficient information (1) • Insufficient resources (1) • Missing completeness check of requirements (1) • Missing concentration on business needs (1) • Missing solution approach (1) • Poor project management (1) • Poor requirements elicitation techniques (1) • Unclear business needs (1) • Unclear project scope (1) • Volatile industry segment that leads to changes (1) • Weak management at customer side (1)	• Explain impact of changes to customers (1) • Have an agile project management (1) • Implement a change management process (1) • Increase awareness to focus on business processes (1) • Increase the support from the project management (1) • Introduce an agile methodology (1) • Introduce and use checklists for monitoring requirements along their life cycles (1) • Orient the customer (1) • Plan and execute trainings (in order to improve skill and performance) (1) • Prioritise activities/goals (1) • Re-plan (1) • Spend more time on requirements validation (1) • Use prototyping (1) • Work with open scope (1)
Time boxing / Not enough time in general	• Complexity of project (1) • Complexity of RE (1) • High workload (3) • Insufficient resources (1) • Lack of time (3) • Missing concentration on business needs (1)	• Ask customers more time (1) • Increase the time management / planning (1) • Explain impact of changes to customers (1) • Get better insight into company direction (1)

	• Missing willingness to change (1) • Policy restrictions (1) • Poor project management (2) • Pressure to not exceed primarily defined resources (1) • Solution orientation (2) • Strict time schedule by customer (4) • Unclear business needs (1) • Unexpected changes in requirements (1) • Unfeasible goals (1)	• Have an agile project management (1) • Implement a change management process (1) • Increase the communication with customer (2) • Introduce and use checklists for monitoring requirements along their life cycles (1) • Introduce an agile methodology (1) • Introduce an early feedback loop with customer (2) • Prioritise activities / goals (3) • Re-plan (1) • Work with smaller project and defined time and goal (1)
Underspecified requirements that are too abstract and allow for various interpretations	• Complexity of RE (1) • Customer does not know what he wants (1) • Insufficient information (1) • Insufficient resources (2) • Lack of experience of RE team members (2) • Lack of time (1) • Language barriers (1) • Missing company wide standard (1) • Missing knowledge about development framework (1) • Missing RE awareness at customer side (1)	• Acquire (external) requirements experts (1) • Create a documentation of models and solutions (2) • Create a requirements specification template (4) • Define a common structure to describe and explain requirements (1) • Detail the most of the information passed by the customer (1) • Explore use cases and scenarios during the RE (1) • Implement actions of knowledge management (1)

	• Missing requirements specification template (1) • Poor requirements elicitation techniques (2) • Requirements remain too abstract (3) • Unavailability of requirements engineer (1) • Weak qualification of RE team members (1)	• Improve customer commitment (1) • Introduce a standard (1) • Redefine the model (1) • Plan and execute trainings (in order to improve skill and performance) (1) • Use prototyping (1) • Use of stronger formal reviews (1)

Table 17. Causes and Mitigation Actions for Plan-Driven organisations on the most critical RE problems

	Plan-Driven	
Problems	**Causes**	**Mitigation Actions**
Communication flaws between the project team and the customer	• Communication flaws between team and customer (4) • Conflicting stakeholder viewpoints (1) • Lack of a well-defined RE process (1) • Lack of time (1) • Language barriers (2) • Missing involvement of developers (1) • Missing tool support (1) • Not following the communication plan (1) • Poor project management (1) • Requirements remain too	• Avoid using technical terms during meetings with customers (1) • Centralise communication not allowing extra official communications (2) • Improve customer commitment (2) • Increase documentation quality (1) • Increase Requirements Elicitation methods (1) • Increase the communication with customer (2) • Introduce a leader / manager for the delivery team (1) • Introduce an agile methodology

	abstract (1) • Subjective interpretations (1) • Too high team distribution (2)	(1) • Introduce communications tools (1) • Involve the production team (1) • Plan and execute regular communication events/ meetings (3) • Plan and execute trainings (in order to improve skill and performance) (1) • Promote a knowledge transfer within the project team (1)
Communication flaws within the project team	• Communication flaws between team and customer (1) • High workload (1) • Insufficient planning of RE (1) • Lack of a well-defined RE process (1) • Lack of time (1) • Language barriers (2) • Not following the communication plan (1) • Poor project management (1) • Requirements remain too abstract (1) • Too high team distribution (2) • Weak qualification of RE team members (1)	• Conduct daily meetings (1) • Create an organisational unit for the software development professionals to keep them centralised and focused (1) • Improve customers commitment (1) • Introduce a leader / manager for the delivery team (1) • Involve the production team (1) • Motivate the project team (1) • Plan and execute regular communication events/ meetings (3) • Plan and execute trainings (in order to improve skill and performance) (2) • Promote a knowledge transfer within the project team (1)
Incomplete and/or hidden	• Complexity of project (1)	• Acquire (external) requirements experts (1)

requirements	• Conflict of interests at customer side (1) • Customer does not know what he wants (1) • High workload (1) • Insufficient planning of RE (1) • Lack of discipline (1) • Lack of experience of RE team members (2) • Lack of time (4) • Missing completeness check of requirements (1) • Requirements remain too abstract (1) • Strict time schedule by customer (1) • Thinking in legacy systems (1) • Unclear business needs (1) • Unclear roles and responsibilities at customer side (1) • Unclear terminology (1) • Unexpected changes in requirements (1) • Weak qualification of RE team members (3)	• Create a documentation of models and solutions (1) • Evaluate and introduce tools (1) • Explore use cases and scenarios during the RE (1) • Implement a pair developers review (1) • Increase documentation quality (1) • Increase Requirements Elicitation methods (1) • Introduce software inspections (1) • Perform cross checks with solution designs (1) • Plan and execute regular communication events/ meetings (2) • Plan and execute trainings (in order to improve skill and performance) (4) • Promote a knowledge transfer within project team (1) • Use prototyping (1) • Use sign-offs before implementation (1) • Use stronger formal reviews (2)
Moving targets (changing goals, business processes and/or requirements)	• Changing business needs (1) • Customer does not know what he wants (1) • Lack of change management at customer side (1)	• Conduct a prior acceptance by the customer about the change manage policy that will be used (1) • Educate the stakeholders (1)

	• Lack of discipline (1) • Missing completeness check of requirements (1) • Poor project management (2) • Requirements remain too abstract (2) • Stakeholders lack business vision and understanding (1) • Volatile industry segment that leads to changes (2) • Weak management at customer side (1)	• Explain the impact of changes to customers. (1) • Implement a change management process (2) • Implement a release process to ensure that requirements are final (1) • Increase documentation quality (1) • Increase Requirements Elicitation methods (1) • Introduce and use checklists for monitoring requirements along their life cycles (1) • Introduce and use a requirements quantification approach (1) • Plan and execute regular communication events/ meetings (1) • Use of backlogs (1) • Wait for a better understanding of the customer on his system needs (1)
Time boxing / Not enough time in general	• Communication flaws between team and customer (1) • High workload (1) • Insufficient planning of RE (1) • Insufficient resources (1) • Insufficient resource plan (2) • Lack of time (4) • Unfeasible goals (1) • Volatile industry segment that	• Implement a change management process (1) • Improve the time management / planning (1) • Introduce a standard (1) • Spend more time on requirements analysis (1)

	leads to changes (1) • Weak qualification of RE team members (1)	
Underspecified requirements that are too abstract and allow for various interpretations	• Communication flaws between team and customer (2) • Complexity of domain (1) • Complexity of RE (1) • Insufficient agility (1) • Lack of a well-defined RE process (1) • Lack of experience of RE team members (5) • Lack of time (1) • Unclear roles and responsibilities at customer side (1) • Requirements remain too abstract (1) • Weak qualification of RE team members (2)	• Acquire (external) requirements experts (2) • Create a preview business process model (1) • Evaluate and introduce tools (1) • Implement a pair developers review (1) • Implement a technical pair review (1) • Increase documentation quality (1) • Increase Requirements Elicitation methods (1) • Introduce and use checklists for monitoring requirements along their life cycles (1) • Introduce a standard (1) • Introduce software inspections (1) • Plan and execute trainings (in order to improve skill and performance) (3) • Use detailed Wireframes (1) • Use methods (1) • Spend more time on requirements analysis (1)

APPENDIX C – ADAPTED ISHIKAWA DIAGRAMS FOR THE MOST CRITICAL RE PROBLEMS

Next it is shown the adapted Ishikawa Diagrams for each cluster of the most critical RE problems for agile and plan-driven organisations. Figure 5.1 to Figure 5.6 show the adapted Ishikawa Diagrams for the most critical RE problems on Agile Organisations and Figure 5.7 to Figure 5.12 on Plan-Driven organisations. The Diagrams are shown in alphabetic order.

