Česká zemědělská univerzita v Praze
Provozně ekonomická fakulta

Management of software projects based on object-oriented technology

Doctoral thesis in Information Management

Sandra Milena Choles Arvilla

Supervisor: Doc. Ing. Vojtěch Merunka, Ph.D.

I would like to thank my supervisor Doc. Ing. Vojtěch Merunka for all his advise, support and guidance during this process.

A mis padres por su apoyo y paciencia.

To my friends for all the fun and the support.

God thanks for the endurance and energy

Sandra

Dissertation Thesis Topic

Keywords

Methodology, process, agile development, risk management.

Abstract

This study examines the existing risk management practices commonly used for classic software development. The goal is to integrate the elements of the traditional risk management methodologies to create a new agile risk management methodology. The thesis focuses on techniques that can be easily implemented in extreme programming (XP) and SCRUM. This study is motivated by the following research questions: What are the elements of existing quality assurance tools that could meet the principles of agile development? And is it possible to use risk estimation for improving quality in agile projects? The thesis presents a synthesis of the most common risk management techniques, as well as an introduction to agile methods XP and SCRUM. The proposal integrates the concepts of Failure Mode and Effect Analysis into the iterative life cycle of an agile software project.

The thesis presents a metamodel which integrates the concepts of agile development methodologies: SCRUM and XP with the FMEA concepts for risk quantification. The model was partly implemented into a real development project. Partial results show the improvement in early identification of failures and allowed to reconsider the Sprint plan.

Table of Contents

Dissertation Thesis Topic 3

Keywords 3

Abstract 3

Table of Contents 4

1 Introduction 6

2 Goals 8

Hypothesis 8

3 Methodology 10

Motivation 10

Thesis structure 11

Limitations and threats 12

Related topics that are not contained in the work 13

4 Literature review 14

4.1 Quality assurance 14

4.1.1 Definition of Quality 15

4.1.2 Risk management 20

4.1.3 Practices and tools for quality assurance 22

4.2 Agile software development 37

4.2.1 SCRUM 39

4.2.2 Extreme programming 43

4.3 Agile risk management practices 48

4.4 Metamodel vs. Model 49

5 Discussion 52

5.1 Analysis of existing methods 52

6 Solution - Elaboration of hypothesis 56

6.1 Analogy 60

6.2 Metamodeling Agile Risk Management 62

6.2.1 Identification of objects and properties 63

6.3 Concept to object 65

6.4 Graphic representation 66

7 Verification and validation 75

7.1 Model validation 80

7.1.1 Scenario 80

7.1.2 Study case 82

8 Discussion of Results 93

9 Conclusion 94

9.1 Summary of dissertation 94

9.2 Suggestions for further research 95

10 List of abbreviations 97

11 List of Tables 97

12 List of Figures 98

13 Bibliography 99

1 Introduction

Agile methodologies were created to provide the user with several releases of the software as fast as possible assuming continuous variability in the requirements and design. Functional software is the only certain measure of progress; therefore continuous deliveries of them are required. Among the characteristics that agile methodologies should accomplish, according to the agile manifesto, customer satisfaction is one of their main focuses as the first principle establishes:

"Our highest priority is to satisfy the customer through early and continuous delivery of valuable software." (Beck, Grenning, Martin, & Beedle, 2001)

We could say that a customer or user is considered satisfied when all the agreed requirements have been delivered on time and on budget.

In order to comply with the main principles of this approach, agile development teams should be totally receptive to continuous changes in the requirements. Experts in software development quality assurance as Lindvall, Boehm and others, have discovered that the quality of personnel required for this type of projects is higher than (Lindvall, Basili, Boehm, Costa, Dangle, & Shull, Empirical Findings in Agile Methods, 2002) Experience and communication skills become as influential as technical knowledge. Therefore, the software development team and the business analysts cannot be independent teams; they should have continuous cooperation and clear communication. Surveys like (Chow T., 2008) shows that what they call high-caliber team is one of the critical success factors in agile projects.

Agile development methodologies have become very trendy and successful software development techniques. However, there are still many critics regarding potential overspending due to the continuous changes in requirements and design. Authors from Carnegie Mellon institute like (Levine, 2005), (Nelson, Taran, & Hinojosa, 2008) Consider that although there are principles of some agile methodologies that contribute to quality assurance, the truth is that there are no formal processes defined for risk identification and control within the agile approach. This statement motivates the development of this thesis, which intends to formalize a risk management (RM) model suitable for agile software development.

2 Goals

This thesis is based on a literature review of most popular agile development and risk management methodologies. The main sources base for this research were (Beck, Extreme Programming Explained: Embrace Change, 2004), (Boehm B. , 1991), (Boehm B. , 2002), (Nyfjord, Towards integrating agile development and risk management, 2008), (Project Management Institute PMI, 2013) with the support of many other sources. This work is divided in 3 phases: literature collection and review, identification of gaps or opportunity for research and modeling of a methodological proposal.

The principal goals of this thesis are:

- To identify the agile practices that ensure quality software projects.
- To define a methodological approach for RM processes applicable to projects developed using XP or SCRUM. This approach will be based on existing methods for identification, evaluation and risk controlling.

In order to pursue these goals, there have been defined sub-goals:

- To review the current state of art in agile practices focusing on risk management activities.
- To identify the most common risks practices that could be compatible with XP and SCRUM projects.
- To design a metamodel of agile processes from a risk management approach and base on this metamodel propose new diagram to integrate the risk management tasks within the agile workflow.

Hypothesis

We believe that implementing more accurate risk analysis at the beginning of agile iteration could help to identify possible user stories that increase the risk of failure and therefore should be changed or avoided.

Our hypothesis is base on the following aspects:

- The literature shows that agile software development is not considering strong practices related to risk management. One of the highest risks within an agile project is the need of back track functionalities that were not necessary to implement or represent a high risk for the project and /or the final product.
- Users demand more and more functionality that the agile team has to divide into iterations.
- We assume the existence of analogy between FMEA and Software engineering projects development that allows us to merge the concepts that aims to quantify risks effect and tolerance.

3 Methodology

Based on principle of analogy, we will apply some of the already existing risk management techniques into the new area of Agile approach in Software Development in order to improve the quality assurance, for example to minimize those initial requirements, which would have later recognized as unfeasible. Our methodology is based on a thoughtful synthesis of agile methods and risk management methods.

The methodology of this dissertation is based on the literature review, analysis of current methodologies and/or tools that allows the author to develop a metamodel for new agile risk management strategy.

The literature review goal is to find a gap in agile software development in terms of risk management processes. In general we could say there are enough tools to synchronize agile approach and risk management practices; however a formal definition is required for a successful integration of both concepts.

The dissertation intends to generate a metamodel proposal to integrate Risk management basic activities with SCRUM and XP practices. The modeling strategy is based on the GOPRR metamodel. MetaEdit+ is used as the modeling tool.

A validation of the metamodel is executed creating corresponding data models based on regular scenarios of real projects.

Motivation

There is extensive research on software project quality assurance and project success forecasting. However, the real application of these methodologies and concepts found still did not provide respectable results.

The last CHAOS reports (Standish Group, 2013) show of slightly increasing in the success rate of software projects. These reports also show interesting data regarding the issues that compromise the success of a project. Time continues being the most critical aspect in terms of overruns.

Most of the factors analyzed and presented in the Chaos report correspond to the concepts part of the SQuaRE (Software product Quality Requirements and Evaluation) (International Standard Organization (ISO), 2011). These are the concepts that we rely on their compatibility with some of the risk management practices used in engineering, as it is FMEA (Stamatis, 2003).

Thesis structure

First chapter is devoted to the introduction following by the intention of the thesis and the goals and sub goals identified. Other topics described in this section are related to the methodology implemented for this thesis and brief concepts that will be described more in general in the following sections.

The second section describes the areas and concepts related to this area. This chapter is divided into four sections:

1. Quality assurance: Basic concepts of quality n disk management practices are presented in this section. Highlighting the relevant concepts for the purpose of this study.
2. Agile software development: Agile manifesto and its foundation is explained. Even that there are many agile development methodologies, this thesis focus strictly in the two more popular and world wide used: SCRUM and Extreme Programming (XP).
3. Agile risk management: As described in the goals of this thesis, our intention is to identify the implicit practices of risk management within SCRUM and XP.

4. Metamodeling: This last section explains the concept of metamodeling and its purpose. General steps in the process of creating a new metamodel are also presented in this section.

The third chapter is an analysis of the literature research findings. The section presents an analysis of the methods explained and the reasoning in order to focus the thesis on some of these methods.

The fourth chapter is devoted to the implementation of the quality assurance and risk management concepts into agile development. The section explains how these concepts are compatible with the agile approach and how the integration of them are suitable for agile iterations. The section concludes with a graphical representation of the metamodel proposal and implementation model for a given scenario.

The fifth chapter contains a summary of the dissertation and interpretation of the results achieved.

The last section of the thesis presents conclusions and suggestions for further studies.

At the end of the dissertation provides an overview of the sources used in this work, list of abbreviations, list of figures, list of tables.

Limitations and threats

We assumed that the proposed method that will be described in detail, finds its place on software development teams that are familiar with the agile technologies and understand the concepts and processes of risk management. This study aims the areas where project manager and scrum master take action. The operators implementing user stories work only on the model according to the presented definition.

Its application in the correct use does not pose high demands on resources (finance, human resources) and its implementation for its relative simplicity does not generate significant additional cost to the firm.

This thesis is limited to the design and verification of the metamodel to integrate risk management into SCRUM and XP. It may need modifications in order to be implemented in a different agile methodology.

We assume that the metamodel described below may have been already developed by some other person/institution. At the moment of the conclusion of this study we did not recognize any similar work aiming the same goals. Related works are mentioned in the literature review and or cited in other sections of this document.

Related topics that are not contained in the work

The areas of project management, risks management and agile software development are very extensive. Therefore, we feel the need delineate the content of this study and specify the topics that do not correspond to the scope. The exclusion of these topics does not affect the achievement of the proposed goals.

This study do not include areas/topics like:

- Detailed description of classic software development. Not about classic development.
- Additional agile approaches and practices like: Adaptive Software Development (ASD), Agile Unified Process (AUP), Crystal Clear, Dynamic systems development method (DSDM), Essential Unified Process (EssUP), Feature Driven Development (FDD), Lean Software Development (LSD).
- Description and or implementation of risk management in other areas than software development.

4 Literature review

This section introduces the theoretical elements that support this thesis. The selection of concepts is based on the goals intended to achieve and previously described. As a result, there are three areas to explore: quality assurance, risk management practices for software development and agile methodologies.

Section 2.1 introduces quality management concepts and describes de SQuaRE model for software development quality assurance. The sections presents risk management processes and the findings of their implementation in the software development industry.

Section 2.2 describes agile software development approach. The sections introduces

Section 2.3 and 2.4 have been developed with the intention of present the combination of agile software development and risk management practices.

There have been several investigations about how agile development can coexist with classic practices in order to assure quality.

At last there is a description of metamodel and modeling concepts with the purpose of the development of this thesis.

4.1 Quality assurance

Quality assurance has been the focus of many studies (Příbrský, Kvantifikovaný přístup k jakosti informačního zabezpečení pro podporu evaluace informačních technologií, 2012), (Boehm B. , 1991), (Project Management Institute PMI, 2013), (International Standard Organization (ISO), 2011), (Standish Group, 2013) for a long time. Several techniques, frameworks and models have been defined with the intention to assure success in software project and deliver quality final product according to the user expectations. This section explores the basic concepts of quality and its relation with software development.

4.1.1 Definition of Quality

Project management as a science discusses the aspects influencing the development and success of any project. (Project Management Institute PMI, 2013)

These aspects define a triangle as shown in the figure No. 1

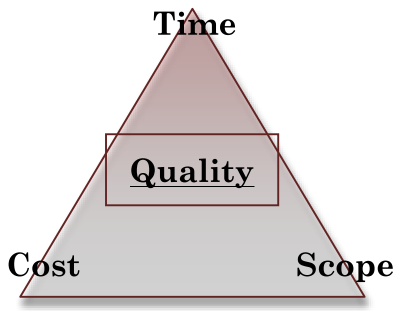


Fig. 1. Project management triangle

- Scope: Defines in terms of software engineering the functionalities so called requirements of the final product. Later on, we will see how this definition is transform in agile terms and reflects more the relationship with the user/client.
- Cost: is not necessarily expressed in a specific currency and amount. This aspect of project describes the resources necessary for the execution or itself. Usually the budget or the cost is defined or calculated by the product owner, which habitually is not the business analyst or the project manager.
- Time: This aspect refers to the desired or already agreed time of execution and expected release date. Regularly the team uses several tools for time estimation in order to plan releases based on the complexity of the requirements. When the time is fixed, the number of requirements or functionalities to be included within a release should be adjusted to the available time.

Not surprisingly *Quality* is in the very middle of the triangle. The three aspects explained before have a big impact in the quality of the project and its final product.

The project team and specially the project manager have the not-so-easy task to “play” with the vertices of the triangle to agree with the stakeholders each of these aspects. The goal is to achieve a more realistic and achievable distribution, which may not be according to every stakeholder demands. However, it should guarantee a higher rate of success and quality product.

Despite the fact that the triangle may allow the reduction of one or two of the vertices, the concept of quality (in the middle) remains as the core of the triangle. Meaning that any change in any of the vertices will immediately have a n impact in the quality.

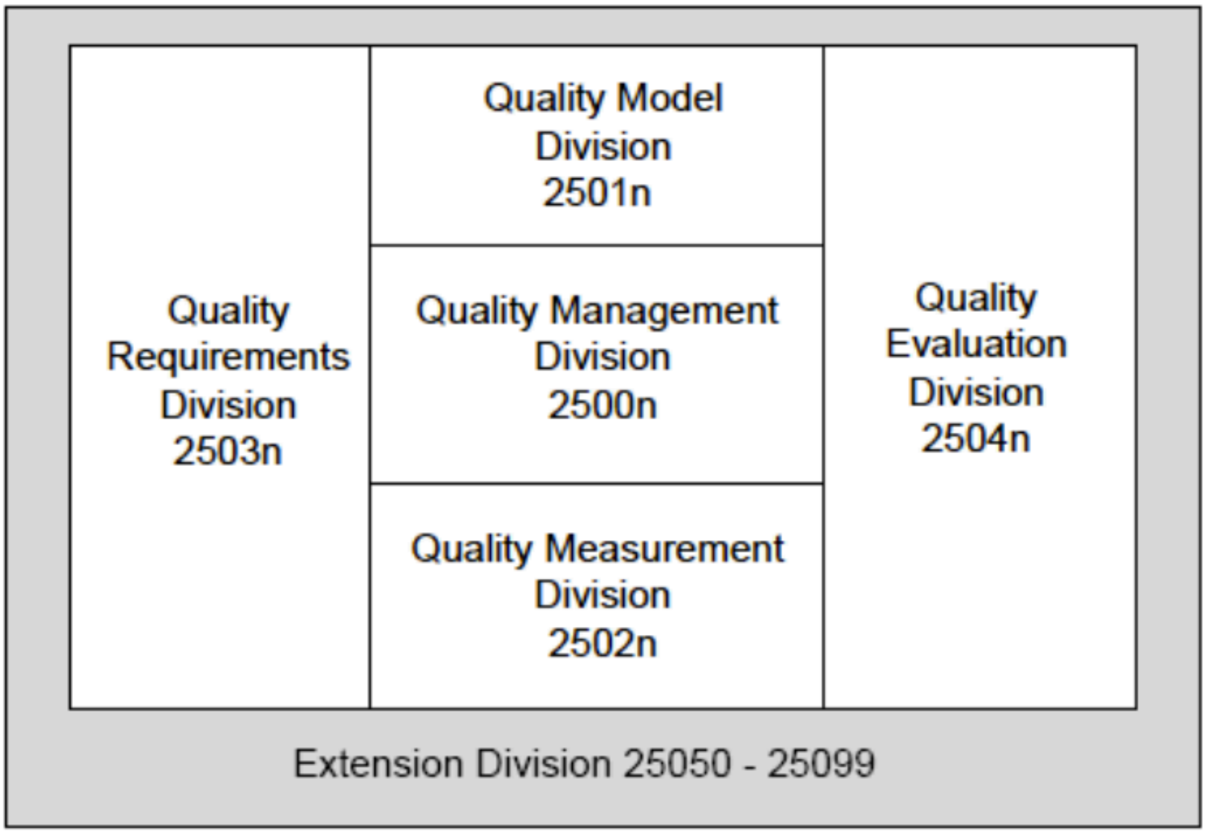
Why is not quality one of the vertices? Simple, it is in the best interest of all stakeholders and development team to achieve a quality product.

The reasons may seem very obvious: organizations, whether are of commercial nature, academia or any other field, are directly affected by the information systems that operation within their processes.

As explained in (Příbrský, Kvantifikovaný přístup k jakosti informačního zabezpečení pro podporu evaluace informačních technologií, 2012) information systems affect directly and/or indirectly all the processes of any organization. Operational processes are usually controlled or even completely automated by technological tools that manage all the data and workflows involved.

Administrative and/or managerial processes are equally affected by information system within a business. Communications are digital and reporting or decision tools are used on a daily basis.

Keeping in mind the strong relationship between IS and business process, there is no question that organizations require high quality tools to guarantee their daily operation and satisfactory results. Regardless the nature of the business or operation, every product owners expects to have a reliable


Quality Model Division 2501n
Quality Requirements Division 2503n
Quality Management Division 2500n
Quality Evaluation Division 2504n
Quality Measurement Division 2502n
Extension Division 25050 - 25099

technological solution that allows them to improve the response to their clients and provide full support to employees (users) or operators tasks.

Fig. 2. Organization of SQuaRE series of International Standards

What does it define a product project quality?

ISO is highly recognized for their purpose to guarantee quality in process, services and products. The organization publishes standards for almost everything including software.

The SQuaRE model also known as ISO/IEC 25010:2011 (International Standard Organization (ISO), 2011) defines a framework for assuring quality in computer systems and software in use.

The standard is divided in five sections to cover all the areas related to the production and use of software:

1. Quality Requirements: Covers de definition of functional and non-functional quality requirements based on quality models and quality measures.
2. Quality Management: This section supports the definition of conditions and models for the management of the requirements, specification and evaluation of software product quality.
3. Quality Model: Presents models for quality software, in terms of use, internal and external structure.
4. Quality Measurement: Defines quality measures for software and its use. Provides guidance for the use of quality measure elements.
5. Quality Evaluation: This section aims to support the evaluation of the quality software. It provides guidance and requirements necessary for the evaluation of the quality. Its intention is to help the evaluation actions of developers, testers and users.