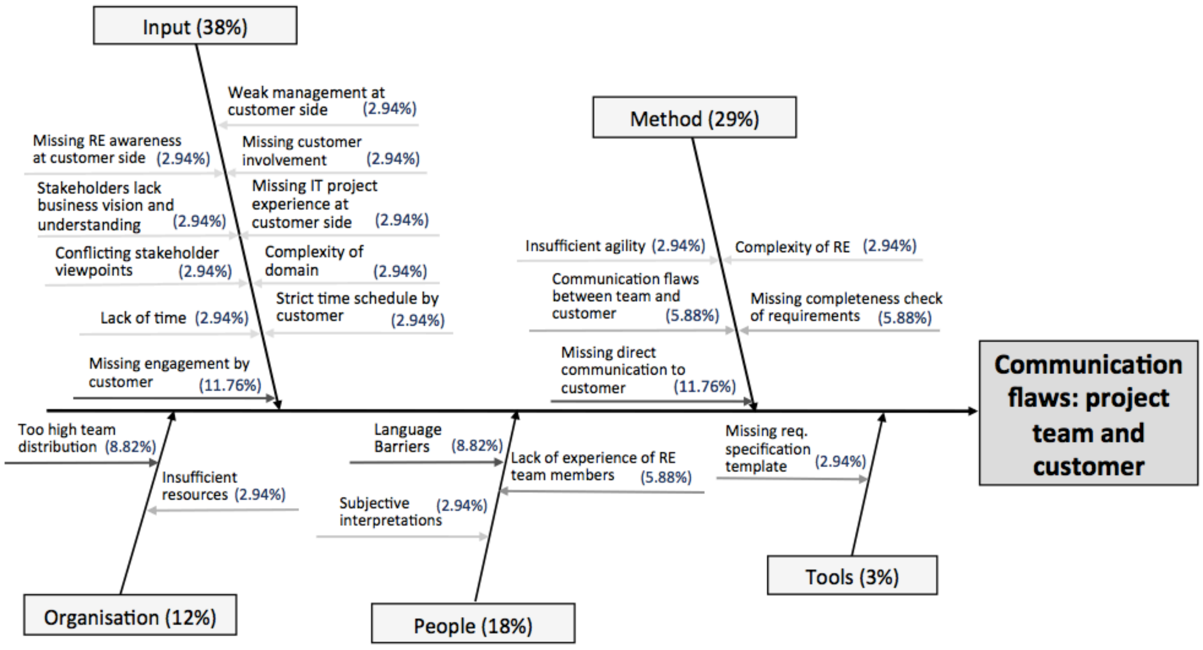
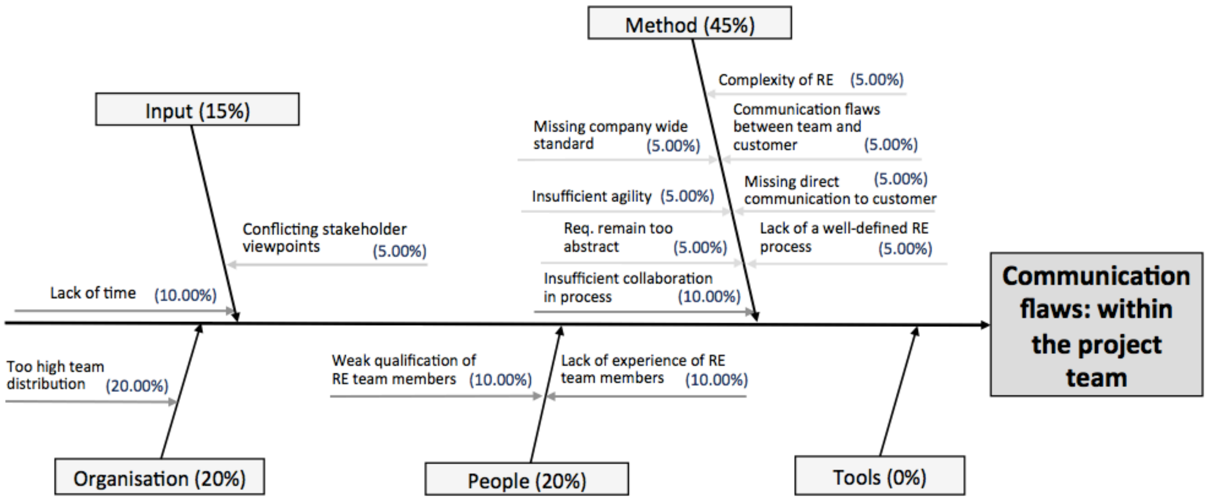
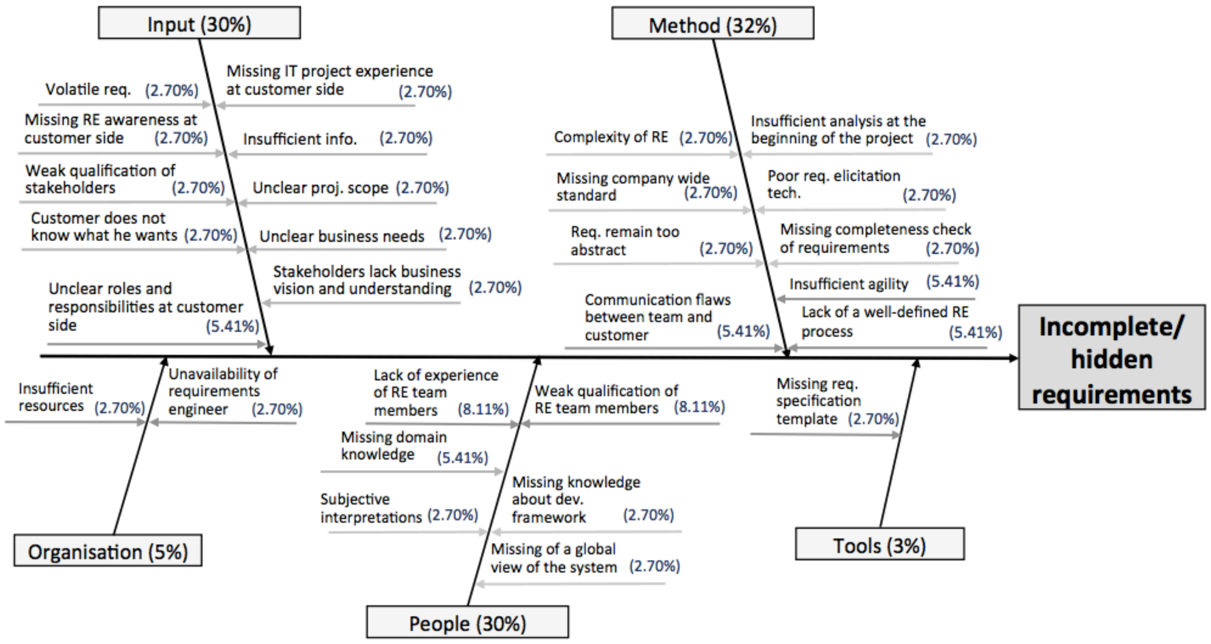

Input (38%)
Weak management at customer side (2.94%)
Missing RE awareness at customer side (2.94%)
Missing customer involvement (2.94%)
Stakeholders lack business vision and understanding (2.94%)
Missing IT project experience at customer side (2.94%)
Conflicting stakeholder viewpoints (2.94%)
Complexity of domain (2.94%)
Strict time schedule by customer (2.94%)
Lack of time (2.94%)
Missing engagement by customer (11.76%)
Method (29%)
Insufficient agility (2.94%)
Complexity of RE (2.94%)
Communication flaws between team and customer (5.88%)
Missing completeness check of requirements (5.88%)
Missing direct communication to customer (11.76%)
Communication flaws: project team and customer
Too high team distribution (8.82%)
Insufficient resources (2.94%)
Language Barriers (8.82%)
Subjective interpretations (2.94%)
Lack of experience of RE team members (5.88%)
Missing req. specification template (2.94%)
Organisation (12%)
People (18%)
Tools (3%)

Figure 5.1. Causes for *Communication flaws between the project team and the customer* problem on Agile organisations



Method (45%)
Input (15%)
Complexity of RE (5.00%)
Communication flaws between team and customer (5.00%)
Missing company wide standard (5.00%)
Missing direct communication to customer (5.00%)
Insufficient agility (5.00%)
Req. remain too abstract (5.00%)
Lack of a well-defined RE process (5.00%)
Conflicting stakeholder viewpoints (5.00%)
Insufficient collaboration in process (10.00%)
Lack of time (10.00%)
Communication flaws: within the project team
Too high team distribution (20.00%)
Weak qualification of RE team members (10.00%)
Lack of experience of RE team members (10.00%)
Organisation (20%)
People (20%)
Tools (0%)

Figure 5.2. Causes for *Communication flaws within the project team* problem on Agile organisations



Input (30%)
Method (32%)
Volatile req. (2.70%)
Missing IT project experience at customer side (2.70%)
Missing RE awareness at customer side (2.70%)
Insufficient info. (2.70%)
Weak qualification of stakeholders (2.70%)
Unclear proj. scope (2.70%)
Customer does not know what he wants (2.70%)
Unclear business needs (2.70%)
Stakeholders lack business vision and understanding (2.70%)
Unclear roles and responsibilities at customer side (5.41%)
Complexity of RE (2.70%)
Insufficient analysis at the beginning of the project (2.70%)
Missing company wide standard (2.70%)
Poor req. elicitation tech. (2.70%)
Req. remain too abstract (2.70%)
Missing completeness check of requirements (2.70%)
Insufficient agility (5.41%)
Communication flaws between team and customer (5.41%)
Lack of a well-defined RE process (5.41%)
Incomplete/ hidden requirements
Insufficient resources (2.70%)
Unavailability of requirements engineer (2.70%)
Lack of experience of RE team members (8.11%)
Weak qualification of RE team members (8.11%)
Missing req. specification template (2.70%)
Missing domain knowledge (5.41%)
Subjective interpretations (2.70%)
Missing knowledge about dev. framework (2.70%)
Missing of a global view of the system (2.70%)
Organisation (5%)
Tools (3%)
People (30%)

Figure 5.3. Causes for *Incomplete/hidden requirements* problem on Agile organisations

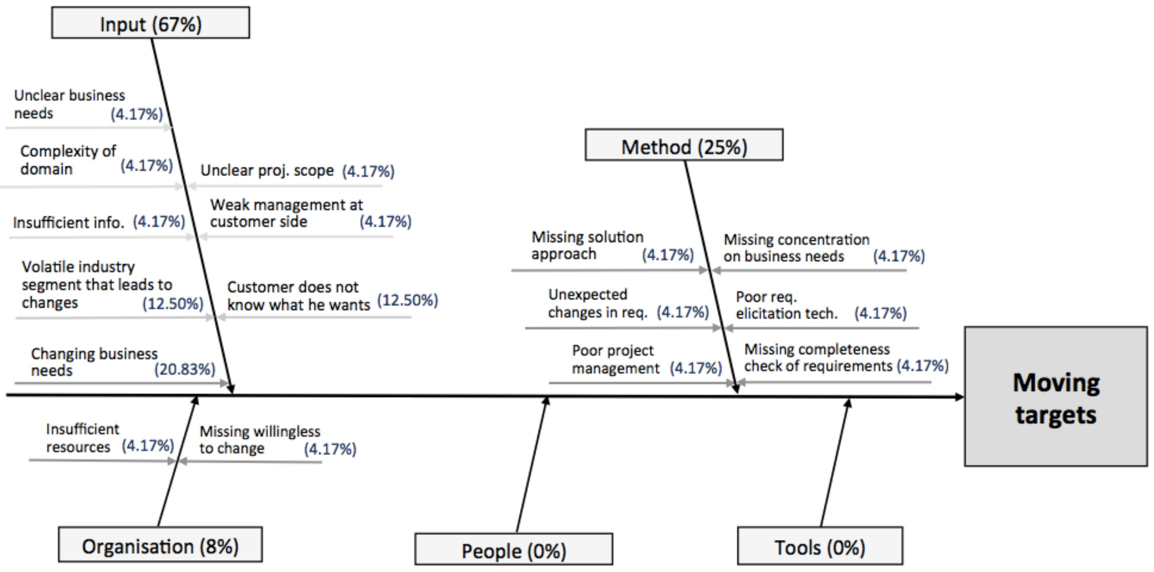

Input (67%)
Unclear business needs (4.17%)
Complexity of domain (4.17%)
Unclear proj. scope (4.17%)
Insufficient info. (4.17%)
Weak management at customer side (4.17%)
Volatile industry segment that leads to changes (12.50%)
Customer does not know what he wants (12.50%)
Changing business needs (20.83%)
Method (25%)
Missing solution approach (4.17%)
Missing concentration on business needs (4.17%)
Unexpected changes in req. (4.17%)
Poor req. elicitation tech. (4.17%)
Poor project management (4.17%)
Missing completeness check of requirements (4.17%)
Moving targets
Insufficient resources (4.17%)
Missing willingless to change (4.17%)
Organisation (8%)
People (0%)
Tools (0%)

Figure 5.4. Causes for *Moving targets* problem on Agile organisations

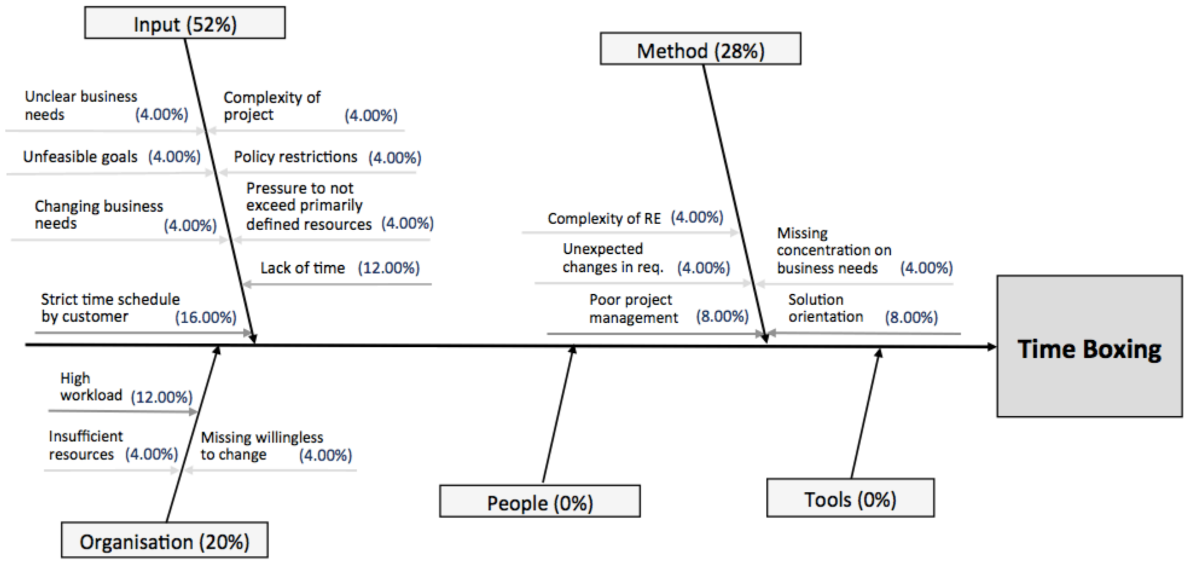

Input (52%)
Unclear business needs (4.00%)
Complexity of project (4.00%)
Unfeasible goals (4.00%)
Policy restrictions (4.00%)
Changing business needs (4.00%)
Pressure to not exceed primarily defined resources (4.00%)
Lack of time (12.00%)
Strict time schedule by customer (16.00%)
Method (28%)
Complexity of RE (4.00%)
Unexpected changes in req. (4.00%)
Missing concentration on business needs (4.00%)
Poor project management (8.00%)
Solution orientation (8.00%)
Time Boxing
High workload (12.00%)
Insufficient resources (4.00%)
Missing willingless to change (4.00%)
Organisation (20%)
People (0%)
Tools (0%)

Figure 5.5. Causes for *Time Boxing* problem on Agile organisations

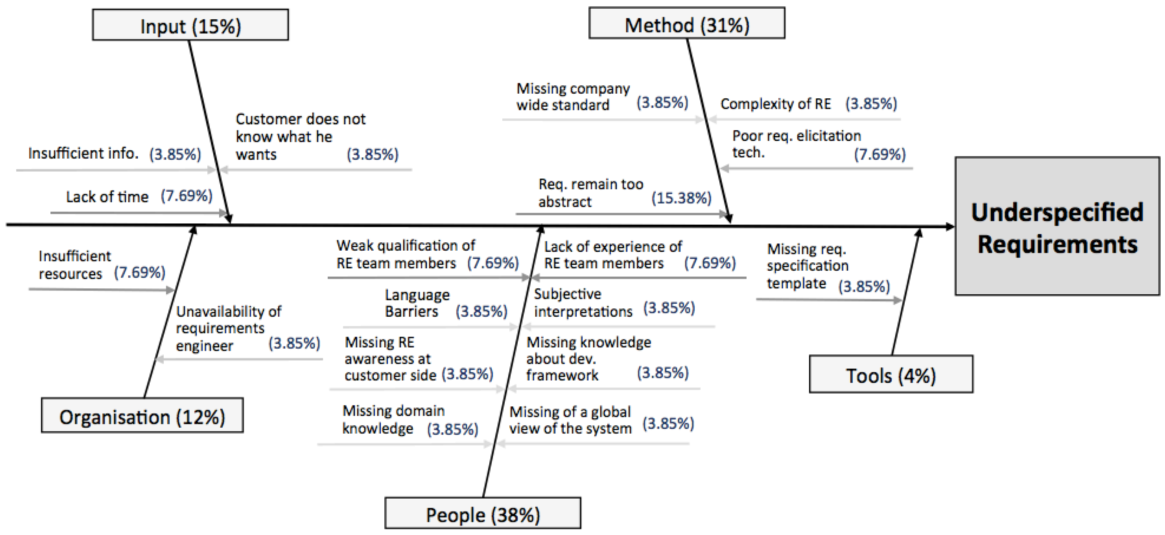

Input (15%)
Method (31%)
Missing company wide standard (3.85%)
Complexity of RE (3.85%)
Customer does not know what he wants (3.85%)
Insufficient info. (3.85%)
Poor req. elicitation tech. (7.69%)
Req. remain too abstract (15.38%)
Lack of time (7.69%)
Underspecified Requirements
Insufficient resources (7.69%)
Weak qualification of RE team members (7.69%)
Lack of experience of RE team members (7.69%)
Missing req. specification template (3.85%)
Unavailability of requirements engineer (3.85%)
Language Barriers (3.85%)
Subjective interpretations (3.85%)
Missing RE awareness at customer side (3.85%)
Missing knowledge about dev. framework (3.85%)
Tools (4%)
Organisation (12%)
Missing domain knowledge (3.85%)
Missing of a global view of the system (3.85%)
People (38%)

Figure 5.6. Causes for *Underspecified Requirements* problem on Agile organisations