According to the standard the quality of a product is defined by the following characteristics who defines set of attributes for each category:

- Functionality: Set of attributes related to the functions and their specific properties. These functions response to certain needs. Among the attributes regarding functionality we can find: Accuracy, security, interoperability, functionality compliance and others.

- Reliability: Set of attributes related to the stability of the software within certain environment for an expected period of time. Among these attributes we find: Maturity, fault tolerance, recoverability and reliability compliance.

- Usability: Set of attributes that evaluate the effort needed to use the software. These attributes are sourced by final user point of view in terms of: Understandability, Operability, Learnability and other aspects related to the user interaction with the software.

- Efficiency: Set of attributes regarding the relation between the performance level of the software and the resources needed. This section focused in time and resources behavior.

- Maintainability: This set of attributes refers to the elements that evaluate the effort required for extending, amending and modifying the software. Among these elements we find: Analyzability, Changeability, Stability, Testability.

- Portability: ISO in its revision: ISO/IEC 9126-1: 2001 incorporates the same software quality characteristics with some amendments. Portability has been split into transferability and compatibility (including interoperability)

- Transferability: These attributes refer to the ability to transfer the software from one environment to another one. Among the attributes we find: Adaptability, Installability and transferability compliance.

- Compatibility: These attributes are relates to capacity of one or more software components to share the same environment with efficiency. The attributes evaluated in this section are coexistence and interoperability.

- Security: This section covers the aspects related to the protection of the operability of the system and its data. Among the attributes we find: Confidentiality, integrity, authenticity, accountability and others.

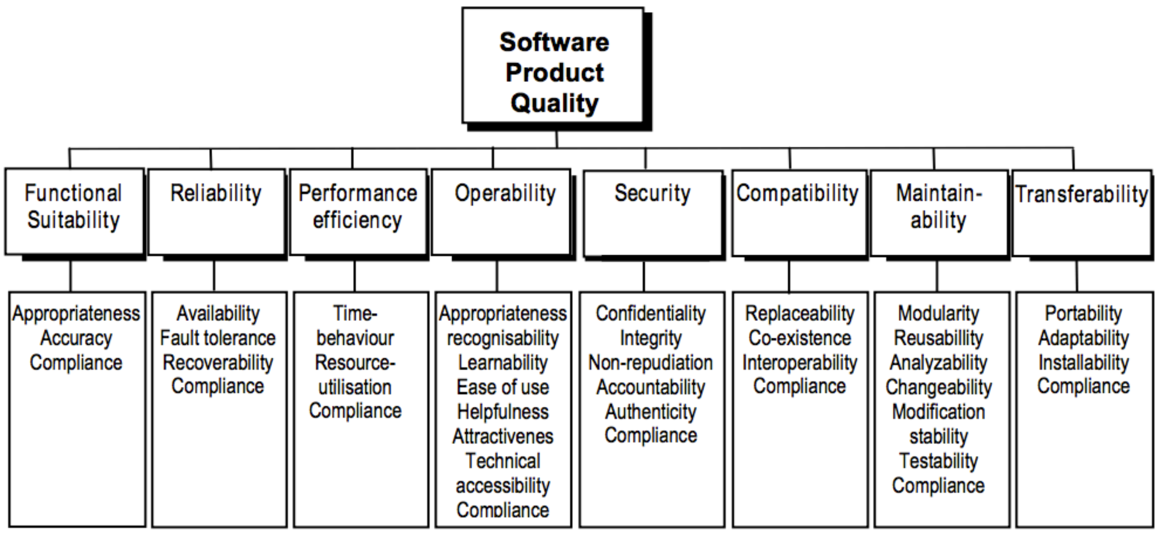

Software Product Quality
Functional Suitability
Reliability
Performance efficiency
Operability
Security
Compatibility
Maintain-ability
Transferability
Appropriateness Accuracy Compliance
Availability Fault tolerance Recoverability Compliance
Time-behaviour Resource-utilisation Compliance
Appropriateness recognisability Learnability Ease of use Helpfulness Attractivenes Technical accessibility Compliance
Confidentiality Integrity Non-repudiation Accountability Authenticity Compliance
Replaceability Co-existence Interoperability Compliance
Modularity Reusabillity Analyzability Changeability Modification stability Testability Compliance
Portability Adaptability Installability Compliance

Fig. 3. Software quality model categories and attributes

4.1.2 Risk management

Risk management embraces all the processes related to the identification, analysis and mitigation of dangerous events, known as risks that may affect the expected results of a project. (Project Management Institute PMI, 2013)

Risk management is not an independent discipline; it is a part of the whole project management strategy. As a consequence, risk management processes are directly connected with the additional processes of the project development.

As we have mentioned before, the key steps to be developed for risk management are identification and response.

These significant steps have subsidiary steps that may vary according to the situation or business. According to (Boehm B. , 1991) risk management main processes are more specified and imply: identification of the risks, analysis of risks to evaluate and prioritize them according to their possible occurrence and impact, planning of mitigation and contingency actions, tracking of risks as the project unfolds, and control of risk responses.

Barry W Boehm known as the father of the software risk management proposed a model in 1991 which distinguish two principal steps, each one divided in sub-steps as it follows risk assessment, which implies risk identification, analysis, prioritization; and risk control, which implies risk management planning, solution and monitoring.

The types of risks considered by Boehm are: personnel shortcomings, unrealistic schedules and budgets, wrong development, extra development, unstable requirements, shortfalls in externally components and tasks, and forced computer-science capabilities. He also proposed a scale for the probability of occurrence of risks and according to this scale; the impact of the risk can be evaluated (Boehm B. , 1991)

Another valuable risk management model is the one proposed by The Software Engineering Institute (SEI); they defined a framework (Higuera & Haimes, 1996) based on three groups of practices: software risk evaluation, based on three groups of practices: software risk evaluation, continuous risk management and risk management team. SEI has defined a risk taxonomy, which classifies a risk into several categories: requirements risks, design risks, coding and testing risks, contract risks and resources risks. What distinguishes this approach from the others is the team risk management, which defines methodologies, processes and tools for developing working relationships between customers and suppliers. As a summary, we can say that the most fundamental aspects of risk management that should be present in any project are: risk identification, analysis of occurrence, and measurement of the negative effect in case of occurrence and plan of a mitigation plan. A minimal of documentation is required in order to avoid improvised response actions.

In my opinion risk management practices are natural and therefore seem to be easy to implement. For some industries it is crucial to follow risk driven processes. However, software industry still finds challenging the implementation of these processes. Mainly because there is still the idea that quality cost is higher since it requires additional resources, people and time.

In the previous paragraphs we have presented the general definition of risk management from the perspective of project management. Following the initial motivation of this work, it is necessary to analyze the elements of risk management and its potential integration with the agile practices.

4.1.3 Practices and tools for quality assurance

The aim of this section is to explore some of the most well known quality practices. These methods and/or tools have been applied in the software industry to classic development projects. The idea in order to achieve my goals is to identify the elements of these techniques that may coexist with agile software development methods.

The idea of all the tools and techniques presented here is to identify possible risk that may occur during the project in order to plan activities that could mitigate the unexpected effects that could affect the project/product.

CMMI (Capability Maturity Model Integration)

CMMI believes that "early and aggressive" detection is necessary if the stakeholders want to avoid extra effort and costs. This is the natural motivation for risk management, the earlier the better.

CMMI divides the risk management activities in 3 parts: 1) Definition of risk management strategy, 2) Identification and analysis of risks and 3) Implementation of risk mitigation plans if required. The focus area of CMMI is the project; however they specified that it could be applied to organizational risks.

CMMI recognizes the inherit risk management practices in Agile methods:

"In Agile environments, some risk management activities are inherently embedded in the agile method used. For example, some technical risks can be addressed by encouraging experimentation (early “failures”) or by executing a “spike” outside of the routine iteration” (CMMI Product Team , 2010)

However, the SEI insists that the risk management approach is more systematic but could be implemented in the life cycle of an agile method. CMMI for development does not specify in detail how this integration could be possible.

In (Glazer H. D., 2008) there is a comparison there is a comparison of CMMI and Agile methods and the complementary aspects of both approaches. An interesting call for action is done for both specialists in CMMI and Agile to improve the approach in areas that the other is strong. The main problem addressed in CMMI is its generality or neutrality. It can be applied to almost any situation that involves software development. The information provided by the SEI is sometimes confusing depending on the level of expertise and perception of the receiver.

The situations identified where agile experts should learn from CMMI are many, for my purpose is important to point out risk management, measurement and analysis. As Glazer mentioned:

“CMMI provides a path for the effective use of processes, measurement, training, and improvement.”

This is exactly the key to quality assurance in a project. In my opinion these four activities (Effective use of processes, measurement, training and improvement) are the core of successful projects.

CMMI for development proposes three groups of activities for risk management as follows (CMMI Product Team , 2010):

Prepare for Risk management, this first group's purpose is to establish a risk management strategy that will define the steps to follow in order to identify, categorize, evaluate and control risks. The activities in this group are:

1. Determine risk sources and categories: These sources could be internal or external. The typical categories could be risks related to project phases, technical performance or types of products. A deliverable of this activity could be the risk taxonomy.

2. Define risk parameters: In this activity the idea is to identify the parameters that will be used to evaluate, qualify and prioritize risks. One technique that could be used here is FMEA (Failure mode effect analysis) (Banerjee, 1995).

3. Establish the risk management strategy: Final activity and core of this group. This strategy includes among other elements, the methods and tools to be used for risk management main processes. Also includes the classification and categorization of risks as well as the mitigation techniques to be used and the risk measures.

Identify and Analyze Risks, is the second group of activities proposed by the CMMI. The aim of this group is to determine the importance of risks identified. Therefore, the activities are to 1) identify and 2) prioritize risks.

The methods suggested by CMMI to identify risks include examination of each element of the project work breakdown structure, consult experts, examine projects related or for similar products where risk management activities have been conducted.

CMMI proposes sub practices for identifying risks. It is necessary to perform a review of environmental elements that affect the project, as well as the elements of the project plan and the work breakdown structure. The objective is to identify risk related to cost, schedule, requirement and performance. As a last step all the risks should documented. This should include the surrounding conditions, context, stakeholder associated and consequences for each risk.

In order to evaluate and prioritize risks, it is necessary to perform risk analysis. The evaluation process proposed by CMMI includes the measurement of likelihood, impact and severity of each risk. This method is equivalent to the Risk Priority Number calculation defined by FMEA explained before.

The third group of activities in CMMI is *Mitigate Risks* which objective is to reduce the negative impact that risks could have on the expected results of the project. For this purpose it is necessary to establish for which risk it is worth a mitigation action, this is decided after assigning levels of tolerance to each risk. These risks selected are included in the mitigation plan and for each of them should be assigned a responsible person and a mitigation activity depending of the type of risk. The possible mitigation activities are avoidance, control, transfer, monitoring or acceptance (not taking action)

The implementation of the mitigation plan is a continuous activity during the project. The risk should be monitored periodically and if necessary the risk mitigation plan should be updated.

Zachman framework is an extensive framework for enterprise architecture (Zachman J. , 2008) The framework is a combination of natural classifications. The intention is to combine the answers to interrogatives What, How, When, Who, Where and Why. It also includes a second classification postulated by ancient Greek philosophy: Identification, Definition, Representation, Specification, Configuration and Instantiation.

The author of the framework insists that it is not a methodology but a "structure whereas a methodology is a process" The structure provides definition, classification not a set of steps to follow as a process does.

The first published version of the model included only 3 columns: Data Function and Network. It is important to note that the Zachman framework uses diagrams like Chen Diagram, Bachman diagram and IMS-Root Segment diagram for each descriptive representation. Each of these representations corresponds to a cell, result of columns that represent the Interrogatives and the rows that represent the Transformations). The intention of Zachman is to use primitive models to describe each aspect of the enterprise.

Since the first version of the Zachman framework in 1987, there have been eight releases. Zachman updated the framework to guarantee the use of only primitive models in the representations. Also additional columns were added to represent responsibility, timing and motivation; aspects related to enterprise architecture. Please note that initially the framework was conceived only for

Information Systems Architecture. Also there was a slight change in the column Network (Where) to make the framework international, initially the framework supported only U.S.

The current version 3.0 is a 6x6 matrix that contains the 36 concepts to describe almost anything. It offers the possibility to look from six different perspectives represented by each row: Executive, Business Mgmt., Architect, Engineer, Technician and User perspective. Besides a dramatic change in the graphics, the last version of the framework changes its name adding a subtitle: The enterprise ontology. There are already many frameworks for enterprises that describe methodologies based on the Zachman framework. This situation made the academics to suggest a new name that entitles the framework as ontology (Zachman J. , 2011).

The figure 4. shows the current version of the Zachman framework: The enterprise ontology.

Even all the updates done in the past 28 years the original Zachman theory remains.

"All descriptive representations can be expressed in terms of Things and Relationships"[1]

The logic also remains; the schema is bi-dimensional interrogatives against transformations. And in each cell there is a primitive model that represents the enterprise models' words.

[2] K. Beck, Extreme Programming Explained. 2nd ed. 2005. p.42

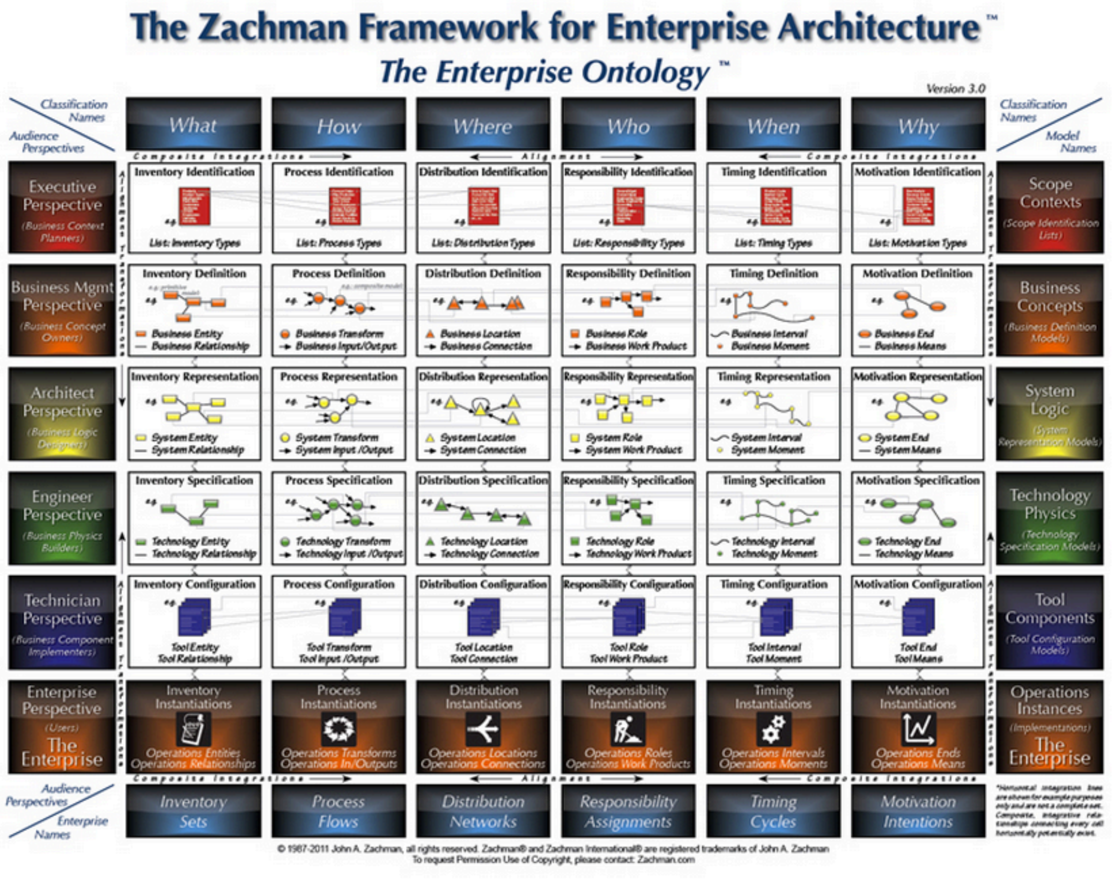


Fig. 4. The Zachman framework

PMBOK (Project Management Book of Knowledge) – includes two chapters dedicated to quality assurance: Quality Management and Risk Management (Project Management Institute PMI, 2013). The PMBOK published by the Project Management Institute is a complete guideline for project management of any nature. It is not strictly connected to software development, however it suits many aspects of a software project in general.

Quality management chapter introduces the processes related to planning, performing and controlling quality assurance. Some of the processes covered in PMBOK are explained in more detailed in this document (e.g. Six Sigma, ISO standards) The PMI analyzes quality management for both, the project and the final product. Any issue that affects any of these two aspects may have serious consequences for any of the stakeholders.

As for any of the knowledge areas described in the PMBOK, quality management defines a group of processes. Each process is formed by a set of inputs, tools and outputs.