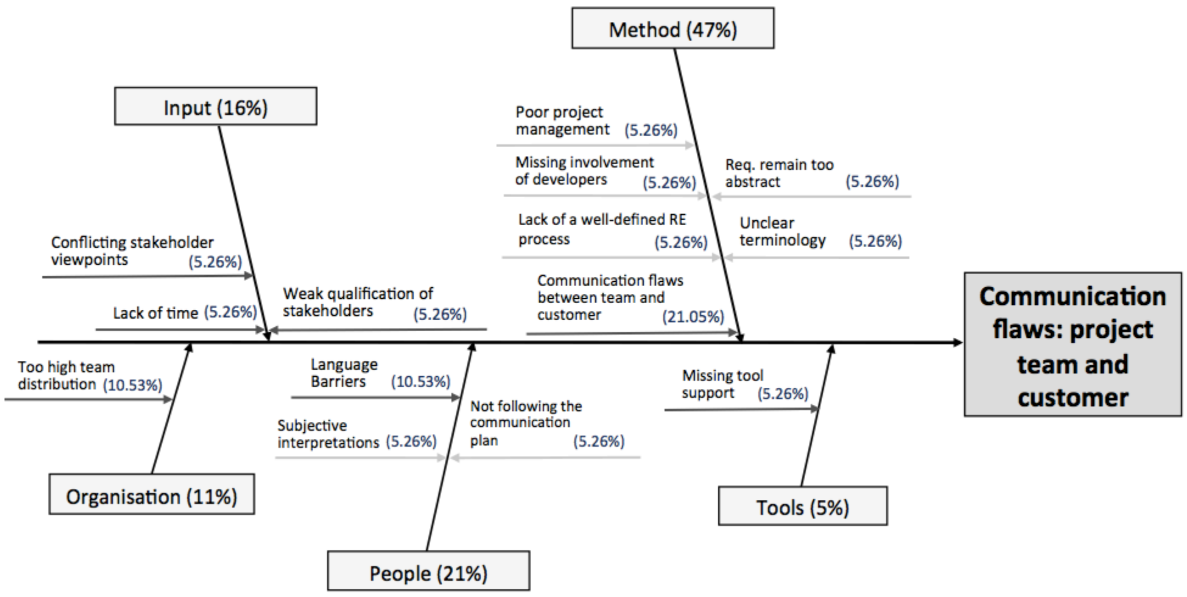

Method (47%)
Input (16%)
Poor project management (5.26%)
Missing involvement of developers (5.26%)
Req. remain too abstract (5.26%)
Lack of a well-defined RE process (5.26%)
Unclear terminology (5.26%)
Conflicting stakeholder viewpoints (5.26%)
Communication flaws between team and customer (21.05%)
Weak qualification of stakeholders (5.26%)
Lack of time (5.26%)
Communication flaws: project team and customer
Too high team distribution (10.53%)
Language Barriers (10.53%)
Missing tool support (5.26%)
Not following the communication plan (5.26%)
Subjective interpretations (5.26%)
Organisation (11%)
Tools (5%)
People (21%)

Figure 5.7. Causes for *Communication flaws between the project team and the customer* problem on Plan-Driven organisations

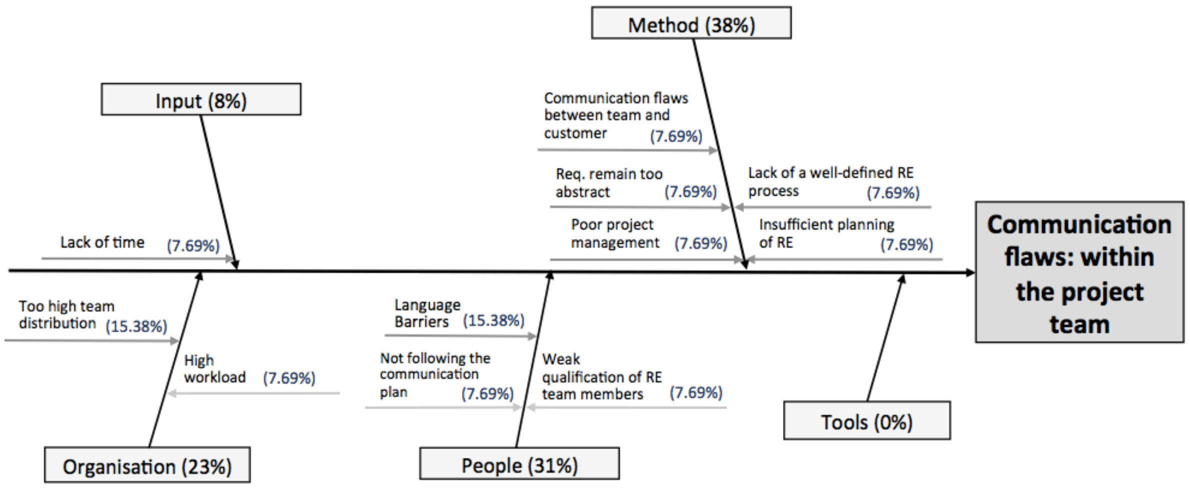

Method (38%)
Input (8%)
Communication flaws between team and customer (7.69%)
Req. remain too abstract (7.69%)
Lack of a well-defined RE process (7.69%)
Poor project management (7.69%)
Insufficient planning of RE (7.69%)
Lack of time (7.69%)
Communication flaws: within the project team
Too high team distribution (15.38%)
High workload (7.69%)
Language Barriers (15.38%)
Not following the communication plan (7.69%)
Weak qualification of RE team members (7.69%)
Organisation (23%)
People (31%)
Tools (0%)

Figure 5.8. Causes for *Communication flaws within the project team* problem on Plan-Driven organisations

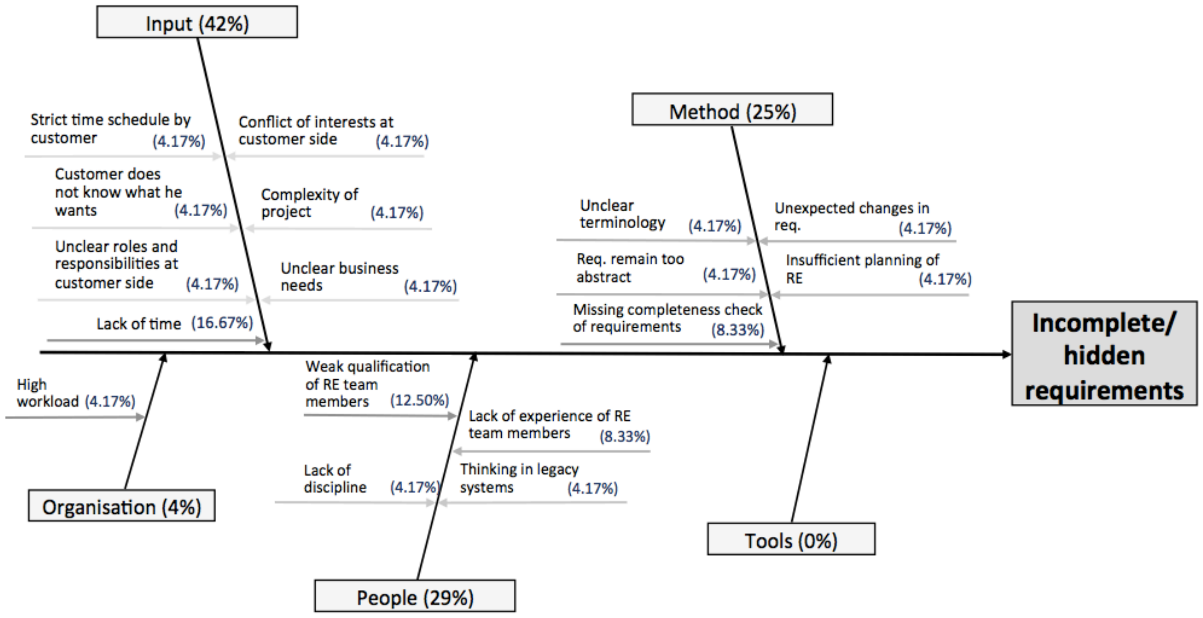

Input (42%)
Strict time schedule by customer (4.17%)
Conflict of interests at customer side (4.17%)
Customer does not know what he wants (4.17%)
Complexity of project (4.17%)
Unclear roles and responsibilities at customer side (4.17%)
Unclear business needs (4.17%)
Lack of time (16.67%)
Method (25%)
Unclear terminology (4.17%)
Unexpected changes in req. (4.17%)
Req. remain too abstract (4.17%)
Insufficient planning of RE (4.17%)
Missing completeness check of requirements (8.33%)
Incomplete/ hidden requirements
High workload (4.17%)
Weak qualification of RE team members (12.50%)
Lack of experience of RE team members (8.33%)
Lack of discipline (4.17%)
Thinking in legacy systems (4.17%)
Organisation (4%)
Tools (0%)
People (29%)

Figure 5.9. Causes for *Incomplete/hidden requirements* problem on Plan-Driven organisations

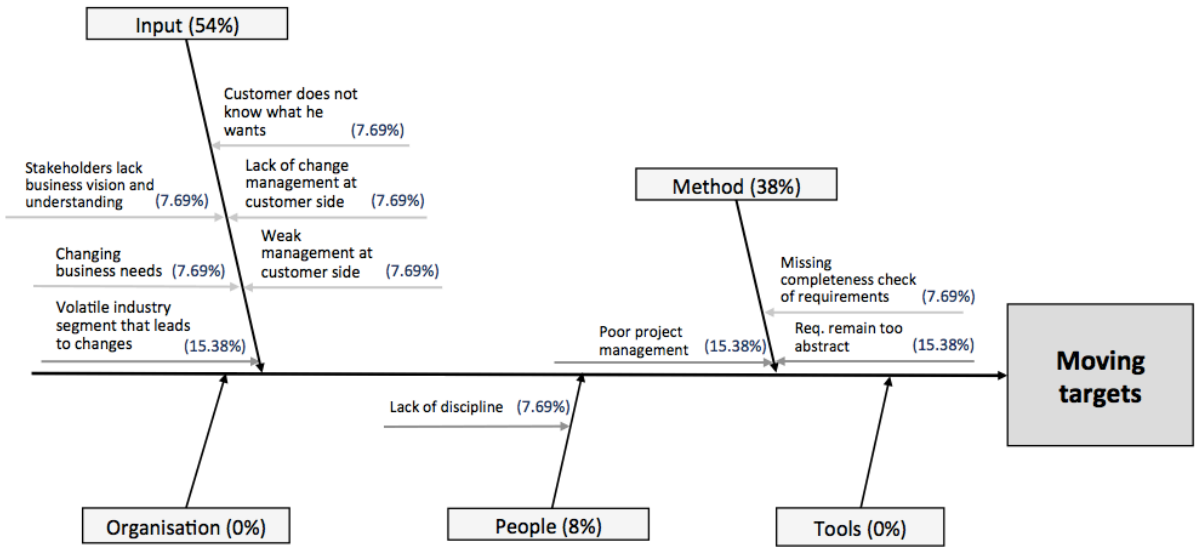

Input (54%)
Customer does not know what he wants (7.69%)
Stakeholders lack business vision and understanding (7.69%)
Lack of change management at customer side (7.69%)
Changing business needs (7.69%)
Weak management at customer side (7.69%)
Volatile industry segment that leads to changes (15.38%)
Method (38%)
Missing completeness check of requirements (7.69%)
Poor project management (15.38%)
Req. remain too abstract (15.38%)
Lack of discipline (7.69%)
Moving targets
Organisation (0%)
People (8%)
Tools (0%)

Figure 5.10. Causes for *Moving targets* problem on Plan-Driven organisations

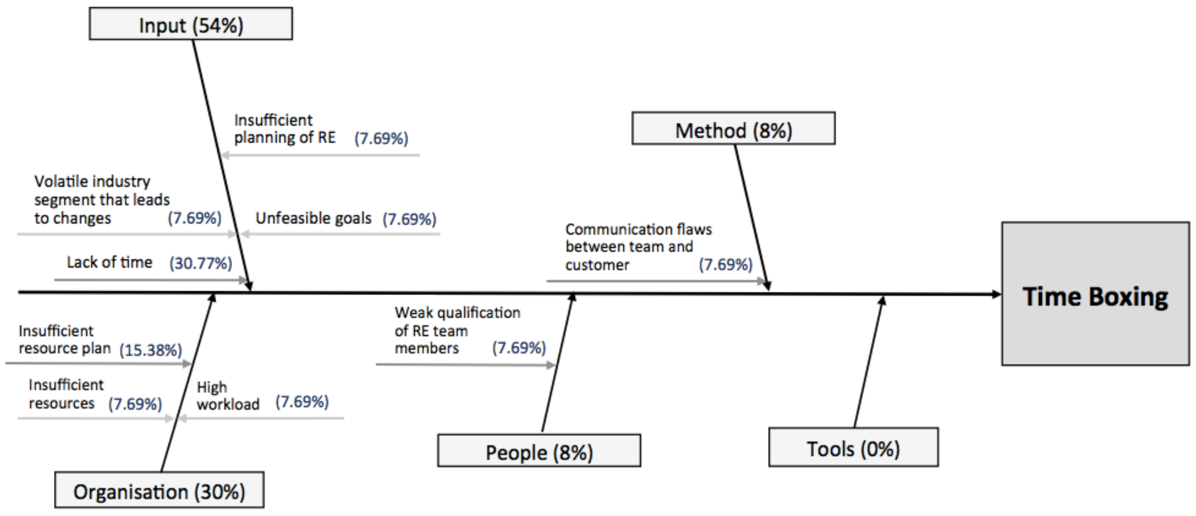

Input (54%)
Insufficient planning of RE (7.69%)
Volatile industry segment that leads to changes (7.69%)
Unfeasible goals (7.69%)
Lack of time (30.77%)
Method (8%)
Communication flaws between team and customer (7.69%)
Time Boxing
Insufficient resource plan (15.38%)
Insufficient resources (7.69%)
High workload (7.69%)
Weak qualification of RE team members (7.69%)
Organisation (30%)
People (8%)
Tools (0%)

Figure 5.11. Causes for *Time Boxing* problem on Plan-Driven organisations

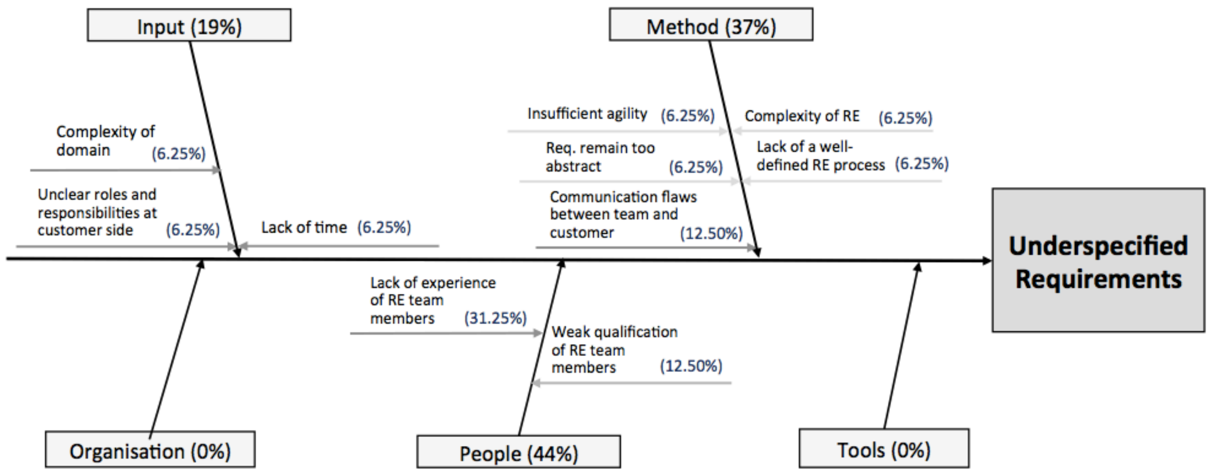

Input (19%)
Method (37%)
Complexity of domain (6.25%)
Unclear roles and responsibilities at customer side (6.25%)
Lack of time (6.25%)
Insufficient agility (6.25%)
Complexity of RE (6.25%)
Req. remain too abstract (6.25%)
Lack of a well-defined RE process (6.25%)
Communication flaws between team and customer (12.50%)
Underspecified Requirements
Lack of experience of RE team members (31.25%)
Weak qualification of RE team members (12.50%)
Organisation (0%)
People (44%)
Tools (0%)

Figure 5.12. Causes for *Underspecified Requirements* problem on Plan-Driven organisations

APPENDIX D – PRELIMINARY GUIDELINES FOR PREVENTING CRITICAL RE PROBLEMS ON AGILE AND PLAN-DRIVEN ORGANISATIONS

Next it is shown the preliminary guidelines for each cluster of the most critical RE problems for agile and plan-driven organisations. Table 18 to Table 23 show those preliminary guidelines for preventing the top 6 most critical RE problems and are disposed in alphabetic order of the problems.

Table 18. Preliminary guidelines for preventing *Communication flaws between the project team and the customer* on Agile and Plan-Driven organisations

Communication Flaws between the project team and the customer	
Agile	**Plan-Driven**
• Allocate a project manager to the project with large experience and expertise • Allocate a requirements engineer to the project with large experience and expertise • Avoid using technical terms during meetings with customers • Conduct daily stand-up meetings with the project team • Conduct regular meetings with the customer • Create a requirements specification template • Educate user representatives and managers about software requirements • Explain customers the importance of their contribution • Focus on business needs, assuring that requirements are sufficient and really necessary	• Allocate a project manager to the project with large experience and expertise • Allocate a requirements engineer to the project with large experience and expertise • Avoid using technical terms during meetings with customers • Centralize communication not allowing extra official communications • Conduct regular meetings with the customer • Create a project glossary with domain concepts • Create a well-defined RE process • Explain customers the importance of their contribution • Improve project management practices • Improve the documentation of the requirements specifications • Improve the management of the project schedule

• Improve the documentation of the requirements specifications • Improve the management of the project schedule • Introduce an early feedback loop with customer (e.g., conducting regular demos of new aggregated value) • Negotiate more time with the customer if necessary • Promote the integration of the project team • Provide training (if needed) to improve team skills • Spend more effort in requirements specification • Use an agile method • Use prototyping	• Introduce communication tools (e-mails, instant messages, telephone/video conference, intranet) • Promote the integration of the project team • Promote knowledge transfer within project team • Provide training (if needed) to improve team skills • Spend more effort in elicitation • Spend more effort in analysis and modelling • Use an agile method • Use prototyping • Work with small teams

Table 19. Preliminary guidelines for preventing *Communication flaws within the project team* on Agile and Plan-Driven organisations

Communication Flaws within the project team	
Agile	**Plan-Driven**
• Allocate a requirements engineer to the project with large experience and expertise • Centralize communication not allowing extra official communications • Conduct daily stand-up meetings with the project team • Conduct regular meetings with the customer • Create a requirements specification template	• Allocate a project manager to the project with large experience and expertise • Allocate a requirements engineer to the project with large experience and expertise • Allocate the team members to work no more than 40 hours a week to maintain the productivity • Avoid using technical terms during meetings with customers • Centralize communication not allowing extra official communications