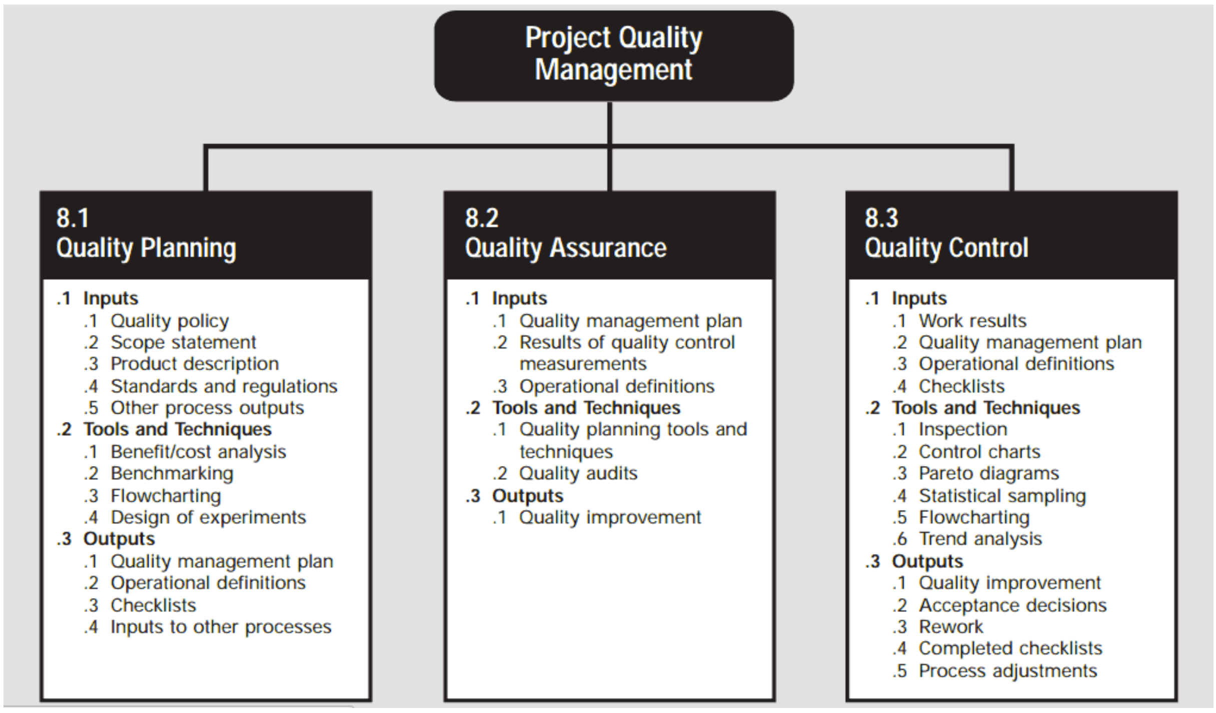


Fig. 5. Project Quality Management Overview

- Quality Planning: PMBOK establishes that every project should include plan quality. The aim is to identify the quality standards and regulations relevant to the project. The inputs for this process are: documents that describe the quality policy of the organization, the scope of the project and the relevant standards that may apply to the final product. The main output of the quality planning is the quality management plan, which describes the implementation of the quality policy and quality standards. This document is usually highly detailed and should be part or an input for the general project plan.
- Quality Assurance: Refers to the execution of all the planned tasks to guarantee the project and the final product will comply with the quality standards. This process uses different tools and techniques that should be

performed early in the project. Some of these techniques are: Cause and Effect diagram, Pareto chart, Statistical sampling. The tools to be used and the results to be measure are generally responsibility of the project management team. The output of this process is quality action, which refers to any possible improvement, change or action to take in order to guarantee efficiency in the project.

- Quality Control: This last process described in the chapter of quality assurance in the PMBOK, refers to the control measures to take over the quality activities planned and their results. This process has two objectives: to identify possible failures in the process or the product quality to suggest possible actions and to validate that all the quality standards have been met and the output comply with the requirements of the stakeholders. The inputs for this process include the project management plan that as mentioned before, the quality plan is also part of it. Other documents that include quality metrics, checklists and product specifications and deliverables, are also used as input for quality control. The output of the quality control is formed by documents that present and validate the results of the project and quality activities. Change requests can also be output of this process in case there is a quality requirement not met that demands an action. All the updates affect directly the project plan; therefore an update of this document is also expected as an output of this process.

Risk management chapter (Project Management Institute PMI, 2013) introduces the processes related to planning, measure and controlling possible risks in a project. The processes included in this section have as goal to decrease the impact of possible failures y identifying them prior to occurrence and planning response actions in case it is needed. The figure 6 shows the detailed structured proposed by PMI for risk management processes. As described in quality management chapter, risk management processes also are conformed by a set of inputs, tool and outputs.

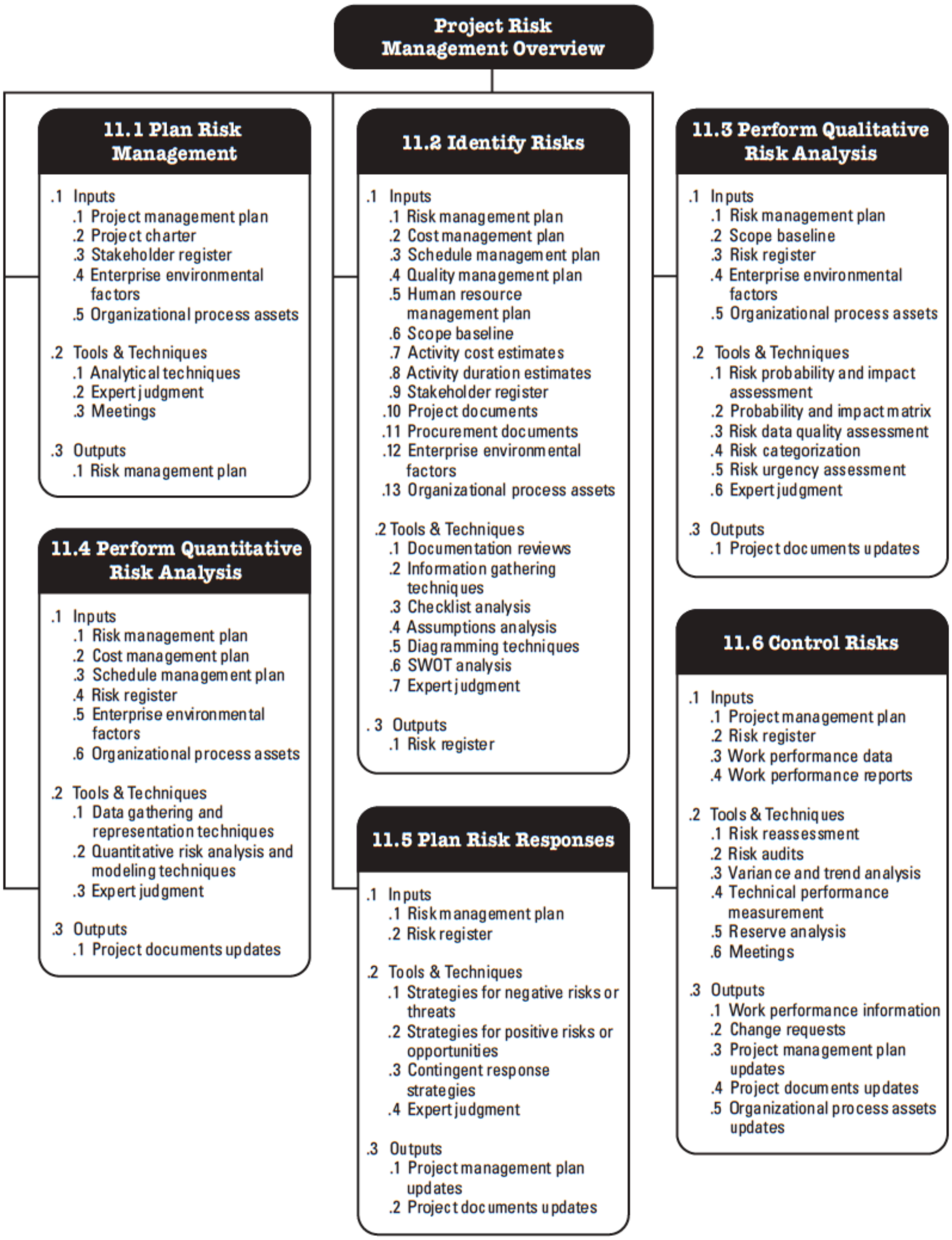

Project Risk Management Overview
11.1 Plan Risk Management
.1 Inputs
.1 Project management plan
.2 Project charter
.3 Stakeholder register
.4 Enterprise environmental factors
.5 Organizational process assets
.2 Tools & Techniques
.1 Analytical techniques
.2 Expert judgment
.3 Meetings
.3 Outputs
.1 Risk management plan
11.2 Identify Risks
.1 Inputs
.1 Risk management plan
.2 Cost management plan
.3 Schedule management plan
.4 Quality management plan
.5 Human resource management plan
.6 Scope baseline
.7 Activity cost estimates
.8 Activity duration estimates
.9 Stakeholder register
.10 Project documents
.11 Procurement documents
.12 Enterprise environmental factors
.13 Organizational process assets
.2 Tools & Techniques
.1 Documentation reviews
.2 Information gathering techniques
.3 Checklist analysis
.4 Assumptions analysis
.5 Diagramming techniques
.6 SWOT analysis
.7 Expert judgment
.3 Outputs
.1 Risk register
11.3 Perform Qualitative Risk Analysis
.1 Inputs
.1 Risk management plan
.2 Scope baseline
.3 Risk register
.4 Enterprise environmental factors
.5 Organizational process assets
.2 Tools & Techniques
.1 Risk probability and impact assessment
.2 Probability and impact matrix
.3 Risk data quality assessment
.4 Risk categorization
.5 Risk urgency assessment
.6 Expert judgment
.3 Outputs
.1 Project documents updates
11.4 Perform Quantitative Risk Analysis
.1 Inputs
.1 Risk management plan
.2 Cost management plan
.3 Schedule management plan
.4 Risk register
.5 Enterprise environmental factors
.6 Organizational process assets
.2 Tools & Techniques
.1 Data gathering and representation techniques
.2 Quantitative risk analysis and modeling techniques
.3 Expert judgment
.3 Outputs
.1 Project documents updates
11.5 Plan Risk Responses
.1 Inputs
.1 Risk management plan
.2 Risk register
.2 Tools & Techniques
.1 Strategies for negative risks or threats
.2 Strategies for positive risks or opportunities
.3 Contingent response strategies
.4 Expert judgment
.3 Outputs
.1 Project management plan updates
.2 Project documents updates
11.6 Control Risks
.1 Inputs
.1 Project management plan
.2 Risk register
.3 Work performance data
.4 Work performance reports
.2 Tools & Techniques
.1 Risk reassessment
.2 Risk audits
.3 Variance and trend analysis
.4 Technical performance measurement
.5 Reserve analysis
.6 Meetings
.3 Outputs
.1 Work performance information
.2 Change requests
.3 Project management plan updates
.4 Project documents updates
.5 Organizational process assets updates

Fig. 6. Project Risk Management Overview

The processes of this knowledge area covered the same mentioned previously in section 2.1.2.

- Plan Risk Management: This process defines how risk management will be implemented in the project. The output is the risk management plan, which contains all the tasks and responsibilities of the risk management team and the project manager.
- Identify Risks: This process aim is to list the possible risk that may affect the project and describe their behavior. The tools used for tis process include informal techniques like brainstorming and expert judgment as well as some more formal like diagramming tools and SWOT analysis.
- Perform Qualitative Risk Analysis: This process analyzes the probability of occurrence and impact of the risk in order to prioritize them for further action. The tools used in this process
- Perform Quantitative Risk Analysis: This method is based on the prioritization in the previous process. The aim of this process is to quantify the effect of the already risks identified. For this purpose the project manager should use the documents that describe the project plan, budget and schedule, as well as the risks register and other documents that describe the environmental conditions of the project. The output of this process represents updates in the project documents with probabilistic analysis and risk priority updates. The methods used in this process depend on which and how much data is available. Graphical representations, probabilistic analysis and modeling tools can be used. However, in case of lack of data, expert judgment is the tool to be used to analyze the possible quantitative effect of each risk.
- Plan Risk Response: The goal is to prepare response action for each risk identified. The process should help to assign a strategy of response to each risk depending on the priority and the quantitative effect already calculated in the previous process. The PMBOK considers not only the negative risks but also the positive ones. The strategies of response for

negative risks include: Avoid, Transfer, Mitigate and Accept. While the responses for the positive risks could be: Enhance, Exploit, Share and Accept. Generally the suggested tool for this analysis is expert judgment. The output of this process represents several updates in the project documents. Depending on the action planned; the scope, the schedule and the allocation of resources would need adjustment.

- Control Risk: This process has many sub-processes: tracking identified risks, implementing the planned response action, identifying new risks and evaluating the risk plan. The inputs for this process include the risk plan and register as well as the work performance reports. The methods used for these processes include analysis of performance, risk audits, periodical meetings and risk reassessment. The general the output of this phase are the different updates to the project documents involving: change requests, project plan updates, risk register and response plan updates.

The PMBOK covers all the phases of risk management and clarify the different activities that should be implemented. This approach suggest and extensive use of documentation, planning and analysis. Depending on the project and its size, the PMBOK suggestion may demand a lot of resources for risk management activities.

Six (Six Sigma) is a business management strategy which aim is to increase at maximum the percentage of defect-free products. Motorola originated this strategy and is in continuous development and research by the Motorola University.

Six sigma considers a defect any output that does not meet the customer specifications; just exactly as the other methods I have been studying. The focus of this strategy is to achieve quantifiable financial return from any project. Six sigma defines a special structure of roles: Executive leader (CEO), Champion, Master Black Belt, Black Belt, Green Belt and Yellow Belt. Each of these roles is responsible for the implementation and execution of six sigma. Especially Black belts who should devote 100% of their time for six sigma project execution.

The origin of Sig sigma is statistical, is based on a standard deviation. The idea is to reduce it till the point where the product is within the limits of customer requirements. A sigma level measure the level of efficiency, level 6 is equal to 99.999966%, which means that from a million units 3.4 will have defects.

Six sigma follows two types of methodologies: DMAIC which aim are existing business process and DMADV used for new product development.

Each letter of these two acronyms represents one of the phases of the methodology: (DMAIC) Define the project requirements, Measure key aspect of the current process, Analyze the existing data to find out the root-cause of defects, Improve the current process, and Control to ensure that defects will be prevented before happening. (DMADV) Define the design according to the customer expectations, Measure parameters for design and CTQ (Critical to Quality), Analyze to create a feasible high-level design, Design based on the data collected, Verify the design and implement the production process.

Six sigma uses additional quality management tools as the ones we have explained before, FMEA, 5 Whys, Root-cause analysis, etc.

FMEA (Failure Mode and Effect Analysis) is used to identify potential failures within a system, evaluating their effects, which mean to rank their severity and occurrence. The purpose is to recommend possible actions to prevent these failures from reaching the customer/user. (Stamatis, 2003)

A failure is considered any error or defect in any part of the system, which affects the customer. The effects are the consequences of a failure during the operation of the product.

Severity is defined according to the harm produced to the customer or the seriousness of the effect on the functionality. There is a correlation between effect and severity; if the effect is critical then severity is high and vice versa.

The process FMEA is evolutionary and includes application of several technologies and methods. The aim is a quality product with the minimum of failures, prioritizing the customer requirements; partly the reason of agile methodologies as well.

Severity (SEV): The first step in a risk analysis is to quantify the severity of the effects; they are evaluated on a scale of 1 to 10 with 10 being most severe. The ranking is shown in Table 1 (Stamatis, 2003).

RANK	DESCRIPTION
Dangerous (10)	Failure affects safety or government regulations with alarm
Very High (8)	The product is inoperable with loss of primary function.
High (7)	The product is operable, but at the reduced level of performance
Moderate (6)	The product is operable, but the item (s) from the comfort or convenience is inoperable.
Low (5)	The product is operable at a reduced level of operation.
Very low (4)	Most customers notice failures
Minor relevancy (3)	Minor customers notice the defects.
Very minor relevancy (2)	Demanding customers notice failures.
None (1)	No effect

Table 1. FMEA-Risk Severity Ranking

Occurrence: Represents a remote likelihood that customers experience the failure effect. The Table 2 defines the value of the occurrence, where intermediate values are assumed to obtain immediate superior, and if it is ignored failure probability must assume an occurrence is equal to 10.

Probability of occurrence	Percentage of failure	Rank
Very high: Failure is almost unavoidable.	1 in 2 ≥	10
	1 in 3	9
High: repetitive incidents	1 in 8	8
	1 in 20	7
Moderate: occasional incidents	1 in 80	6
	1 in 400	5
	1 in 2000	4
Low: Relatively few incidents	1 in 15.000	3
	1 in 150.000	2
Remote: The incident is unlikely	1 in 1.500.000≤	1

Table 2. FMEA-Occurrence criteria

Detection: Is the rank corresponding to the probability that the current control will detect causes of failure modes before the product leaves the manufacturing area. It's very important not assume low probabilities just because the occurrence is low; these two rankings may or not may be correlated. Complete ranking is shown in Table 3.

Rank	Description
Very High (1)	Remote possibility that the product will be delivered. The defect is functionally obvious and detected
High (2-5)	The defect is obvious identified
Moderate (6-8)	The defect is easily detected
Low (9)	High likelihood that the product would be delivered with the defect
Very low (10)	Item is usually not checked and will be delivered with the defect

Table 3. FMEA-Detection ranking

Risk Priority Number: Known as RPN, defines the priority of the failure. In FMEA the goal is always to reduce RPN through a reduction in severity, occurrence and detection. The risk priority number (RPN) is the mathematical product of the severity, occurrence and detection:

$$\mathbf{RPN = S * O * D}$$

Recommended action: There is no point to do FMEA analysis without a recommended action.

Typical recommendations may be:

- No action at this time (Tolerate)
- Add built-in detection devices (Increase detection or predictability)
- Provide alternatives to the design (Avoid before occurrence)
- Add a redundant subsystem (Tolerate with Action)
- Response action to effect (Mitigation)

FMEA in software development

Even that FMEA was originally created for assessing risk related to hardware, there are several studies that confirm its use in software development (Banerjee, 1995) (Bicchierai , Bucci, Nocentini, & Vicario, 2012) (Lauritsen & Stålhane, 2005)

In Lauritsen they propose to use FMEA in the agile development. They specify two types of FMEA: Functional and Detailed. Functional FMEA refers to requirements definition phase. Detailed FMEA is used between the design and coding activities. The disadvantage of this proposal is the addition of extra activities to the workflow, instead of integrating the FMEA concepts within the current workflow. This may seems as lack of agility in this proposal. The advantage is the potential use of the FMEA results to easily create test cases.

Banerjee became the base reference of the FMEA in software development. This paper concludes that FMEA brings several advantages to the development process, mainly accurate effort estimation and quality assurance.

4.2 Agile software development

My intention is to introduce the concept of agile software focusing on the elements that are base for the development of this thesis and the ones we consider as the most relevant to the achievement of the goals established above.

In 2001, the agile manifesto became the official start of a new age of development that would try to improve software development. The agile manifesto consists of a set of principles that define a new approach of development with different priorities, as it is customer satisfaction over strict planning. The manifesto specifies the capability of adjusting to continuous changes, which may affect the whole project or just a requirement (Awad, 2005)

Agile manifesto requires high quality staff; project managers, designers, developers and even customers should have skills that allow producing rapid quality deliveries and reducing the time spent on planning and documentation.