• Create a team with people that has great empathy with each other • Create a well-defined RE process • Explain customers the importance of their contribution • Improve the documentation of the requirements specifications • Improve the management of task priority • Introduce an early feedback loop with customer • Introduce communication tools (e-mails, instant messages, telephone/video conference, intranet) • Improve the management of the project schedule • Promotion of eventual events to increase team interaction • Promote the integration of the project team • Provide training (if needed) to improve team skills • Spend more effort in elicitation • Spend more effort in analysis and modelling • Spend more effort in requirements specification	• Conduct regular meetings with the customer • Create an organisational unit for the software development professionals to keep them centralized and focused • Create a project glossary with domain concepts • Create a well-defined RE process • Explain customers the importance of their contribution • Improve project management practices • Improve the management of the project schedule • Motivate the team • Promote the integration of the project team • Promotion of knowledge transfer within project team • Provide training (if needed) to improve team skills • Spend more effort in elicitation • Spend more effort in analysis and modelling • Use prototyping • Use iterative development, with several increments (or sprints) and small releases • Work with a small teams

Table 20. Preliminary guidelines for preventing *Incomplete and/or hidden requirements* on Agile and Plan-Driven organisations

Incomplete and/or hidden requirements	
Agile	**Plan-Driven**
• Allocate a requirements engineer to the project with large experience and	• Allocate a requirements engineer to the project with large experience and expertise

expertise • Analyse technical feasibility of the requirements • Conduct daily stand-up meetings with the project team • Conduct regular meetings with the customer • Create a DoR (Definition of Ready) for the team, i.e., a checklist for accepting a requirement in a sprint • Create a requirements specification template • Create a requirements traceability matrix • Create a well-defined RE process • Educate user representatives and managers about software requirements • Explain customers the importance of their contribution • Explore use cases and scenarios during requirements engineering • Implement a requirements change management process • Improve the management of the project schedule • Integrate Testing and RE • Introduce an early feedback loop with customer (e.g., conducting regular demos of new aggregated value) • Measure requirements volatility (e.g., the amount of changes) • Provide training (if needed) on the business domain and overall system scope • Provide training (if needed) to improve	• Allocate the team members to work no more than 40 hours a week to maintain the productivity • Avoid using technical terms during meetings with customers • Conduct regular meetings with the customer • Create a project glossary with domain concepts • Educate user representatives and managers about software requirements • Establish a baseline and control versions of requirements documents • Explore use cases and scenarios during requirements engineering • Focus on business needs, assuring that requirements are sufficient and really necessary • Improve the documentation of the requirements specifications • Improve the management of the project schedule • Perform requirements reviews (e.g., peer reviews, technical reviews, inspections) • Promote knowledge transfer within project team • Provide training (if needed) on the business domain and overall system scope • Provide training (if needed) to improve team skills • Reuse requirements across projects • Review lessons learned regarding requirements on other projects

team skills • Renegotiate project commitments when requirements change • Spend more effort in analysis and modelling • Spend more effort in elicitation and analysis • Spend more effort in requirements specification • Spend more effort in requirements validation • Track the status of each requirement • Use a requirements management tool • Use prototyping	• Review the checklist of problems that may occur from similar system built before or if the checklist doesn't exists, create one to help future projects • Spend more effort in analysis and modelling • Spend more effort in elicitation • Spend more effort in requirements validation • Use prototyping • Write a vision and scope document

Table 21. Preliminary guidelines for preventing *Moving targets* on Agile and Plan-Driven organisations

Moving targets (changing goals, business processes and / or requirements)	
Agile	**Plan-Driven**
• Allocate a project manager with large experience and expertise • Allocate a requirements engineer to the project with large experience and expertise • Conduct regular meetings with the customer • Create a requirements traceability matrix • Explain customers the impact of changes • Focus on business needs, assuring that requirements are sufficient and really necessary • Implement a change management process • Improve the management of task priority	• Allocate a project manager with large experience and expertise • Conduct regular meetings with the customer • Educate user representatives and managers about software requirements • Explain customers the impact of changes • Implement a change management process • Improve the documentation of the requirements specifications • Measure requirements volatility (e.g., the amount of changes) • Obtain domain knowledge • Obtain prior acceptance by the customer

• Measure requirements volatility (e.g., the amount of changes) • Provide training (if needed) on the business domain and overall system scope • Spend more effort in analysis and modelling • Spend more effort in elicitation • Spend more effort in requirements validation • Use prototyping • Use a requirements management tool • Use iterative development, with several increments (or sprints) and small releases	about the change manage policy • Spend more effort in analysis and modelling • Spend more effort in elicitation • Spend more effort in requirements validation • Track the status of each requirement • Use a requirements management tool • Use iterative development, with several increments (or sprints) and small releases • Use prototyping • Perform requirements reviews (e.g., peer reviews, technical reviews, inspections)

Table 22. Preliminary guidelines for preventing *Time Boxing* on Agile and Plan-Driven organisations

Time boxing / Not enough time in general	
Agile	**Plan-Driven**
• Allocate a project manager with large experience and expertise • Allocate the team members to work no more than 40 hours a week to maintain the productivity • Analyse technical feasibility of the requirements • Conduct daily stand-up meetings with the project team • Conduct regular meetings with the customer • Create a requirements traceability matrix • Customer and the team willing to change • Explain customers the impact of changes • Focus on business needs, assuring that	• Allocate a requirements engineer to the project with large experience and expertise • Analyse technical feasibility of the requirements • Create a requirements specification template • Implement a change management process • Improve the documentation of the requirements specifications • Improve the management of the project schedule • Measure requirements volatility (e.g., the amount of changes) • Spend more effort in analysis and modelling

requirements are sufficient and really necessary • Implement a change management process • Improve the management of task priority • Improve the management of the project schedule • Introduce an early feedback loop with customer (e.g., conducting regular demos of new aggregated value) • Negotiate more time with the customer if necessary • Perform requirements-change impact analysis • Provide training (if needed) to improve team skills • Renegotiate project commitments when requirements change • Track the status of each requirement • Use a requirements management tool • Use iterative development, with several increments (or sprints) and small releases	• Spend more effort in elicitation • Use iterative development, with several increments (or sprints) and small releases

Table 23. Preliminary guidelines for preventing *Underspecified requirements* on Agile and Plan-Driven organisations

Underspecified requirements that are too abstract and allow for various interpretations	
Agile	**Plan-Driven**
• Allocate a project manager with large experience and expertise • Allocate a requirements engineer to the project with large experience and expertise • Avoid using technical terms during meetings with customers	• Allocate a requirements engineer to the project with large experience and expertise • Conduct regular meetings with the customer • Create a requirements specification template • Create a well-defined RE process

• Conduct regular meetings with the customer • Create a project glossary with domain concepts • Create a requirements specification template • Create a requirements traceability matrix • Create a well-defined RE process • Educate user representatives and managers about software requirements • Explain customers the importance of their contribution • Explore use cases and scenarios during requirements engineering • Improve the documentation of the requirements specifications • Improve the management of the project schedule • Obtain domain knowledge • Perform requirements reviews (e.g., peer reviews, technical reviews, inspections) • Provide training (if needed) to improve team skills • Spend more effort in analysis and modelling • Spend more effort in elicitation • Spend more effort in requirements specification • Spend more effort in requirements validation • Use prototyping	• Explain customers the importance of their contribution • Improve the documentation of the requirements specifications • Improve the management of the project schedule • Perform code peer reviews • Perform requirements reviews (e.g., peer reviews, technical reviews, inspections) • Provide training (if needed) to improve team skills • Spend more effort in analysis and modelling • Spend more effort in elicitation • Spend more effort in requirements specification • Track the status of each requirement • Use prototyping

APPENDIX E – EVALUATION OF THE PRELIMINARY GUIDELINES - SURVEY

Next it is shown the survey used to evaluate the preliminary guidelines for each cluster of the two most critical RE problems (*Communication flaws between the project team and the customer* and *Incomplete and/or hidden requirements*) for agile and plan-driven organisations.

Survey

Preventing Critical Requirements Engineering Problems

We compiled mitigation actions for preventing critical Requirements Engineering (RE) problems and would like to count on your expertise to give us feedback on their relevance.