Agile methodologies arose from the need for a faster response to client satisfaction, which includes late changes acceptance and tangible results as early as possible.

There are many different agile methods, which promote the general principles of the agile manifesto. Most of them divide the tasks in small groups known as iterations. Each iteration will be following the main phases of classical software development, but in a shorter time. Most popular agile software development methods are Scrum, Crystal, Dynamic System Development, Feature Driven Development and Extreme programming.

Despite the popularity of all these methods and some others, there are still some critics related to agile practices that may compromise the success of a project. The biggest limitation of agile methodologies is their implementation in large development groups (Turk, France, & Rumpe, 2002). The rate of success of lightweight methodologies in groups of more than 20 developers is still remarkably low.

Another limitation of agile methodologies is the considerable reduction of documentation. Agile methodologies required more time for coding than for planning or documenting. This may result in faults that could be hard to identify by external reviewers. The other problem not having a standard method of communication, documentation, is the misunderstanding that could take place during the project, due to the different points of view and perception that stakeholders may have.

Clear and precise communication is the key of success of any project. In agile development project communications is a crucial point as there are no standards that control the flow of information. Preferred method of communication in agile development is face-to-face meetings. This makes agile practices, not a suitable option for teams, which are distributed; this situation has become more common in the last years.

In my opinion, agile methodologies require a group of characteristics that depend much more on people than on processes or methods. Therefore, the staff required for an agile project has to be exceptional.

Agile development methodologies recognize people as the drivers of project success, counting with skillful people. It is necessary to point out that, when the success of a project depends on a person's behavior, the risks are bigger and with a higher probability of occurrence.

4.2.1 SCRUM

Scrum is defined as a framework to agile software development in order to reach a common goal (final product). The framework works very well together with other methodologies as well. (Sutherland & Schwaber, 2011)

Scrum challenges the concepts of classical development as waterfall. Scrum is actually the most popular agile approach for software development.

The basic principles of Scrum are: transparency, inspection and adaption.

- *Transparency:* Specifies a common language and share status of the project among the participants.
- *Inspection:* Regular inspection by skillful participants in order to identify unexpected variances.
- *Adaption:* As mentioned before, one of the principles of agile software development is the ability to adaption and high tolerance to changes. Scrum follows this principle, specifying that if inspection shows a probability of unacceptable results, the current process should be adjusted.

Scrum defines a set of roles, artifacts and meetings that are integrated in the core of the methodology, the Sprint. The Sprint is defined as a period of time where the scrum team should focus on a group of tasks to be developed. Everyone involved or affected by the project is identified with a role, as it is product owner, development team, scrum master and stakeholders. Some authors may consider manager also a role in scrum projects.

Scrum also defines a set of events that are used to guarantee the implementation of the three principles mentioned before. These events are:

- Sprint Planning Meeting
- Daily Scrum
- Sprint Review
- Sprint Retrospective

Sprint: As mentioned before this event is the core of the Scrum methodology. It is time limited and a release increment is expected as a result. The scope of the Sprint as well as the participants, are defined at the beginning of the Sprint and remains the same till the end. If changes are required in the product, the new requirements are reserve for the next Sprint. During the Sprint planning meeting it is define the work to be developed during the Sprint. Also, during this meeting it is decide how the functionalities chosen will be developed. The functionalities may be split in tasks and assigned to the developers. If the team considers that the expected results will not be met on time, the functionalities may be negotiated with the project owner and move to the next Sprint.

Daily Scrum: This event happens everyday of a Sprint. It is a meeting schedule for only 15 min with the development team. The goal of this meeting is to plan the work for the next 24 hours and identify possible issue that may affect the plan schedule. The responsible person for calling the meeting is the Scrum Master however the entire development team is responsible for conducting the meeting. Basically, every person should answer three questions during the meeting:

- What was done?
- What will be done?
- Are there any open issues/obstacles?

The daily scrum helps to increase the possibility of achieving the Sprint goal. The team is involved in the progress of the planned release increment and informed about the status every day.

Sprint Review: During this meeting the development team and the product owner discusses the increment delivered. The development team informs and shows what has been done. The backlog is discussed in order to track progress and identify future work. Therefore, the output of this meeting is the update of the backlog and a potential list of functionalities to include in the next Sprint.

Sprint Retrospective: This meeting takes place after the Sprint review and it's an internal evaluation of team performance during the last Sprint. The Scrum Master and the development team identify opportunities of improvement in terms of processes, communications and tools.

The figure 7 (Ambler, 2008) shows the life cycle of Scrum. In this graph we can see when each of the events explained above happen.

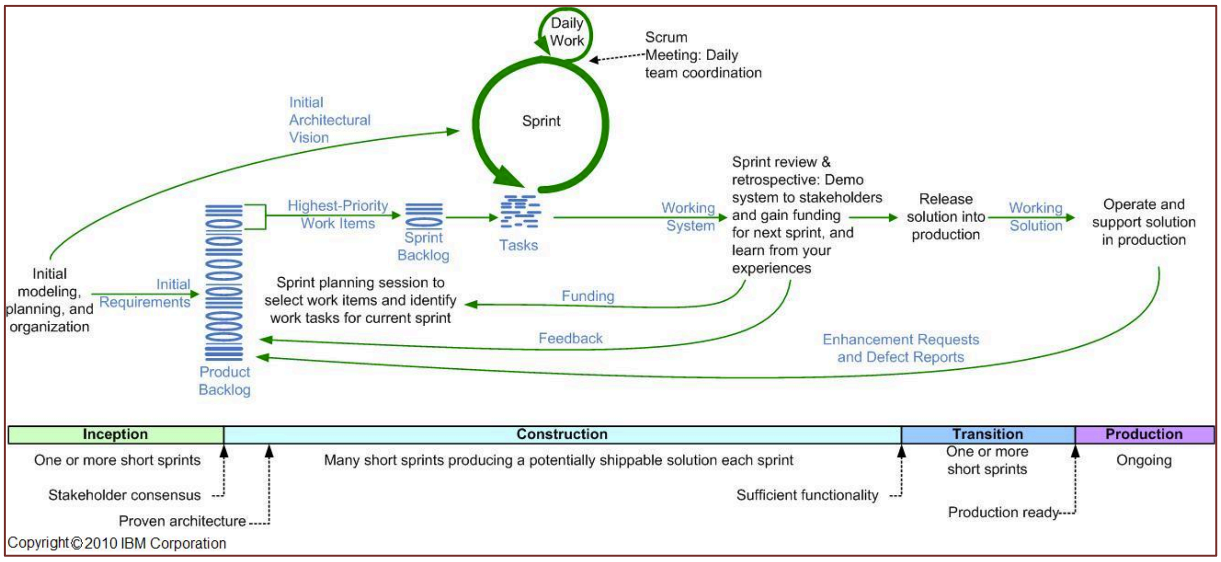

Daily Work
Scrum Meeting: Daily team coordination
Initial Architectural Vision
Sprint
Highest-Priority Work Items
Sprint Backlog
Tasks
Working System
Sprint review & retrospective: Demo system to stakeholders and gain funding for next sprint, and learn from your experiences
Release solution into production
Working Solution
Operate and support solution in production
Initial modeling, planning, and organization
Initial Requirements
Sprint planning session to select work items and identify work tasks for current sprint
Funding
Feedback
Enhancement Requests and Defect Reports
Product Backlog
Inception
Construction
Transition
Production
One or more short sprints
Many short sprints producing a potentially shippable solution each sprint
One or more short sprints
Ongoing
Stakeholder consensus
Proven architecture
Sufficient functionality
Production ready
Copyright©2010 IBM Corporation

Fig. 7. SCRUM Life Cycle

Scrum and risk management

None of these roles is explicitly responsible for any risk management activity (identification, measure or control). However, we would like to analyze how their defined functions may include or guarantee the implementation of risk management in the projects.

The product owner is responsible for the development of the product backlog, which contains the full list of functions/requirements that should be included in the software in development. He / She should guarantee that every member of the development team is aware of the product backlog and understands it. The product owner is also responsible for organizing this backlog in order to achieve the goals according to the plan and the agreement with stakeholders.

One of the functions of the product owner is to turn the issues in the backlog into features to be developed (Pekka, Outi, Jussi, & Juhani, 2010). This action shows an intention of addressing the risks related to the product performance. As explained in (Nelson, Taran, & Hinojosa, 2008) the agile processes tend to be mitigation strategies for risks, however the prioritization of tasks is not always considering the risks that may have been informally identified.

Development team members are responsible for the product development. Nevertheless, the team members participate actively in other processes as backlog updates, effort and time estimations, as well as identifying impediments that may compromise the agile development. Again as with the product owner, the team is not explicitly responsible for identifying, measuring or controlling risks. Even though, it could be considered the intention of identifying impediments as an informal task to list risks.

The Scrum Master is the person responsible for "cleaning the path" for the development team. His / Her function is to address all the impediments that may affect the labor of the development team, jeopardize the goals and therefore, the success of the project. This interesting role approximates the process of risk mitigation, considering that the impediments have been identified previously informally and correspond to the definition of risk.

Stakeholders and managers are all the people affected by the project from the client as well as vendor side.

Usually the process that supports the idea that SCRUM is a risk driven methodology is considered only as a mitigation process where risky tasks are prioritized. As explained in (Nelson, Taran, & Hinojosa, 2008) risks are not tracked or managed explicitly.

4.2.2 Extreme programming

Extreme programming (XP) is one of the methods more used for the agile approach. XP is a lightweight style of programming which purpose is to take all the classic elements of software development to the extreme, starting from the people involved in the project. Kent Beck known as the originator of this methodology considers XP as a social change that focus on the excellent practice of programming techniques communication and teamwork (Beck, Extreme Programming Explained: Embrace Change, 2004)

The most distinguishable difference between XP and other methodologies is the periodicity and size of development cycles. XP works with short cycles and continuous feedback to the user. Behind this difference there are some others that make this approach possible. XP encourages incremental planning and design, which makes the whole plan to evolve during the project. An XP team should trust each other skills and keep a clear oral communication during the whole project. XP projects are open to continuous changes in requirements, due to the short releases and incremental design it becomes less costly to integrate these changes.

XP is based on a set of practices divided in two groups the primary and the corollary practices. In general all the practices are very natural and are an extension of common sense for success. However, the corollary practices are not possible to implement without the full abstraction of the primary practices.

A big part of the primary practices are dedicated to people, motivated and skillful teams develop successful projects. Basic needs of team member can be easily achieved and could have a high positive impact on the flow of the project. Good communication is the key for teamwork; keeping the team sitting together saving the space for each one's privacy allows people to work in a confortable environment counting with each other support if needed.

Taking the previous concept to the "extreme" as XP requires, brings up the idea of pair programming.

"Write all production programs with two people sitting at one machine"[2]

As we say in Spanish: "Dos cabezas piensan mejor que una". Two heads think better than one. This practice is just an extreme of this saying. A noticeable aspect is that pair programming keeps members concentrate on programming. It can be tiring as this couldn't be performed for extensive hours but the level of productivity is higher than programming alone. Here is important to respect each other's ideas and personal space. Pairs should rotate from time to time to promote the interchange of new ideas and make all members be part of the process.

Divide and conquer another well-known saying takes us to the next practice of XP: **User stories.**

XP suggest separating the user requirements into independent user functionalities. A simple sample would be: "Provide the option to generate a zip file containing all the individual task reports for a given project".

Following the team should estimate the time required for development. The stories should be written on separate cards indicating a short title and the time estimated for development. All cards should be placed in a wall visible for the team all the time.

As soon as a considerable group of user stories have been collected, the team can plan the next iteration. XP suggests short cycles, weekly. The week should

[2] K. Beck, Extreme Programming Explained. 2nd ed. 2005. p.42

start by choosing the most important stories according to user interests. Afterwards, the team should write unit tests that will be performed when the stories are developed. Each story can be broken into tasks to be assigned to individuals and make more precise estimations.

Incremental design is an important aspect of XP methodology. The whole team and each member should invest time in design every day. The key is to start with enough design to get going. The same applies for planning, architecture and management. There is one exception and it is for testers, in classic development testing is left at the end. In XP testing should take place as early as possible.

According to Beck (Beck, Extreme Programming Explained: Embrace Change, 2004) the corollary practices are not possible the corollary practices are not possible while the primary practices have not been implemented, assimilated and well adopted by the entire team. Basically primary practices are pre-conditions for the implementation of corollary practices.

Some of the important corollary practices include:

Real customer involvement: Customer should be one more of the team members. Agile manifesto established as a high priority customer satisfaction (Beck, Grenning, Martin, & Beedle, 2001). This is only possible with continuous communication between the team and customer. Regular feedbacks should support the incremental design in order to achieve the customer expected result.

Incremental deployment: As a consequence of an iterative process, there should be incremental deployment. The suggestion is to start with little functionalities are ready to handle and deploy them.

Low index of staff rotation: XP requires a team well formed that can cooperate with good communication and more importantly that trust each other's. This is possible when the teams have enough time to get to know each other and get used to each ones behavior and way of work. That doesn't mean the teams should be static, a reasonable index of rotation is also beneficial for spread knowledge and experience.

Root cause analysis: The idea is to eliminate the defects and also the cause of them. XP process for this situation is to write a test that demonstrates the defect, write a unit test as small as possible to reproduce the defect, fix the system and find out the cause of the defect. Beck mentioned the use of Taiichi Ohno 5 whys exercise finds the defect cause. This simple exercise of asking 5 times "why" could be used for risk identification. We will explore this exercise later in this thesis.

Shrinking teams: As soon as the team is synchronized it will be possible to identify which members' participation is not required for a project. Keeping teams as small as possible contributes to eliminate the waste and form new teams for other projects and increase these members' efficiency.

Daily deployment: New software should be put into production every day. This contributes in two ways; to reduce the gaps between what the user see and what is in the developer's desk. And complies with the XP theory of small and frequent releases of functional software.

Shared code: All members of the team are encouraged to change any part of the code if necessary. For this practice is necessary to have a well-formed team with a sense of collective responsibility. This may reduce the risk of a low truck factor risk of a low truck factor (Torchiano, Ricca, & Marchetoo, 2011)

XP is an extreme implementation of the classic phases of software development. By extreme it refers to extracting the maximum value of each element in the minimum time possible. Each iteration of XP includes all the required phases of software development.

Every iteration starts with a planning meeting where the development team collects the user requirements and defined the user stories. Each user story corresponds to each feature the user desire to have in the system. Each feature is divided into engineering tasks, which should be assigned to a developer. Initial estimation of time is defined and the implementation starts. For each engineering task a unit test is written, when the test is passed the task is considered finished. Afterwards, the user performs test for each user. At the end of the iteration, the programmers deliver functional software that is accepted by the client. A new planning meeting resumes the cycle and all the phases are repeated once again. The figure 8 illustrates the extreme programming general process within one iteration. The length of one XP iteration is between 1 to 3 weeks.

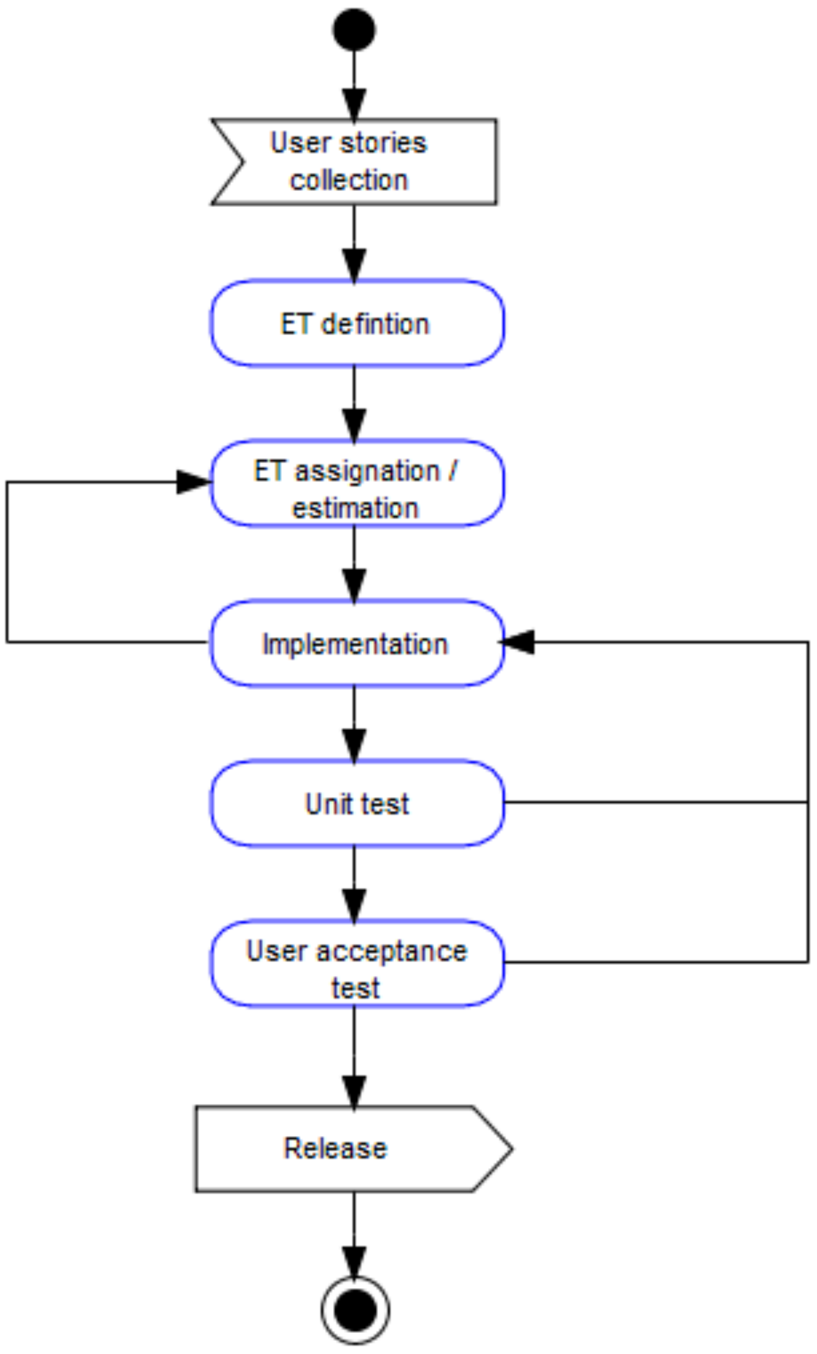


Fig. 8. XP Activity diagram of XP iteration

Extreme programming is not limited to a development workflow. It is better know as a set of practices that comply with the agile manifesto. Practices that we have explained previously.

4.3 Agile risk management practices

There have been many discussions regarding which aspects of risk management are already included in the agile methods. Most of the analysis done, (Boehm B. , 2002), (Nyfjord, Integrating Risk Management with Software Development: State of Practice, 2008), (Nyfjord, Commonalities in risk management and agile process models, 2007), (Moran, 2014) conclude that there should be a mix of plan-driven and agile methods, in order to increase the probabilities of success. The intention of this article is to analyze each aspect of risk management and their presence within agile methodologies.

As mentioned before, risk is defined as an unexpected event that may compromise the quality of the project and therefore, jeopardize the success of the project. Both types of methodologies; risk management models and agile development methods recognize the same definition of risk. Only with some exceptions that may consider risk also as a positive event and an opportunity of progress and improvement. This positive conception of risk is present in the SEI model (Von Scoy, 1992)

Regarding the methods used to store all the risk management information during a project, there are some differences between methodologies. Classic risk management suggests different models that include risk management documentation. Collection, classification, analysis and response to risk is recorded in structured documents for the purpose of experience collection that could be used as a reference tool for future decisions. Agile models reduce the documentation as much as possible; thus, the risks identified are not kept in any repository. Most of the information regarding risk management processes is displayed on walls or boards (workspace) and consulted during the progress meetings.

As part of the main risk management activities is the analysis of certain aspects as risk probability, impact and priority. All these characteristics determine the mitigation or tolerance decisions that could be taken for each risk identified. It is clear that classic risk management methodologies follow structured models, which qualified and quantified the aspects mentioned above. Some methods like the one proposed by SEI have a defined taxonomy to classify the risks. In contrast, agile methodologies do not describe any specific risk classification or estimation. During the progress meetings in agile development, risk identification is done intuitively, severity is based on team knowledge and experience, and usually there is no formal quantification of any aspect of the risks.

The main purpose of risk management is to eliminate risks or transform them into acceptable (tolerable uncertainty), in order to make decisions with less subjectivity. Therefore, the impact and the probability of occurrence of risk should be measured.

While risk management models define clear stages of risk assessment, agile methodologies do not describe any risk management phases within their activities and processes. All decisions regarding the action to be taken are based on team member's opinion. Agile teams do not use any metrics to evaluate and/or determine the risk impact and probability of occurrence.

4.4 Metamodel vs. Model

The definition of every model is based on a specific language, a set of rules and processes. This language is represented as a metamodel. Graphically represented the metamodel indicates the model elements, properties, relationships and rules. In this section we introduce the concepts of metamodeling and graphical representation that were used for the development of this dissertation.

The process to define a metamodel covers the following steps (Picka, 2004), (MetaCase, 2014):

- Define the basic concepts: Every language is defined by its syntax and semantics. For modeling purpose the same concepts applied. The basic concepts defined in a metamodel correspond to the description, conditions, constraints, models and theories that would be part of the language and should be possible to represent them graphically. In this phase the metamodel author should identify the objects that act in the model.

- Design of model symbols: In software engineering the use of graphic tool become the preferred option for communication among stakeholders or member of a development team. The description of the symbols to be used in a model is defined by the metamodel. Each of the objects previously identified should be graphically represented. These symbols will be then use for models that will aid the communication of requirements, processes and rules between the members of a project team. The design of these symbols is up to the author of the metamodel. However, it is advice to keep some consistency and relation with the description of the object within the metamodel.
- Define properties: The properties of each object, as in a class diagram, represent the attributes that characterize the static structure of the language concepts. Each property has a defined data type and can be part of the relationship with another object or external source.
- Define relationships and roles: The behavior is described by the relationships between objects. Roles describe the relationship ends and its direction. This step is necessary to define the bindings between all the elements of the metamodel.
- Define rules: As in any language, there are certain rules defined for the combination of the elements. In metamodeling, these rules can be related to bindings, cardinality, etc.

The concepts used in this document are based on the GOPRR metamodeling concepts. (Kelly, Towards a Comprehensive MetaCASE and CAME Environment: Conceptual, Architectural, Functional and Usability Advances in MetaEdit+,

1997) Kelly described the structured required for a metamodel with five basic elements, which have been previously described:

- Objects
- Properties
- Relationships
- Roles

Later in 2013 in (Kelly & Pohjonen, Dynamic Symbol Templates and Ports in MetaEdit+, 2013) introduced the concept of port making GOPRR in GOPPRR.

Ports are additional connection points to objects, where semantics of roles have different definition.

Metamodeling is used to define a new language for modeling new methodologies.

5 Discussion

5.1 Analysis of existing methods

There are several studies that intend to analyze the integrations of agile methodologies with classic development practices in order to assure quality. One of the most interesting samples is (Nyfjord, Towards integrating agile development and risk management, 2008)

The two critical aspects that affect users in classic development are delivery time of functional software and the variable requirements. The long periods of development do not consider the possible changes in the environment that may inquire and adjustment in requirements. Therefore these two issues are entirely related. Classic development lacks capacity of adaptability to new requirements. Usually these "late" changes are very expensive for the client and the team.

XP is a good method to attack the first situation, long periods of development are exchange for short iterations, which result on functional software, which could be checked and approved by users.

Additionally XP offers a group of practices that are not strictly connected with a workflow and can be implemented easily for any type of project.

SCRUM focuses on system flexibility in a constantly changing environment. This approach facilitates the team to receive and process voluble requirements.

For this reason and the already studied cases like (Marcal, De Freitas, Soares, & Belchior, 2007) and (Kiniberg, 2007) we consider XP and SCRUM guarantee better approach for successful projects. The opportunity of research is the integration of these combined methodologies with formal practices of risk management.

There are several samples of research towards comparing and combining XP and CMMI. Studies at the University of Pernambuco, Brazil (Santana, 2009) analyze the elements from both technologies that analyze the elements from both technologies that were implemented in two companies with the intention of merging these two technologies. The authors insist on a false impression when current studies try to map practices of CMMI into Agile approaches. It is not clear how CMMI and Agile could be merged in order to improve the company processes.

Other studies like (Martisson, 2003) claim that XP and CMMI are complementary tools. XP defines how to develop software, while CMMI guides trough what to do from an organizational level. In terms of risk management practices there is not enough evidence of merging practices of these two technologies to improve the identification, measurement and control of risks during the project.

In regards to Zachman framework there is evidence of research towards using Zachman classification for software development. In (Stoll & Wall, 2009) it is assumed that Zachman framework describes the software development organization and the customer. This approach could be valid for identifying root-cause risk from an organizational point of view.

Most of the literature found in risk management using the Zachman framework was focused on the construction field not the software industry. The nature of this framework requires extended documentation; this fact makes the method incompatible with any agile method.

FMEA and agile development has not been addressed extensively. However, due to nature of this technique we can say that fits accordingly the agile approach. The effort, documentation and time required can be easily integrated in an agile team. However, its implementation cannot be performed independently to the iterations. FMEA concepts should be integrated within the agile terms of the iteration. This thesis presents a proposal to integrate the mentioned concepts with the agile practices and processes.

Outlining a model for integration of risk management and agile software development has been the purpose of certain research as in (Nyfjord, Towards integrating agile development and risk management, 2008). However, the solution proposed in this research does not construct a specific language to be used as a tool for modeling and documenting risk management process within a agile development project.

My intention with this thesis is to propose an abstraction of the two most well know practices of agile development: XP and SCRUM. This combination has been already addressed in other studies and applied in many institutions in Europe (Jensen & Zilmer, 2003), (Mar & Schwaber, 2002) however my approach is different. The main focus would be the risk identification and metrics applicable for these methodologies. This outlines two main objectives (1) Formal description of already existing risk management processes into agile methodologies and (2) to approximate a better implementation of risk management activities suitable for XP and SCRUM.

Recent statistics show how these two methodologies have become the preferred selected by most organizations that need to develop with less planning and documentation; which is the case of many institutions in the Czech Republic (Buchalcevová, 2009) and in general all around the globe (Ambler, 2008).

The combination of these two practices requires metrics for validating the performance of their coexistence in order to be able to implement a risk management process like FMEA (Stamatis, 2003), which could be applicable for both.

The next steps are towards a definition of the tasks that should be used to cover the main processes of risk management (identification, evaluation and control) within extreme programming projects. Therefore, is necessary to identify which principles, metrics and processes for existing classic risk management (i.e. Impact, occurrence, severity) can be adopted by the extreme programming methodology.

Regarding risk management, agile development methodologies do not provide sufficient guidelines to meet the primary activities of risk management (Levine, 2005). However, a mix between traditional and agile methods is entirely possible and necessary in order to make decisions regarding the response to the uncertainty of a project (Boehm B. , 2002). Actions to mitigate or define the risk tolerance require the use of measures that quantify the impact and probability of occurrence of a risk. To merge risk management and agile development, we have chosen extreme programming as an example. Three risk management activities should be included within the extreme programming basic process: Risk identification, risk association to engineering tasks and risk measurement. As agile methodologies do not provide any metrics to quantify the risk effect and occurrence, it is necessary to find already existing and suitable practices for these processes.

From my point of view, there are still blurred areas in the agile approach, especially in terms of quality assurance and risk evaluation. As Boehm implied it is needed to combine the agile approach with some of the plan-driven characteristics.

The reasons for what we have chosen this area of research are the continuous development and increase in popularity within the European industry and the lack of formal research. Due to the recent popularity of agile methods, more companies and development teams are interested in their implementation. However, there is still a lack of formality as well as discrepancies between authors regarding its definition in certain aspects, as it is quality assurance.

6 Solution - Elaboration of hypothesis

According to the hypothesis established in section 1.2 the aim is to evaluate the risk associated to user stories and their engineering tasks. The purpose is to support the decision of inclusion of the ET within a Sprint according to the risk impact and necessary response.

We have seen in the available literature that agile approaches do not formalize the activities related to risk evaluation and quantification of their impact.

The concept of short iterations used in agile approaches gives the opportunity to include risk evaluation during the whole project development and not only at the beginning. This represents an advantage since the risks identified in previous Sprint can represent a change for the next Sprint in order to avoid unexpected failures.

The challenge of this study is to find the activities and practices from risk management that comply with the agile manifesto and do not jeopardize the agile nature of a team implementing SCRUM and/or XP methodologies.

FMEA has been widely used in project of a different nature than software development, showing excellent results for risk prioritization.

As expected the origins of FMEA are in the military field. The original goal was to have a process to evaluate the effect of equipment failures, with successful cases of use in NASA. Therefore, manufacture is where FMEA finds most of its applications.

One of the recognized uses of FMEA is in air traffic management (Raspotnig & Opdahl, 2012) where combined techniques of FMEA and sequence diagrams result in a more accurate visualization of error propagation.

Variations of the FMEA technique have been developed by different companies and institutions whit the intention of accommodate failure evaluation to their process. One well-known case is the one of FORD where now they have their own FMEA handbook specifically with FORD concepts (Ford Motor Company, 2008)

The following table (Table No. 5) Shows the FMEA form used in beer brewery. The team divided the form in process and defined potential failures for each process step (e.g. Preparing the wort, Mixing content till ingredients solved, etc)

Effects	S	C	Failure Mode	Causes	Preventive Action	O	Detection Action	D	RPN	R/D
Process Element: Sterilisation										
Function: Sterilise all of the equipment and then rinse with clean water and allow to dry										
[5% Indian Pale Ale Beer] 40 pints of beer is scapped	8		Equipment is not sterilised	[Instructions] Ambiguous methodology	Initial State: 13/03/2013					
[5% Indian Pale Ale Beer] Beer makes the party goers unwell	9				Use of good quality brew kit ingredients	2	Instructions are read thoroughly before commencing brewing	3	54	
[5% Indian Pale Ale Beer] Beer taste has a bitterness to it	7				Shop owner / supplier advice					
[5% Indian Pale Ale Beer] Beer is too gaseous (bad)	9			[Instructions] Metric / imperial weights and volumes not specified (including U.S. imperial)	Initial State: 13/03/2013					
[5% Indian Pale Ale Beer] Beer aroma is poor (e.g. vinegary)	9				Use of good quality brew kit ingredients	2	Directions are read and re-read at each stage of the brewing process	2	36	
[5% Indian Pale Ale Beer] Beer is not ready for Gavin's 40th birthday party	8				Shop owner / supplier advice					
[5% Indian Pale Ale Beer] Beer is cloudy/hazy	7				Measuring devices at Gavin's home display both imperial & metric units					
				[Brewer] Fails to sterilise container used to carry water	Initial State: 13/03/2013					
					Instructions clearly express that all equipment needs to be sterilised	3	none	10	270	
				[Brewer] Does not wash hands beforehand	Initial State: 13/03/2013					
					Common sense / hygiene	3	none	10	270	
				[Instructions] Insufficient detail about alternative ingredients (does & don'ts)	Initial State: 13/03/2013					
					Use of good quality brew kit ingredients	2	Instructions are read thoroughly before commencing brewing	5	90	
					Shop owner / supplier advice					
				[Steriliser] insufficent concentration / ratios	Initial State: 13/03/2013					
					Quantity is specified in the instructions	3	Sterilising solution has a strong smell	4	108	
					Instructions specify imperial and metric weights and volumes					

				[Home] Poor air quality (micro organisms present)	Initial State: 13/03/2013 Controlled environment	3	none	10	270	
				[Brewer] Washes out sterilising flu- id too early	Initial State: 13/03/2013 Instructions clearly express that all equipment needs to be sterilised	3	Sterilising solution has a strong smell	2	54	
				[Water] Domestic supply quality is contaminated (micro organisms present)	Initial State: 13/03/2013 Water quality is assured by water board	2	none	10	180	
					Revision State: 20/03/2013	2	Beer clarity and smell are checked during the sec- onadary fermenta- tion stage	8	{144}	Robbins, Gavin, Penkridge, FMEA Facilitator / Trainer 22/03/2013 in revision
[5% Indian Pale Ale Beer] Beer tastes of detergent	8		Steriliser not ful- ly rinsed from equipment	[Brewer] Does not rinse equipment properly	Initial State: 13/03/2013 Instructions clearly express that all equipment needs to be sterilised	3	Sterilising solution has a strong smell	2	48	
[5% Indian Pale Ale Beer] Beer is not ready for Gavin's 40th birthday party	8									
[5% Indian Pale Ale Beer] 40 pints of beer is scapped	8			[Brewer] Forgets to rinse equip- ment properly	Initial State: 13/03/2013 Instructions clearly express that all equipment needs to be sterilised Past experiences with simi- lar brew kits that are being drank with gusto!	2	Sterilising solution has a strong smell	2	32	
Process Element: Preparing the Wort										
Characteristics: Place Malt & Hops container into warm water for a time of >= 5 mins +5										
[5% Indian Pale Ale Beer] Brewing process takes longer than required	6		Malt & Hops not warmed sufficiently	[Instructions] Ambiguous methodolo- gy	Initial State: 13/03/2013 Use of good quality brew kit ingredients Shop owner / supplier advice	2	Instructions are read thoroughly before commenc- ing brewing	3	42	
[5% Indian Pale Ale Beer] Beer is watery	7									
[5% Indian Pale Ale Beer] Alcohol content is not as per product description	6			[Kettle] Thermostat failure	Initial State: 13/03/2013 TPM on all devices	2	Visual inspection	1	14	
				[Brewer] Impatience / care- lessness or inexperience	Initial State: 13/03/2013 Past experiences with similar brew kits that are being drank with gusto! Instructions supplied with each brewing kit & can be downloaded from the internet	3	none	10	210	
					Revision State: 20/03/2013	3	Regular referral to the	5	{105}	Robbins, Gavin, Penkridge, FMEA

	🟨					🟩	instructions	🟩	🟩	Facilitator / Trainer 03/04/2013 in revision

Table 4. FMEA form real example (Gavin Robbins Ltd., 2013)

Several studies have been developed proving that FMEA finds its application also in the Agriculture sector. Existing methods for risk assessments have been also combined with FMEA in order to guarantee more accuracy.

T.H. Varzakas from the Technological Educational Institute of Kalamata in Greece has several use cases of FMEA in Agriculture. In all of them the main emphasis is on the quantification of risk assessment by determining the RPN per identified processing hazard.

In (Varzakas, et.al., 2010) there is a comparison of ISO22000 analysis with HACCP over pistacchio processing and packaging. The processes of salting and roasting, hand grading of split nuts to remove defects and debris, packaging and storage or shipping, drying of split and non-split nuts to 5-7% moisture as well as dumping of nuts and conveying over an air leg to remove debris were identified as the ones with the highest RPN (280, 240, 147, 144, 130, respectively).

As FMEA suggests, corrective action were taken, depending on the level of tolerance of the identified risks. Following these actions RPN was calculated again obtaining significantly lower values.

Other methods were also applied, like the Ishikawa (Cause and Effect or Tree diagram). The results corroborated the validity of conclusions derived from risk assessment and FMEA. Therefore, the author considered that the incorporation of FMEA analysis within the ISO22000 system of a pistachio processing plant is considered essential.

In (Arvanitoyannis and Varzakas, 2008) as in the previous one a combination of the Failure Mode and Effect Analysis (FMEA) and ISO 22000 was applied for risk assessment, this time in salmon manufacturing processes.

Critical Control points were identified and implemented in the cause and effect diagram (also known as Ishikawa, tree diagram and fishbone diagram).

The processes with highest RPN identified were: Fish receiving, casing/marking, blood removal, evisceration, filet-making cooling/freezing, and distribution (252, 240, 210, 210, 210, 210, 200 respectively). As in the previous example the authors recalculated the RPN after the corrective actions were taken. The result once more shows that the incorporation of FMEA analysis within the ISO 22000 is anticipated to prove advantageous to industrialists, state food inspectors, and consumers.