Your Name (or Nickname):

For each RE problem listed below, analyse if the mitigation action would help preventing the problem and inform your level of agreement by marking it with an X:

Problem "Communication flaws between the project team and the customer" for Agile Organisations:

	Disagree	**Partially Disagree**	**Partially Agree**	**Agree**	**Not Sure**
Allocate a project manager to the project with large experience and expertise					
Allocate a requirements engineer to the project with large experience and expertise					
Avoid using technical terms during meetings with customers					
Conduct daily stand-up meetings					

with the project team					
Conduct regular meetings with the customer					
Create a requirements specification template					
Educate user representatives and managers about software requirements					
Explain customers the importance of their contribution					
Focus on business needs, assuring that requirements are sufficient and really necessary					
Improve the documentation of the requirements specifications					
Improve the management of the project schedule					
Introduce an early feedback loop with customer (e.g., conducting regular demos of new aggregated value)					
Negotiate more time with the customer if necessary					
Promote the integration of the project team					
Provide training (if needed) to improve team skills					
Spend more effort in requirements specification					
Use prototyping					

Others (other relevant actions not in the list):	

Problem "Communication flaws between the project team and the customer" for Plan-Driven Organisations:

	Disagree	**Partially Disagree**	**Partially Agree**	**Agree**	**Not Sure**
Allocate a project manager to the project with large experience and expertise					
Allocate a requirements engineer to the project with large experience and expertise					
Avoid using technical terms during meetings with customers					
Centralise communication not allowing extra official communications					
Conduct regular meetings with the customer					
Create a project glossary with domain concepts					
Create a well-defined RE process					
Explain customers the importance of their contribution					
Improve project management practices					
Improve the documentation of the requirements specifications					
Improve the management of the project schedule					
Introduce communication tools (e-mails, instant messages, telephone/video conference, intranet)					
Promote the integration of the project					

team					
Promote knowledge transfer within project team					
Provide training (if needed) to improve team skills					
Spend more effort in elicitation					
Spend more effort in analysis and modelling					
Use an agile method					
Use prototyping					
Work with small teams					

Others (other relevant actions not in the list):	

Problem "Incomplete and/or hidden requirements" for Agile Organisations:

	Disagree	**Partially Disagree**	**Partially Agree**	**Agree**	**Not Sure**
Allocate a requirements engineer to the project with large experience and expertise					
Analyse technical feasibility of the requirements					
Conduct daily stand-up meetings with the project team					
Conduct regular meetings with the customer					
Create a DoR (Definition of Ready) for the team, i.e., a checklist for accepting a requirement in a sprint					
Create a requirements specification template					

Create a requirements traceability matrix					
Create a well-defined RE process					
Educate user representatives and managers about software requirements					
Explain customers the importance of their contribution					
Explore use cases and scenarios during requirements engineering					
Implement a requirements change management process					
Improve the management of the project schedule					
Integrate Testing and RE					
Introduce an early feedback loop with customer (e.g., conducting regular demos of new aggregated value)					
Measure requirements volatility (e.g., the amount of changes)					
Provide training (if needed) on the business domain and overall system scope					
Provide training (if needed) to improve team skills					
Renegotiate project commitments when requirements change					
Spend more effort in analysis and modelling					
Spend more effort in elicitation and analysis					
Spend more effort in requirements					

specification					
Spend more effort in requirements validation					
Track the status of each requirement					
Use a requirements management tool					
Use prototyping					

Others (other relevant actions not in the list):	

Problem "Incomplete and/or hidden requirements" for Plan-Driven Organisations:

	Disagree	**Partially Disagree**	**Partially Agree**	**Agree**	**Not Sure**
Allocate a requirements engineer to the project with large experience and expertise					
Allocate the team members to work no more than 40 hours a week to maintain the productivity					
Avoid using technical terms during meetings with customers					
Conduct regular meetings with the customer					
Create a project glossary with domain concepts					
Educate user representatives and managers about software requirements					
Establish a baseline and control versions of requirements documents					
Explore use cases and scenarios during requirements engineering					
Focus on business needs, assuring					

that requirements are sufficient and really necessary					
Improve the documentation of the requirements specifications					
Improve the management of the project schedule					
Perform requirements reviews (e.g., peer reviews, technical reviews, inspections)					
Promote knowledge transfer within project team					
Provide training (if needed) on the business domain and overall system scope					
Provide training (if needed) to improve team skills					
Reuse requirements across projects					
Review lessons learned regarding requirements on other projects					
Review the checklist of problems that may occur from similar system built before or if the checklist doesn't exists, create one to help future projects					
Spend more effort in analysis and modelling					
Spend more effort in elicitation					
Spend more effort in requirements validation					
Use prototyping					
Write a vision and scope document					

Others (other relevant actions not in the list):	

ANNEX A – NAPIRE SURVEY

The NaPiRE questionnaire is presented next. All the research conducted in this work was based on the answers of this survey, which was run in 2014-2015.

Version: 1.4 (Changes in questionnaire from 2013 from feedback rounds)

Questionnaire

1. Home

Dear Survey Participant, thank you very much for sparing 15–30 minutes of your valuable time by answering this questionnaire!

The Requirements Engineering Survey 2014 is conducted by an internationally distributed group of researchers and shall help us getting a better understanding of trends in Requirements Engineering (RE).

Goal of the survey: We are interested in your personal expectations on and experiences with Requirements Engineering to understand the status quo and expectations Requirements Engineering process definitions, their improvement, and their application in projects – all relying on your personal expert opinion. This shall give us insight trends in RE and lay the foundation to steer academic and industrial research in a problem-driven manner, i.e. it shall help detect practically relevant problems and go Engineering.

Structure of the survey: The Requirements Engineering Survey includes (at most) 35 questions, structured into 4 categories:

1. General information about you and your company
2. Status quo in RE at your company
3. Status quo in RE improvement at your company
4. Contemporary problems you experienced in RE and how these problems manifest themselves in the process

Please answer the questions as accurately as possible.

At the end of the survey, you will be asked to enter your email address. In case you agree, we will provide you with an overview of the survey results. In any case, the survey follows a high academic standard and is conducted anonymously. We will not associate your email address with your answers and exclusively use the address with the survey results.

For further information / questions, please contact: Dr. Daniel Mendez Technische Universität München http://www4.in.tum.de/~mendezfe

2. Metadata

The following questions ask about general information about your company and you.

What is the size of your company (software and other areas)?

- o 1-10 employees
- o 11-50 employees
- o 51-250 employees
- o 251-500 employees
- o 501-1000 employees
- o 1001-2000 employees
- o more than 2000 employees

Please briefly describe the main sector of your company and the application domain of the software you build.

Does your company participate in globally distributed projects?

- o Yes
- o No

In case your company participates in globally distributed projects, in which country are you personally located?

To which project role are you most frequently assigned to in those projects?

- o Business Analyst
- o Requirements Engineer
- o Project Lead / Project Manager
- o Test Manager / Tester
- o Architect
- o Developer
- o Other

How do you rate your experience in this role?

- o Novice (up to 1 year experience)
- o Experienced (1-3 years experience)
- o Expert (more than 3 years experience)

Which organisational role does your company take most frequently in your projects?

- ○ Customer
- ○ Product development
- ○ Contractor
- ○ Other

Which process model (or variation of it) do you follow in your projects?

- ○ Waterfall
- ○ V-Modell
- ○ XT
- ○ Scrum
- ○ Extreme Programming (XP)
- ○ Rational Unified Process
- ○ Other

---

3. Status Quo in Requirements Engineering

The following questions address the status quo in RE in your company.

How do you elicit requirements?

- ○ Interviews
- ○ Scenarios
- ○ Prototyping
- ○ Facilitated meetings (including workshops)
- ○ Observation
- ○ Other

How do you document functional requirements?

	Domain / Business Process Models	**Use Case Models**	**Goal Models**	**Data Models**	**Structured Requirements Lists**
Free form textual					
Textual with constraints					
Semi-formal (UML)					

Formal					

How do you document non-functional requirements?

- o We use quantified textual requirements
- o We use non-quantified textual requirements
- o Other

How do you deal with changing requirements after the initial release?

- o We regularly change the requirements specification.
- o We update our product backlog.
- o We only work with change requests.
- o Other

Which traces do you explicitly manage?

- o Traces between requirements and code.
- o Traces between requirements and design documents.
- o Other
- o None.

How do you analyse the effect of changes to requirements?

- o We do impact analysis between requirements.
- o We do impact analysis on the code.
- o Other
- o We do not analyse the effect of changes to requirements.

How do you align the software test with the requirements?

- o Testers participate in requirements reviews.
- o We check the coverage of requirements with tests.
- o We define acceptance criteria for requirements.
- o We derive tests from system models.
- o Other
- o We do not align test and requirements.

---

4. RE Process Standard Question

What requirements engineering company standard (RE reference model) have you established at your company?

- o A standard that is predefined according to a regulation (e.g., ITIL)

- ○ A standard that is predefined by the development process (e.g., Rational Unified Process, Scrum)
- ○ An own standard that defines the coarse process with deliverables, milestones, and phases
- ○ An own standard that defines the process including roles and responsibilities.
- ○ An own standard that defines artefacts and offers document templates
- ○ None
- ○ Other

---

5. Status Quo in RE Process Standard

The following questions consider the status quo in your company-specific RE standard including its application in projects, and, if reason controlling.

Which reasons do you agree with as a motivation to define a company standard for requirements engineering in your company?

	I disagree	I partially disagree	Neutral	I partially agree	I agree
Compliance to regulations and standards (like CMMI)					
Seamless development by integrating Requirements Engineering into the development process					
Better tool support					
Formal prerequisite for project acquisition in your domain					
Support of distributed development Better support of progress control					
Better quality assurance of the artifacts (e.g., within quality gates)					
Support of benchmarks and / or comparison of different projects					
Support of project management and planning					

Higher efficiency					
Knowledge transfer					

Which reasons do you see as a barrier to define a company-wide reference model for requirements engineering in your company?

	I disagree	**I partially disagree**	**Neutral**	**I partially agree**	**I agree**
Higher process complexity					
Higher demand for communication					
Lower efficiency					
Missing willingness for changes					
Missing possibilities of standardisation					

Is the requirements engineering standard mandatory and practiced?

- o It is mandatory and practiced.
- o It is mandatory but not practiced.
- o It is practiced but not mandatory
- o No

How do you check the application of your requirements engineering standard?