The University of Bonn, Germany also has carried out studies of FMEA in Agriculture. In (Gödderz, 2007) the motivation to apply FMEA were the strong regulations of the government and other organizations related to hazard control in agrofood. Quality assurance becomes the aim of these regulations. The authors considered that FMEA could be an appropriate tool to enable animal health services to support farmers to fulfill these requirements. The paper presents a computer aided FMEA tool, which includes elements of the HACCP concept. The tool allows documenting efforts made to meet the claims of quality assurance and simultaneously provides gathered knowledge in form of a knowledge data base supporting the advisory service to solve concrete problems on farm. During the study, it was discovered that FMEA allows proving the execution of these procedures for health certification and health insurance purposes according to the demands of EU-regulations and distributive trade.

6.1 Analogy

The FMEA is an analysis of potential failures or mal function in a system. It was originally intended for use in the military institutions and projects from NASA. As many of the other risk assessment methodologies, it became popular in the manufacturing and service industries.

The use of FMEA in software development started timidly some decades ago however there are more actual studies, which analyze its implementation, like (Ern, Nguyen, & Noll), (Chang, 2013), (Bicchierai , Bucci, Nocentini, & Vicario, 2012)

(Ern, Nguyen, & Noll) is based on the theory that the effects considered in FMEA analysis are only the anticipated ones. They explain that unexpected effects during the project are not approached by FMEA. They proposed a Java based solution to inject and AADL model as input to generate a matrix of effects. However, their final implementation remains in the construction and hardware field.

Our intention with this study is to propose an iterative use of the FMEA as part of the process of an agile project, by nature iterative. Not only for identifying failures in the final product but also possible obstacles that may affect the development of the project itself.

This iterative analysis of risk will allow the team to consider update of the risk register after every iteration or Sprint.

We would like to evaluate the relationship between the concepts of FMEA and Software risk management previously described separately:

Failure: According to FMEA a failure is described as any malfunction in a system. In Software engineering, especially in the agile approaches, a failure can be described as fail to meet any of the functional or nonfunctional requirements.

Samples of failures in a software project can be divided in two categories, the one related to the final product and the failures related to the project management.

Sub categories of failures/risks related to final product operation include:

- User interfaces fails to meet user expectation and/or needs
- Compatibility issues with external systems or subsystems
- Functionalities not included
- Time and/or budget exceeded

Sub categories of failures/risks related to project management include:

- Overestimated release increment
- Truck factor (see section 4.2.2) (Torchiano, Ricca, & Marchetoo, 2011)
- Change of requirements
- Technical failure in the systems used for development
- Human communication errors

Effect: The definition of a risk effect is given by the consequences of its occurrence. This effect may or not affect the final user. However, it has an effect for at least some of the stakeholders or the development team. The consequences can be categorized as: process, operation, product, user or government regulations. The question to ask in order to identify the possible effects would be: What happens if the risk becomes a failure? Historical data as well as expert judgment can be consulted to identify effects. It is important to evaluate not only the internal consequences but also the consequences related to the environment where the product/project is develop/used. Generally a consequence of a risk is the proof performance of the product or a dependent subsystem.

6.2 Metamodeling Agile Risk Management

In order to fulfill the metamodel generation, the following steps should be completed:

1. Identify the elements that correspond to <u>objects</u> within the agile risk management activities.
2. Identify the <u>properties</u> of each element (object)
3. Define the <u>relationships</u> between objects according to risk management workflow activities
4. Define <u>roles</u> for the relationships previously established.
5. Represent graphically the structure based on GOPRR
6. Define <u>scenarios</u> for validation of metamodel
7. Prepare models for validation.

6.2.1 Identification of objects and properties

In order to identify the objects of the metamodel we have to keep on mind the three areas covered by this study: Software development, Risk Management and Agile development practices.

Software development in essence requires a life cycle, which guarantee a final product according to the user, needs. The life cycle or development processes considered in this thesis are strictly agile methodologies (XP and SCRUM). Therefore, all the objects and properties to be identified are concepts related to the agile approach. No waterfall concepts have been taken into consideration for the metamodel.

6.2.1.1 Risk management concepts

The risk management concepts identified for the metamodel correspond to the base requirements of risk management idea. These concepts are not strictly attached to a particular method. The intention is to guarantee that the model covers the basic activities of risk management. However, some particular properties from FMEA have been implemented in the model. As mentioned, in section 2.4, due to the nature of FMEA we consider this method to be ideally compatible with the agile development approach.

Objects to model

- Risk: Principal object that represents the already identify and defined uncertainty.
- Cause: Describes the origin of the risk. It may correspond to an event or characteristic. The source of the cause can be internal or external.
- Response: Planned action to be taken in case the risk occurs. Each risk should have at least one response action planned, depending risk effect. In some cases the response can be undefined if the risk is considered tolerable.

FMEA concepts to be implemented as properties

- Severity: Measure to quantify the level of impact that has the risk occurrence.
- RPN: The priority that will be assigned to each risk in order to be addressed. Depending on this measure the team should plan an action to avoid the risk.
- Occurrence: This measure is to find out the probability of occurrence of a risk.
- Detection rank: This measurement indicates how possible is to detect the risk at an early stage, before it becomes an issue.

6.2.1.2 Agile development concepts

- Backlog: Base directory of all the requirements of the final product.
- User story: Individual functionality described as the user expresses it.
- Engineering task: Precise task to be programmed by and assigned developer. A user story may be split into several engineering tasks.
- Sprint: Iteration given in a restricted period of time where a group of engineering tasks are developed, tested and implemented. After each Sprint, a piece of functional software should be delivered and presented to the user.
- Increment: Represent the set of user stories completed in a Sprint and previous ones. It is an object set of user stories already completed. The whole set of functionalities related to the user stories should be in usable state. The decision of product release is up to the product owner.
- Test scenario: For each engineering task there is a test case defined. A user story is considered completed where all the test cases have been successful.
- Velocity: Refers to the calculated amount of work that a team can compromise for a Sprint. This rate is calculated based on the complexity rate assigned to each user stories. The number of engineering tasks

derived from a user story is a clear parameter of the effort required. The history of the velocity provides information to evaluate the amount of work capable to handle by the team.

Concept	Object decomposition	Property decomposition
Risk	RMrisk	NA
Cause	NA	RMrisk:cause
Response	RMaction	NA
Severity	NA	RMrisk:sR
Occurrence	NA	RMrisk:oR
Detection	NA	RMrisk:dR
RPN	NA	RMrisk:rpnR
User story	ADuserStory	NA
Engineering task	ADengineeringTask	NA
Sprint	ADiteration	NA
Increment	setObjects: ADiteration	NA
Test scenario	ADuserTest	NA
Velocity	NA	ADuserStory:developmentEffort
		ADengineeringTask:developmentEffort

Table 5. Concept mapping

6.3 Concept to object

The concept describe in the previous section have been discomposed as objects or properties. The mapping is described in the Table No. 4

The concepts related to risk management are named starting with RM. The concepts related to agile development (Scrum and XP) are named starting with AD.

The concepts related to risk management are named starting with RM.

6.4 Graphic representation

The tool used for the graphical representation of the metamodel was MetaEdit+ version 4.0. by MetaCase. MetaEdit Plus is a Domain-Specific Modeling (DSM) environment. (MetaCase, 2014)

The tool provides all the necessary options to developers for modeling design.

The tools support the metamodel creation, its concepts, rules, notations, diagrams and code generators.

MetaEdit+ provides support for multiple metamodels and allows automatic update of model when the metamodel changes. It also support code generator debugger.

The figure 16. shows the metamodel diagram created with MetaEdit+

The symbol editor allows the developers to design their our model symbols or import them in several graphic formats.

Concepts were implemented in the model. Following, the description of the model objects and its properties:

6.4.1.1 ADIteration

- Semantics: Represents a Sprint in SCRUM or iteration in XP. As described before, it is a time box element where developers should develop a set of functionalities ready for use.
- Attributes: Each iteration has a code for identification (ID). Besides ID additional attributes are defined:
    - startDate and endDate: The iteration is timeboxed, therefore it has a start and end date
    - length: Number of weeks predefined for the iteration.
    - developmentPoints: Depending on length and the effort calculated for each ET, the iteration would have a number of

points to be developed. The number of points per ET is estimated based on complexity. This estimation is assigned as developmentEffort in ADEngineeringTask

  - Velocity: The time necessary to complete the assigned ET in the iteration.

- Symbol: Circle with arrow-end black filling – Text below the symbol

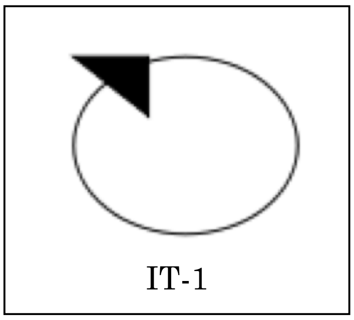


Fig. 9. ADIteration metamodel symbol

- Bindings: ADIteration has one defined binding to ADUserStory. The relationship is described as: ADIteration consists of ADUserStory(s)
- Cardinality: M:N one or many (at least one) ADIteration consists of at least one ADUserStory.

6.4.1.2 ADEngineeringTask

- Semantics: Represents each of the engineering tasks derivate from user stories. An engineering task represents concrete task that developers should do. The approach of these tasks is not necessary user oriented. It corresponds to the translation of a user story into real programming sub-tasks.
- Attributes: In addition to the attributes inherited from WorkUnit, ADEngineeringtask defines an additional attributed called developmentEffort. DevelopmentEffort correspond to calculated effort

required to complete the task. This attribute is also used to calculate the total developmentPoints of a Sprint (ADIteration)

- Symbol: Rounded rectangle with dark green border and light green filling. "Person" icon in dark blue on the right top corner.

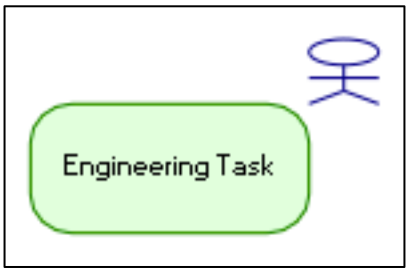


Fig. 10. ADEngineeringTask metamodel symbol

- Bindings: ADEngineeringTask has inheritance relationship with WorkUnit.
- Cardinality: ADEngineeringTask takes the cardinality defined for WorkUnit: N:1 One or many Work Unit(s) belongs to one ADUserStory.

6.4.1.3 ADUserStory

- Semantics: Represents an agile "user requirement". Each of the functionalities required by the user is defined in natural language to be later fragmented into single development tasks called engineering tasks.
- Attributes: Identification code (ID) and additional attributes:
    - Description: Corresponds to the text given or agree with the user that describes the desired feature for the system.

    - Status: Describe the current status of the user story (e.g. planned, in progress, developed, tested, release)

- Symbol: "Person" icon in dark blue. Please note the reference to the UML symbol for User.

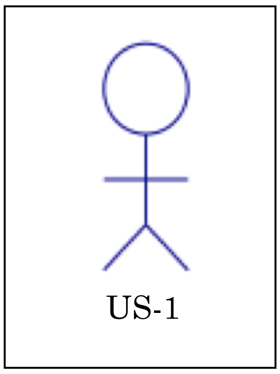


Fig. 11. ADUserStory metamodel symbol

- Bindings: ADUserStory has defined three bindings, one to ADUserTest, one to ADIteration and one to WorkUnit.

  Relationships are described as follows:

    - ADUserStory calls ADUserTest
    - ADUserStory has a set of WorkUnit (See cardinality)
    - ADUserStory belongs to ADIteration

- Cardinality:

    - 1:1 ADUserStory belongs ADIteration
    - 1:1 ADUserStory calls ADUserTest
    - 1:N ADUserStory has a set of WorkUnit (See cardinality)

6.4.1.4 ADUserTest

- Semantics: Represents a testing component for each ET. An ET is considered completed after successful UT.
- Attributes: Besides ID and description of the test, additional attributes are defined:
  - Status: Defines the current status of the test (e.g. developed, run, not run, in development, planned)
  - Success: Defines the result of the test run (e.g. failed, passed)
  - TTC (Merunka, 2009): Unit that expresses the time required measured by the number of days that are needed to complete the test. It can be calculated as:

TTC = MD / (team size * FTE team)

For example, if the task requires working 6 working MD (Man/day) and the team are 3 workers who devote 50% to the project.

TTC = 6 / (3* 0.5) = 4 days

- Symbol: Orange diamond with an orange square in the middle and the U letter. ID of the user test is written bellow the diamond.

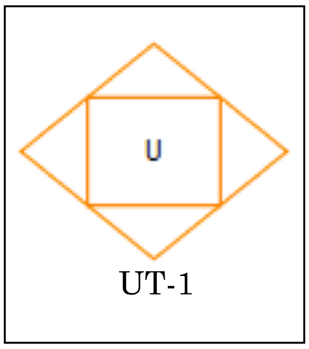


Fig. 12. ADUserTest metamodel symbol

- Bindings: ADUsertest has one defined binding to ADUserStory. The relationship is described as: ADUserStory calls ADUserTest

- Cardinality: 1:1 one ADUserStory calls one ADUserTest.

6.4.1.5 RMAction

- Semantics: Represents the risk response action to be taken in case of risk occurrence.
- Attributes: RMAction inherits the attributes of WorkUnit.
- Symbol: Green diamond with a red triangle in the middle. Identification text is underneath the diamond.

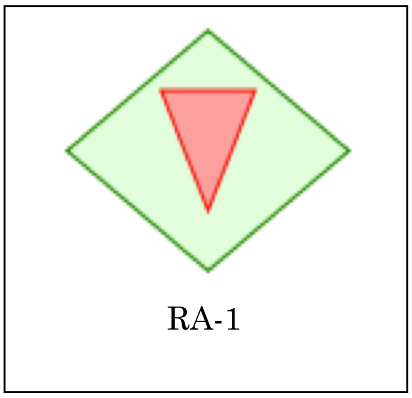


Fig. 13. RMAction metamodel symbol

- Bindings: RMaction has inheritance relationship with WorkUnit. No new attributes are defined. RMaction also is bound to RMrisk.

  RMrisk calls RMaction

- Cardinality: ADEngineeringTask takes the cardinality defined for WorkUnit: N:1 One or many Work Unit(s) belongs to one ADUserStory. RMrisk and RMaction has cardinality 1:1

6.4.1.6 RMrisk

- Semantics: Represents the risk or potential failure identified.

- Attributes: Besides ID and description of the test, additional attributes are defined based on FMEA concepts:
    - oR: corresponds to the occurrence rank assigned to the risk.
    - sR: corresponds to the severity rank assigned to the risk
    - dR: corresponds to the detection rank assigned to the risk
    - cause: description of the possible cause of the risk.

- Symbol: Red triangle. Identification text is written underneath the triangle.

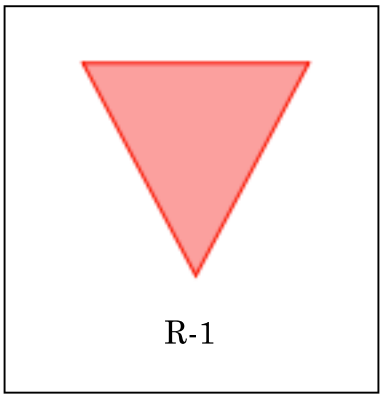


Fig. 14. RMrisk metamodel symbol

- Bindings: RMrisk is bound to RMaction. RMrisk calls RMaction
- Cardinality: 1 :1 RMrisk to RMaction

6.4.1.7 WorkUnit

- Semantics: Represents the risk or potential failure identified.
- Attributes: Besides ID and description of the test, additional attributes are defined based on FMEA concepts:

- oR: corresponds to the occurrence rank assigned to the risk.
- sR: corresponds to the severity rank assigned to the risk
- dR: corresponds to the detection rank assigned to the risk
- cause: description of the possible cause of the risk.

- Symbol: Green diamond. Identification text is underneath the diamond.

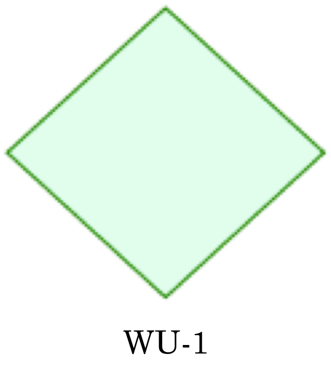


Fig. 15. WorkUnit metamodel symbol

- Bindings: WorkUnit is bound to ADUserStory. Also RMAction and ADEngineering task have a inheritance relationship with WorkUnit
- Cardinality: 1 :N UserStory is a set of WorkUnit(s)

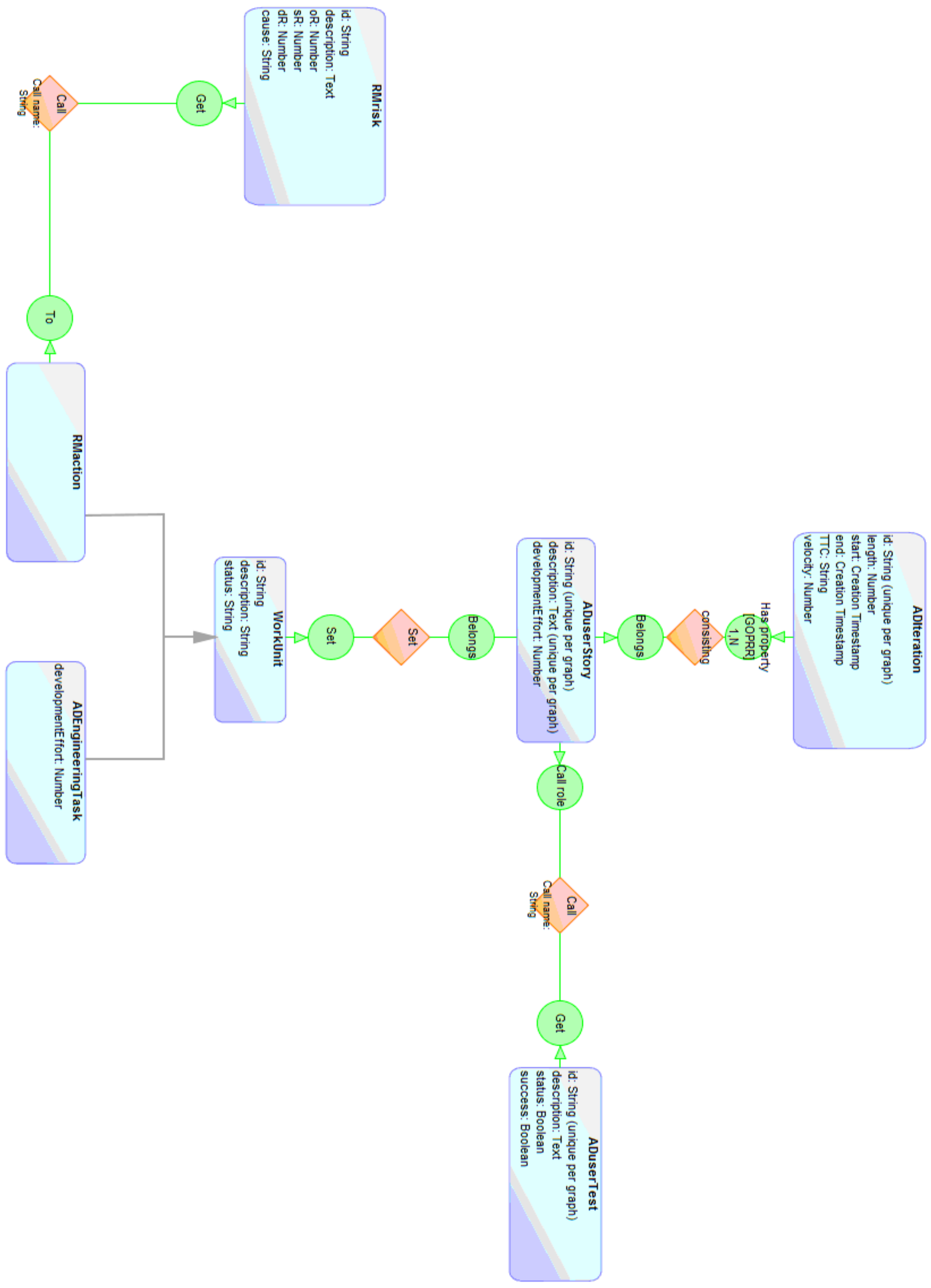


Fig. 16. Metamodel diagram for Agile Risk Management

7 Verification and validation

Sprint integration: During the Sprint planning meeting the Scrum master and development team discusses the user stories that will be included in the Sprint. As part of the agile risk model integration, the scrum master should lead questions that result in risk identification and can be easily associated to an engineering task.

One of the elements evaluated during the Sprint planning meeting are the obstacles that were present in the previous Sprint.

The intention of our model is to enforce the association of each risk, obstacles or failure to an engineering task. Subsequently, based on the calculated impact of the risk the team can prioritize the ET and/or plan new ETs as response to the risk if necessary.

In order to identify risks the team should evaluate each of the user stories from the backlog and address the potential risk for each one. For this purpose the team can answer the following questions:

- What is the risk?
- Can the cause be identified?
- Can the risk be quantified?

These questions are inspired on and are a complement to the suggested questions a SCRUM team should answer on each daily scrum meeting (Sutherland & Schwaber, 2011).

"What did I do yesterday that helped the Development Team meet the Sprint Goal?

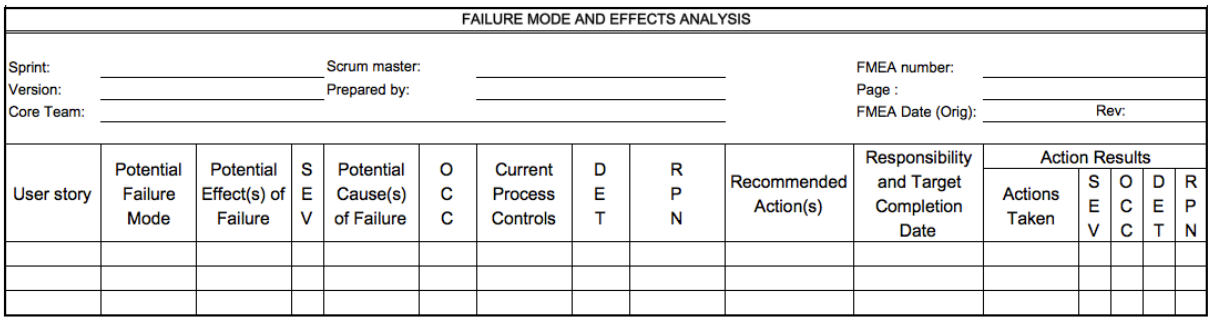
What will I do today to help the Development Team meet the Sprint Goal?

Do I see any impediment that prevents me or the Development Team from meeting the Sprint Goal? "

The questions related to risks and its characteristics should not be address on every daily SCRUM necessarily. However, this analysis should be performed at least at the beginning and the end of each SPRINT.

The identified risk should be collected. For this purpose the team should create an agile FMEA form to be used during the Sprint and updated during daily Scrums.

Table 5 shows the example of the agile FMEA form used for the study case. The form integrates the concepts of user stories and engineering task to a regular FMEA form.



FAILURE MODE AND EFFECTS ANALYSIS

Sprint:		Scrum master:		FMEA number:	
Version:		Prepared by:		Page :	
Core Team:				FMEA Date (Orig):	Rev:

User story	Potential Failure Mode	Potential Effect(s) of Failure	SEV	Potential Cause(s) of Failure	OCC	Current Process Controls	DET	RPN	Recommended Action(s)	Responsibility and Target Completion Date	Action Results				
											Actions Taken	SEV	OCC	DET	RPN

Table 6. Agile FMEA form

The first section of the form is used as identification panel also to connect the form with corresponding Sprint.

Fields required are:

- Sprint number,
- Project name or ID,
- Scrum master responsible,
- FMEA number
- Dates of creation (Original version) and revision

Additional fields that are optional like:

- Core team: Up to 6 participants with expertise for judgment. This team will be responsible for the FMEA, not necessarily the entire team of developers.
- Prepared by: This field is use in cases where the participant filling out the form does not correspond to the scrum master.
- Page: Used to identify the page number of the current FMEA.

The core section of the form is used to list the failures (risks) identify and perform the quantification and binding with user story(s).

- User story (US): This field corresponds to the user story to analyze. It may be identified with a number or the full description can be added here.

- Potential failure mode: Correspond to the identified risk associated to the US. There may be more than one potential failure per user story. In this case they will be listed in separate rows to be quantified and monitor individually.

- Potential effect of failure: For each risk (potential failure) the potential consequence should identify and registered. This consequence should be measurable.

- SEV (Severity): As explained before the severity of the risk must be calculated. SCRUM master and the FMEA team should follow the classification described in Table 1. FMEA-Risk Severity Ranking.

- Potential cause of failure: In order to plan an action the possible cause of the failure needs to be identified. This cause may be related or derivate form another User Story where in such case it is important to identify the source US.

- OCC (Occurrence): The level of occurrence of the potential failure must be quantified. Levels were described in Table 2. Occurrence criteria.

- Associated ET: Each US **derives** one or more engineering task. These ET needs to be associated with the potential failures identified in order to measure the impact and calculate the effort require to avoid/mitigate the risk.

- Current control action: This field is used to indicate the control processes in progress for the corresponding failure mode.

- DET (Detection): Detection rate as explained before, measures how possible is the identification of the occurrence of the risk. Table 3. shows the ranking for this measurement.

- RPN: The risk priority number a FMEA concept to quantify the risks associated to each user story. Numeric value that should be updated after response actions have been taken.

- Recommended action: According to the FMEA method for each potential failure should be an action planned in case of occurrence or in order to avoid the failure. This action depends on the level of tolerance defined for the consequences of the failure.

- Responsibility and Target Completion Date: This field will be used for determining responsible(s) for the planned response action. Target completion date could be determined in unit of effort required for the action.
- Action results: All actions taken are evaluated, registered and SEV, OCC and DET are recalculated. New RPN is achieved.

The team should be able to define the recommended action based on the following categories:

Action	Indicator	Description
No action at this time (Tolerate)	Severity is considerable low (1 or 2)	In the event of occurrence and detected by user, the team should negotiate with the user if it is necessary an action of correction
Add built-in detection devices	Detection rate is low (9-10) and the severity is medium (5-8).	The team should prepare an engineering task that increases the detection of the risk. Base on this detection requalification and the expected severity of the risk, the recommended action should be updated.
Provide alternatives to the design (Avoid before occurrence)	Severity is high (8-10) and occurrence is medium-high (4-10)	The risk should be avoided. The team should reconsider the user stories related to the risk and plan a different solution. If possible avoid the user stories that may increase the occurrence of the risk.
Add a redundant subsystem (Mitigation)	In cases where the occurrence is high (7-10)	The team should plan the response action to the effect. In some cases these requires new engineering tasks/user stories to be implemented.

Table 7. Response action according to FMEA parameters

Additionally the RPN should be used to identify possible user stories that may comprise the entire project. User stories with high RPN should be carefully analyzed and a concrete response action should be formulated.

7.1 Model validation

7.1.1 Scenario

One of the concepts in XP related to risk management is the “Spike solution”. This is a concept not so well documented but essentially created to find an answer to unclear stories and provide a better estimation for the release planning.

However, this method does not fully fulfill the risk management basic activities. There is still a clear need for implementing the risk management activities within the agile life cycle.

Where should the risk management activities be included? At least 3 main activities should be present in each iteration: Identification, association of ETs to risks identified and risk monitoring. The inclusion of these activities should not run parallel to the release planning. They should be part of the release plan, the risk prioritization should be considered at the moment of prioritizing ETs. The process is illustrated in figure 17.

The activities related to risk management included in the cycle are executed easily with the use of the FMEA form. The form allows us to record the identified risk. Every of these risks recorded are associated to and specific engineering task. The form allows us to estimate RPN for each risk and define development effort necessary for each engineering task.

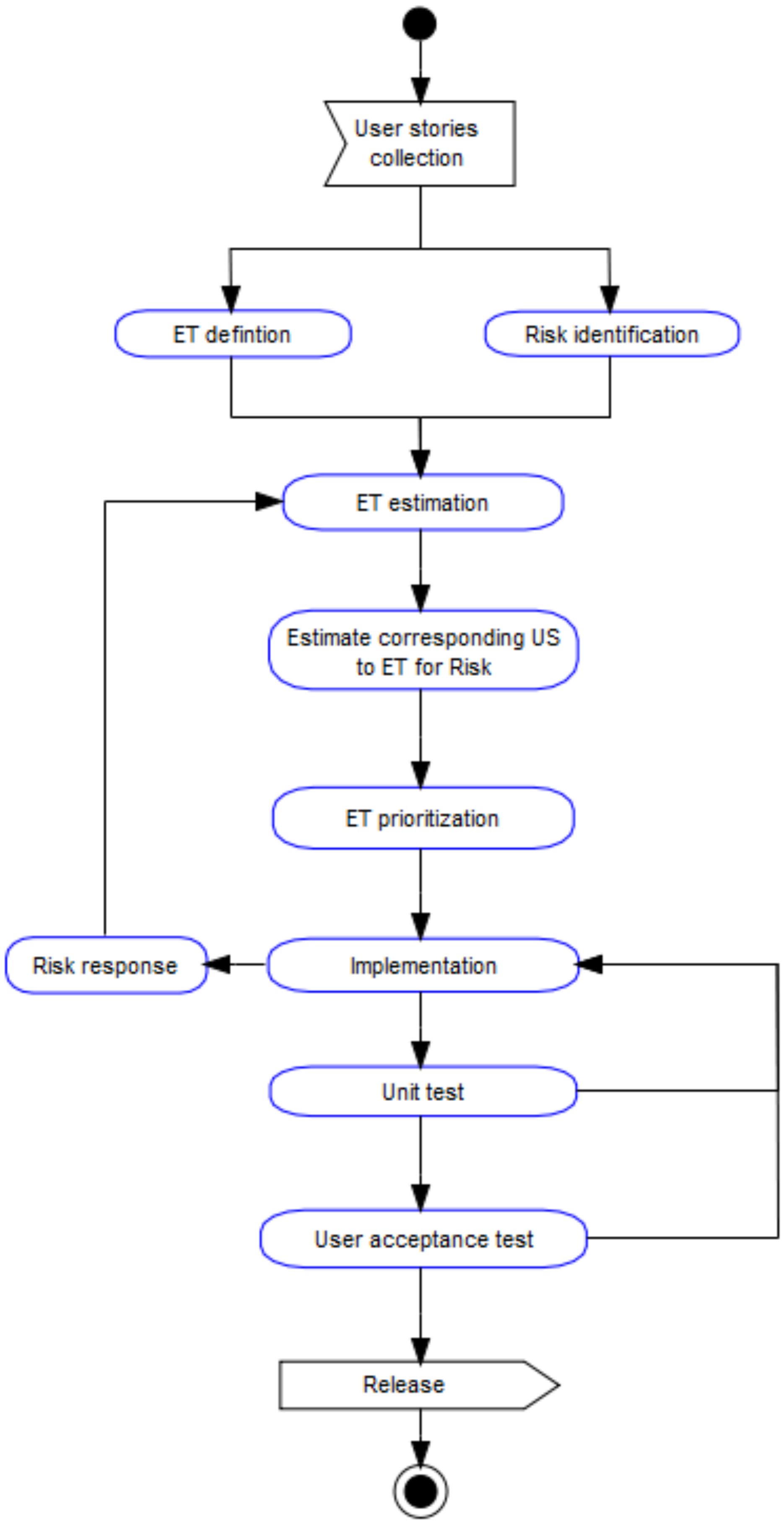


Fig. 17. Agile iteration integrating RM activities

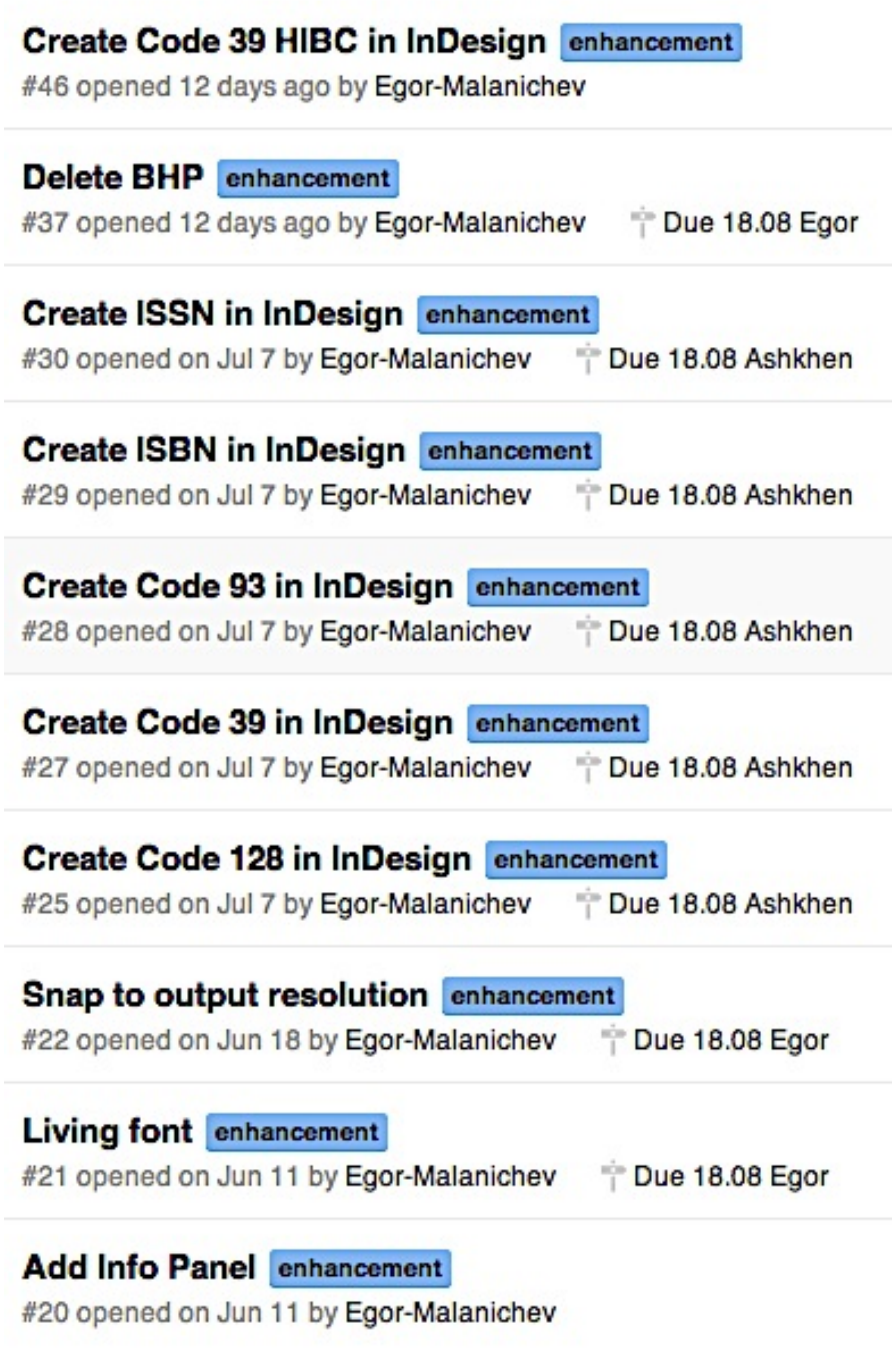
7.1.2 Study case

A job portal provides us with a set of user stories used in of their sprints to take as a sample. Some of the details have been hidden due to confidentiality policy.

12972 Incorporate impression/click tracking on Recommended Jobs page on Core	
12147 Canonical link on all JSR pages	Currently, Canonical link cannot handle special characters. **Examples:** Keyword search with “C++”, Canonical link returns the following: <hidden detail> Keyword search with “R&D”, Canonical link returns the following: <hidden detail> Keyword search with “BJ’s”: <hidden detail> **Participant’s note:** These examples are special characters and should be handled in another story (maybe next sprint)
13993 IE Search Field Hint Text Missing	~~DEV00739931 Story 8009: IE8 shows no description text inside the keyword search boxes~~
14172 Improvement story: identify benchmark stories for 1, 3, 5, 8, 13	

Table 8. Real sample of user stories from web job portal

Another sample of user stories is the given by a development team working on a plug-in for Adobe Illustrator and InDesign. The plugin generates code bars (EAN codes) to be implemented in the design of packages and labels.