- o Via project assessments
- o Via analytical quality assurance, e.g., as part of quality gates
- o Via constructive quality assurance, e.g., via checklists or templates
- o Other ______________________
- o It is not checked.

How do you perform change management in your requirements engineering?

- o We have a continuous change management.
- o We have a change management approach that applies after formally accepting a requirements specification.
- o We have a change management that applies during RE.
- o We do not consider a change management in RE.

How is your requirements engineering standard applied (tailored) in your regular projects?

- o We have defined a tailoring approach that continuously guides the application of the standard in our project
- o We have tool support for tailoring our Requirements Engineering standard
- o At the beginning of a project, the project lead / requirements engineer tailors the standard based on experiences
- o Other
- o We do not consider a particular tailoring approach

---

6. RE Improvement Question

Is your requirements engineering continuously improved?

- o Yes, we improve our requirements engineering via an own business unit / role.
- o Yes, we improve our requirements engineering via external consultants.
- o Yes, our project teams improve requirements engineering.
- o No

---

7. Status Quo in Requirements Engineering Improvement

The following questions consider the status quo in Requirements Engineering improvement in your company.

Why do you continuously improve your requirements engineering?

- o It helps us to determine our strenghts and weaknesses and act accordingly
- o An improvement is expected by our customer
- o We conduct the improvement to obtain a certain certification.
- o An improvement is demanded by a regulation (e.g., CMMI, Cobit, or ITIL)
- o Other

Do you use a normative, external standard for your improvement?

- o Yes, we use an external standard for assessing RE (e.g., CMMI for RE)
- o No, we use an internally defined (company-specific) standard for improving RE

If you use an internal standard for improving your Requirements Engineering and not an external one, what were the reasons?

---

8. Contemporary Requirements Engineering Problems

The following questions of the questionnaire address contemporary problems you experienced in RE including the company standard a experiences. Please answer the questions as honestly as possible.

Please rate the following statements for your requirements engineering standard according to your experiences

Our Requirements Engineering standard...	**I disagree**	**I partially disagree**	**Neutral**	**I partially agree**	**I agree**
...is too hard to understand					
...is too complex					
...is too abstract					
...does not support the specification of precise requirements					
...does not scale to our projects' high complexity					
...is too heavy weight for our projects (e.g., it does not support agility)					
...is not flexible enough (e.g., it offers no means to tackle moving targets / change- intensive requirements)					
...does not sufficiently define a clear terminology					
...gives no guidance on how to create the specification documents					
...does not sufficiently allow for deviations according to project circumstances that cannot be formalised (e.g., politically motivated underspecified					

requirements)					
...does not fit to the variety of our projects (e.g., size or technical domains).					
...does not sufficiently define roles and responsibilities					

Considering your personal experiences, how do the following (more general) problems in requirements engineering apply to your projects?

	I disagree	I partially disagree	Neutral	I partially agree	I agree
Communication flaws within the project development team					
Communication flaws between developers and the customer					
Terminological problems					
Unclear responsibilities					
Incomplete					
Implicit requirements not made explicit					
Insufficient support by project lead					
Insufficient support by customer					
Stakeholders with difficulties in separating requirements from previously known solution designs					
Inconsistent requirements Missing traceability					
Moving targets (changing goals, business processes and / or requirements)					
"Gold plating" (implementation of features without corresponding					

requirements)					
Weak access to customer needs and / or (internal) business information					
Weak knowledge of customer's application domain					
Weak relationship between customer and project lead					
Time boxing / Not enough time in general					
Discrepancy between high degree of innovation and need for formal acceptance of (potentially wrong / incomplete / unknown) requirements					
Technically unfeasible requirements					
Underspecified requirements that are too abstract and allow for various interpretations					
Unclear / unmeasurable non-functional requirements					
Volatile customer's business domain regarding, e.g., changing points of contact, business processes or requirements					

Considering your personally experienced problems (stated in the previous question), which ones would you classify as the five most critical ones (o relevance).

	Problem experienced in your projects:
Problem #1 (most critical one)	Please make a selection: • Communication flaws within the project team • Communication flaws between the project team and the customer • Terminological problems

	• Unclear responsibilities • Incomplete and / or hidden requirements • Insufficient support by project lead • Insufficient support by customer • Stakeholders with difficulties in separating requirements from previously known solution designs • Inconsistent requirements • Missing traceability • Moving targets (changing goals, business processes and / or requirements) • "Gold plating" (implementation of features without corresponding requirements) • Weak access to customer needs and / or (internal) business information • Weak knowledge of customer's application domain • Weak relationship to customer • Time boxing / Not enough time in general • Discrepancy between high degree of innovation and need for formal acceptance of (potentially wrong / incomplete / unknown) requirements • Technically unfeasible requirements • Underspecified requirements that are too abstract and allow for various interpretations • Unclear / unmeasurable non-functional requirements • Volatile customer's business domain regarding, e.g., changing points of contact, business processes or requirements
Problem #2	Please make a selection: • Communication flaws within the project team • Communication flaws between the project team and the customer • Terminological problems

	• Unclear responsibilities • Incomplete and / or hidden requirements • Insufficient support by project lead • Insufficient support by customer • Stakeholders with difficulties in separating requirements from previously known solution designs • Inconsistent requirements • Missing traceability • Moving targets (changing goals, business processes and / or requirements) • "Gold plating" (implementation of features without corresponding requirements) • Weak access to customer needs and / or (internal) business information • Weak knowledge of customer's application domain • Weak relationship to customer • Time boxing / Not enough time in general • Discrepancy between high degree of innovation and need for formal acceptance of (potentially wrong / incomplete / unknown) requirements • Technically unfeasible requirements • Underspecified requirements that are too abstract and allow for various interpretations • Unclear / unmeasurable non-functional requirements • Volatile customer's business domain regarding, e.g., changing points of contact, business processes or requirements
Problem #3	Please make a selection: • Communication flaws within the project team • Communication flaws between the project team and the customer • Terminological problems

	• Unclear responsibilities • Incomplete and / or hidden requirements • Insufficient support by project lead • Insufficient support by customer • Stakeholders with difficulties in separating requirements from previously known solution designs • Inconsistent requirements • Missing traceability • Moving targets (changing goals, business processes and / or requirements) • "Gold plating" (implementation of features without corresponding requirements) • Weak access to customer needs and / or (internal) business information • Weak knowledge of customer's application domain • Weak relationship to customer • Time boxing / Not enough time in general • Discrepancy between high degree of innovation and need for formal acceptance of (potentially wrong / incomplete / unknown) requirements • Technically unfeasible requirements • Underspecified requirements that are too abstract and allow for various interpretations • Unclear / unmeasurable non-functional requirements • Volatile customer's business domain regarding, e.g., changing points of contact, business processes or requirements
Problem #4	Please make a selection: • Communication flaws within the project team • Communication flaws between the project team and the customer • Terminological problems

	• Unclear responsibilities • Incomplete and / or hidden requirements • Insufficient support by project lead • Insufficient support by customer • Stakeholders with difficulties in separating requirements from previously known solution designs • Inconsistent requirements • Missing traceability • Moving targets (changing goals, business processes and / or requirements) • "Gold plating" (implementation of features without corresponding requirements) • Weak access to customer needs and / or (internal) business information • Weak knowledge of customer's application domain • Weak relationship to customer • Time boxing / Not enough time in general • Discrepancy between high degree of innovation and need for formal acceptance of (potentially wrong / incomplete / unknown) requirements • Technically unfeasible requirements • Underspecified requirements that are too abstract and allow for various interpretations • Unclear / unmeasurable non-functional requirements • Volatile customer's business domain regarding, e.g., changing points of contact, business processes or requirements
Problem #5	Please make a selection: • Communication flaws within the project team • Communication flaws between the project team and the customer • Terminological problems

	• Unclear responsibilities • Incomplete and / or hidden requirements • Insufficient support by project lead • Insufficient support by customer • Stakeholders with difficulties in separating requirements from previously known solution designs • Inconsistent requirements • Missing traceability • Moving targets (changing goals, business processes and / or requirements) • "Gold plating" (implementation of features without corresponding requirements) • Weak access to customer needs and / or (internal) business information • Weak knowledge of customer's application domain • Weak relationship to customer • Time boxing / Not enough time in general • Discrepancy between high degree of innovation and need for formal acceptance of (potentially wrong / incomplete / unknown) requirements • Technically unfeasible requirements • Underspecified requirements that are too abstract and allow for various interpretations • Unclear / unmeasurable non-functional requirements • Volatile customer's business domain regarding, e.g., changing points of contact, business processes or requirements

---

9. Contemporary Problems Manifestation

The last questions of the questionnaire consider contemporary your experiences with the severity of the contemporary problems you e Please answer the questions as accurately as possible.

Considering your personally experienced most critical problems (selected in the previous question), which causes do they have?

#v_342#

#v_344#

#v_350#

#v_348#

#v_350#

#v_350#

Considering your personally experienced most critical problems (selected in the previous question), which would you classify as a major cause for p all)?

#v_342#

#v_344#

#v_346#

#v_348#

#v_350#

---

10. Extra question and Email

Is there any other aspect that you experience in your RE process and that remains unaddressed in the questions until now?

In case you would like to be notified about the results, please fill in your email-adress.

---

11. End

Thank you very much for participating in this survey. We very much appreciate the effort you spent in answering the questions that help us investigate trends in industrial RE. In case you entered your email in the previous item notify you about the results as soon as possible.