Create Code 39 HIBC in InDesign enhancement
#46 opened 12 days ago by Egor-Malanichev

Delete BHP enhancement
#37 opened 12 days ago by Egor-Malanichev Due 18.08 Egor

Create ISSN in InDesign enhancement
#30 opened on Jul 7 by Egor-Malanichev Due 18.08 Ashkhen

Create ISBN in InDesign enhancement
#29 opened on Jul 7 by Egor-Malanichev Due 18.08 Ashkhen

Create Code 93 in InDesign enhancement
#28 opened on Jul 7 by Egor-Malanichev Due 18.08 Ashkhen

Create Code 39 in InDesign enhancement
#27 opened on Jul 7 by Egor-Malanichev Due 18.08 Ashkhen

Create Code 128 in InDesign enhancement
#25 opened on Jul 7 by Egor-Malanichev Due 18.08 Ashkhen

Snap to output resolution enhancement
#22 opened on Jun 18 by Egor-Malanichev Due 18.08 Egor

Living font enhancement
#21 opened on Jun 11 by Egor-Malanichev Due 18.08 Egor

Add Info Panel enhancement
#20 opened on Jun 11 by Egor-Malanichev

Fig. 18. Screenshot extracted from the project repository in GitHub

Study case:	Flexibarcode – Stolte Packaging (http://www.flexibarcode.com/)
Case description:	Plugin for Adobe Illustrator and InDesign for creation and edition of bar codes in packages and label design. Commercial license.
Study context:	JavaScript and Action Script.
Data sources:	GitHub and personal interviews
Sample:	3 User stories – 13 Engineering tasks 4 User stories (not exposed)

Table 9. Flexibarcode Project Sample description

User story	**Engineering tasks**
Add ISBN/ISSN text #16	- Define text format specification - Prepare checksum calculation - Identify and separate check digit - Add checkbox for text option of ISBN code - Modify createIllustrator.in protocol ISBN to add parameter for ISBN code to pass in JavaScript file. -Add new parameter in ISBN JavaScript file for ISBN text
Add Vertical Shift and Horizontal Spacing tags #52	- Add spacing field in interface - Add Vertical Shift numeric field in pts - Add spacing tag for readability - Add vertical tag for code readability
Snap to output resolution #22	- Add numeric field in interface for ppi resolution - Transfer parameters of resolution to createBarcodeform on each of the protocols of the affected barcodes - Implement new parameters in the formula to calculate X dimension of the narrowest bar.

Table 10. Project Sample Flexibarcode: User stories and engineering Tasks

The same company developed a project management tool specifically for artwork creation and print management projects. The tool is very customized according to the type of clients the company has. It follows a specific workflow and the business rules are very strict and not easy to find in other commercial software tool. The development team for this project was in different location (Yerevan, Armenia) and had restrictions in terms of access to the engineering tasks. We had access to the user stories and the subsequent tasks derivated from failures in the system. The tram worked following the SCRUM methodology, product owners had an incremental delivery approximately every three weeks.

Key	Summary	Priority	Status	Resolution	Created	Due Date	Description
PM-39	Delete link for uploaded files	Major	Closed	Fixed	01/04/2010 07:41	16-Apr-10	During the flow when a file is being uploaded, should be a link to delete it in case of mistake. Only the last file uploaded can be deleted by the same person who upload the file, and before clicking the button SAVE.
PM-41	New field in packshot form	Minor	Closed	Fixed	01/04/2010 07:55	18-May-10	A new field for the "Material Code" is needed in the "New Artwork" page of the Packshot project. This will be an optional field.
PM-53	Color view per task according to status	Major	Closed	Fixed	10/05/2010 08:42	12-May-10	Each task (Packshot-artwork) should change its color according to the status.
PM-87	Delete artwork link	Minor	Closed	Won't Fix	21/07/2010 13:58	26-Jul-10	Account manager should be able to delete an artwork from a project. I create a task by mistake and I was not able to find the link to delete the task, only I found delist which has completely different meaning.
PM-118	Billing report	Major	Closed	Fixed	01/02/2011 10:14	04-Feb-11	--copy of my last email--- Dear Mr. Vasil, I would like kindly to ask you to restart the development of the billing report. This chart will be per project. It should be generated automatically every month and also manually at any time by supervisor. Please find attached an example of this report. This is exactly the look that we required. It will be also good if the report can be generated in two formats excel and pdf. The name of the files should be PROJECTNAME_OV_DDMMYY The prices will be set up in the settings.

							As always I would like to ask you for the estimation (time and cost). Thank you for your help in advance. Kind regards, Sandra
PM-135	New columns in tasks list	Major	Closed	Fixed	23/05/2011 12:05		We need to include in the AW (tasks) list these columns: -Brand -Type of packaging -Volume (Size)
PM-136	File size note	Major	Closed	Fixed	23/05/2011 12:07		We need to see in the workflow section, the size (KB/MB) of each file uploaded. Please place it next to the date of upload or name of file.
PM-137	Comments section	Major	Closed	Fixed	23/05/2011 12:09		The comments related to a rejection should be shown in one single comment (paragraph) Right now we have a comment saying that someone rejected the task and then separately is the comment. Can the person who rejected made a comment save it and then send on and after that the system will show only one comment with all the information.
PM-138	Comments upload	Major	Closed	Fixed	23/05/2011 12:09		We need to have the option to upload a file as part of a comment.
PM-141	New workflow link	Major	Closed	Fixed	23/05/2011 12:13		We need a link to see the workflow of a task from the edition page of it.
PM-147	New report filter	Major	Closed	Fixed	23/05/2011 12:24		We need to be able to generate reports per project.
PM-149	link to LINKS	Major	Closed	Fixed	23/05/2011 12:26		We need a link to download the files from the folder LINKS of each task/project.
PM-150	A user with multiple roles	Minor	Closed	Fixed	23/05/2011 12:27		When an user is created there should be the option to choose which roles can perform. List of roles with check boxes is required in the user creation panel. In the login user should choose which role will use for the current session. Permissions should apply according to the role selected.
PM-151	Archiving option	Major	Closed	Fixed	23/05/2011 12:28		The projects completed should have the option to be archived which means to ZIP the folder and place it in a different server. The archived project shouldn't appear tin the list of current projects of each user.

PM-158	Applicator	Major	Closed	Fixed	07/07/2011 13:04		Applicator is a new characteristic of the artwork. It should be added in Settings following the same structure like packaging type. Content list is: Shampoo Conditioner Shower Liquid Bath Liquid Soap In the artwork definition should be added as a new field to be filled out by account anager durign the creation of the task. Also it will be required in the view of the task, folder structure and list of tasks.
PM-162	File name	Major	Closed	Fixed	17/08/2011 09:06		AI and PDF files should be automatically named following this structure: MA-Category-Brand-Applicator-Variant-PackagingType-Volume-MaterialCode-Version Example: MA-Skin-Dove-Shampoo-Coconut Milk-FL-250ml-8775738-C0.pdf
PM-165	New packshot panel	Blocker	Closed	Fixed	17/08/2011 09:25		A new characteristic should be added to the tasks: Packshot development. There will be an additional panel Packshot for each task. Packshots will have a separate workflow: 1. Open 2. PS created (3 files should be uploaded: ZIP (JPG,EPS,PNG), AI open(with LINKS), PDF) <If one of this is missing please alert the user> 3. Approved by PM 4. Proof sent to client 5. Proof approved/rejected by client (Always assume 1 country for this approval) 6. Completed Name of file should be automatically generated: PS-Category-Brand-Applicator-Variant-PackType-Volume-MaterialCode-Cycle Description of packshot: <Only one should be chose> Creation of Master 3D model (Standard-Complex) Rendering of individual artworks (Standard-Complex) Color retouching <Text field> minutes Digital color proofs <Text field> (number of proofs done)

PM-166	Generic Layout	Major	Closed	Fixed	17/08/2011 09:35		There will be a new type of task: Generic Layout The Description remains the same as artwork. Workflow will be: 1.Open 2. Assigned 3. GL created (PDF and AI) 3. File Approved by PM 4. File sent for client approval 5.Approved/Rejected by Client 6. Proof sent to client 7. Proof approved/rejected by Client (Always assume 1 country for this approval) 6. Completed It should be part of a project just like another regular task. AW tasks will be associated to one GL.
PM-174	Permission for desginer	Minor	Closed	Fixed	29/08/2011 12:30		Designer have the right to reassign the task to another designer
PM-176	Workflow flexible	Major	Closed	Fixed	29/08/2011 12:42		The workflow should change in this case: E-proof approval: If in task description has been chosen e-proof approval then the workflow should include the steps related to this proof. If it hasn't been chosen, then the workflow will be finished after the ZIP is approved by client.
PM-177	New step in workflow	Major	Closed	Fixed	29/08/2011 12:43		This step applies for the tasks with artwork type. Repro checked: Should be done before uploading the outlined file and after country approval. Designer/Supervisor should click on it and send on the task.
PM-195	LINKS list	Major	Closed	Fixed	07/11/2011 15:05		The files uploaded as LINKS should be shown in a list of files. Each one should have the option to edit and delete. Designer and PM are allowed to upload, edit and delete links. When a LINK is edit, the old one should be placed in the previous version folder for LINKS.
PM-196	Link to Tasks list	Major	Closed	Fixed	07/11/2011 15:06		Please add a link to Tasks list in the PROJECT add/edit page.
PM-197	Tasks view	Major	Closed	Fixed	07/11/2011 15:10		Designer should be able to see all the tasks from a project where he/she has at least one task assigned.
PM-199	Individual report	Critical	Closed	Fixed	13/12/2011 15:35	26-Dec-11	The system should generate report per task. The look and the content should be as the one in attachment.
PM-265	Proofing panel	Major	Closed	Fixed	29/03/2012 11:30	05-Apr-12	It is required to have an additional panel in the artwork workflow page for proofing files. Name of panel: Proofing Uploads: ZIP file Fields: The fields that actually we have in the Edit AW page should be moved

							to this panel -Proof required -Kodak proofs -Digital color proofs
PM-266	Link to edit task	Major	Closed	Fixed	29/03/2012 11:48	05-Apr-12	Please add a link to the edit task page in the workflow page
PM-267	Link to edit project	Major	Closed	Fixed	29/03/2012 11:49	05-Apr-12	Please add a link to the edit project page in the workflow page
PM-268	Brief panel	Major	Closed	Fixed	29/03/2012 14:58		it is needed a panel to upload additional files related to the artwork tasks. Name: Brief Upload: Multiple files at the same time 4 upload dialog boxes: -Brief: Multiple files -Master: Multiple files -Old Artwork: Multiple files -INCI: Multiple files The folders already exist in the folder structure
PM-276	Search in Projects	Major	Closed	Fixed	02/04/2012 11:56	13-Apr-12	The search in project page should allow to find by: Material code V number And all the columns in the projects list view.
PM-277	Archive	Major	Closed	Fixed	02/04/2012 12:53	13-Apr-12	There should be a search function for the projects archived. This function should allow to find tasks withing archived projects. The criteria should be : Material code Brand Vnumber Project name Variant The results should be shown as the tasks list, with the same columns + the date of archive. if its necessary the projects can be archived without being zipped.
PM-279	Financial report	Major	Closed	Fixed	03/04/2012 09:44	17-Apr-12	System should generate a cost report like the one in the attachment.
PM-280	Separation prices panel	Major	Closed	Fixed	16/04/2012 14:09	23-Apr-12	In settings is required to have separation tab for the prices It should look like in the attachment. All prices are editable.

PM-324	Status report update	Critical	Closed	Fixed	11/06/2012 12:51	20-Jun-12	Please amend the following columns: H: Change title to Volume instead of Size I: Start date should correspond to the date when the task was created by AM. J: End date change it to Last update M: Filled the cell with the status color that corresponds. Add the Designer name column after AM Change the report title to Complete <Project/Brand name> Status Update instead of All Status Update. For the status report we need two more filters by AM and by Designer The date filter should allow to choose an specific date to generate the report.
PM-325	Individual report generation	Major	Closed	Fixed	11/06/2012 12:56	25-Jun-12	Please add the possibility to generate the Individual reports by project or by brand. This means the system should generate a ZIp that contains all the individual reports for a given project or a given brand.
PM-330	Fields restriction	Major	Closed	Fixed	19/06/2012 12:35	29-Jun-12	Variant name and Project name should not be longer than 10 characters Cluster should not be longer than 4 characters

Table 11. Original user stories extract from JIRA repository

Study case:	MIS – Stolte Packaging
Case description:	Project Management System based on the workflow for artwork development and print management agency. Internal use.
Study context:	.NET and SQL Server 2000
Data sources:	Issue tracker used by the developers: JIRA (http://dev.sflpro.com/) and personal interviews with developers and project owners.
Sample:	36 User stories – 156 Engineering tasks estimated

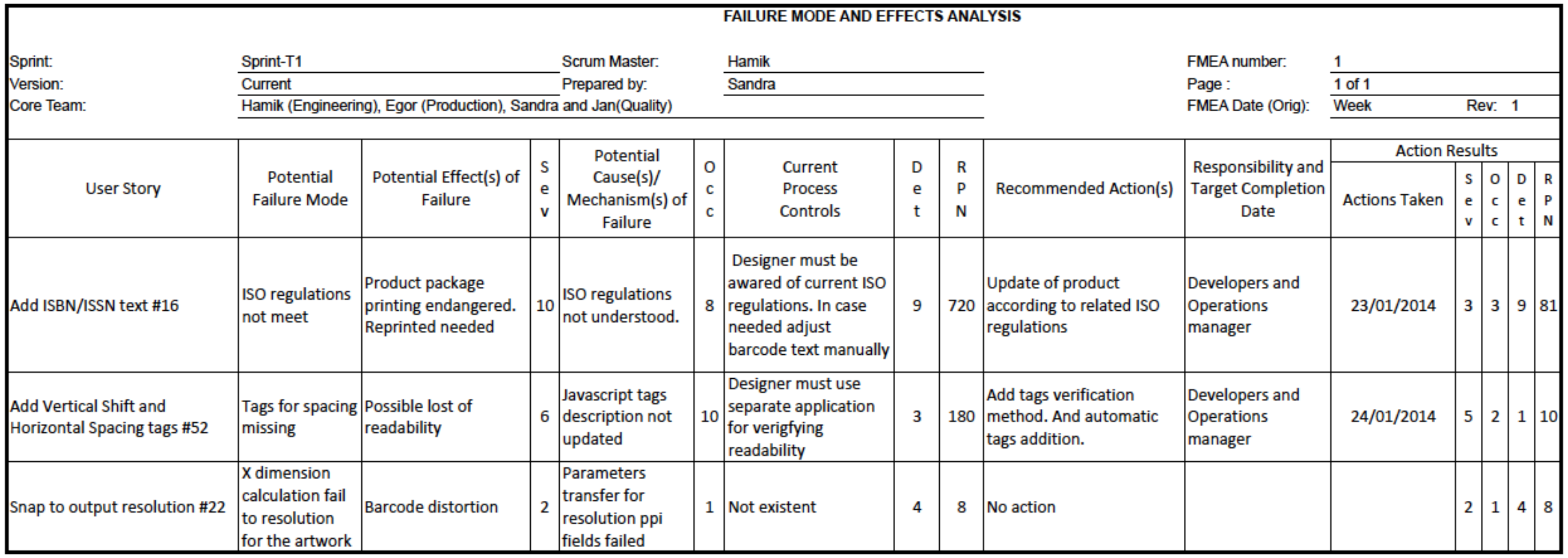
Table 12. MIS Project description details

Due to easy access to information, time of development and low restrictions of confidentiality we were able to work with the second sample mentioned above

(Flexibarcode). The developer team for this project was in-house, this allows to have permanent contact with the developers to get feedback regarding the proposal. The development team allows us to partly implement the methodology and track results.

In order to evaluate results the methodology was implemented only for some parts of the projects. The team chose a group of user stories that will be monitored using FMEA. We proceeded to compare the results of those User stories that were tracked against the ones that were not part of our process.

The figure 19 shows an extract of the FMEA form used in the project.



FAILURE MODE AND EFFECTS ANALYSIS

Sprint:	Sprint-T1	Scrum Master:	Hamik
Version:	Current	Prepared by:	Sandra
Core Team:	Hamik (Engineering), Egor (Production), Sandra and Jan(Quality)		
FMEA number:	1		
Page :	1 of 1		
FMEA Date (Orig):	Week	Rev:	1

User Story	Potential Failure Mode	Potential Effect(s) of Failure	Sev	Potential Cause(s)/ Mechanism(s) of Failure	Occ	Current Process Controls	Det	RPN	Recommended Action(s)	Responsibility and Target Completion Date	Action Results: Actions Taken	Sev	Occ	Det	RPN
Add ISBN/ISSN text #16	ISO regulations not meet	Product package printing endangered. Reprinted needed	10	ISO regulations not understood.	8	Designer must be awared of current ISO regulations. In case needed adjust barcode text manually	9	720	Update of product according to related ISO regulations	Developers and Operations manager	23/01/2014	3	3	9	81
Add Vertical Shift and Horizontal Spacing tags #52	Tags for spacing missing	Possible lost of readability	6	Javascript tags description not updated	10	Designer must use separate application for verigfying readability	3	180	Add tags verification method. And automatic tags addition.	Developers and Operations manager	24/01/2014	5	2	1	10
Snap to output resolution #22	X dimension calculation fail to resolution for the artwork	Barcode distortion	2	Parameters transfer for resolution ppi fields failed	1	Not existent	4	8	No action			2	1	4	8

Fig. 19. FMEA agile project sample

The user stories selected for FMEA use showed an interesting behavior. Most of them showed consequences that may compromise the quality of the product and the timing of the project. Therefore, some of them derivate in separate stories. The developers were able to adjust the plan of the Sprint and schedule first the most critic engineering tasks fitting to the desired length of the Sprint.

Developers found the tool very appropriate for fast planning and accurate identification of additional engineering tasks.

8 Discussion of Results

This dissertation is dedicated to modeling risk management activities within agile software development methodologies like SCRUM and XP. The introduction outlines the motivations for the development of the model. The introduction chapter includes the goals and sub goals defined for this study.

The main goal of this dissertation was to design a methodology style for risk management process applied to agile development methods like SCRUM and XP The literature review illustrate the necessary group of processes to ensure risk management in a project. Risks should be identified, quantified and prioritize. A risk response plan should be prepared, which correspond to the response actions planned for each of the risk previously analyzed.

This work is quite unique due the following aspects:

- Uses GOPRR to combine FMEA and Agile software development
- The use of GOPPR guarantees a practical use. It is supported by a wide variety of commercial tools. The one used for practical purposes in this work was MetaEdit+ from Meta Case.
- The model is not fix to a specific type of project. The model is defined as an ontology, which is not bonded to a specific type of software neither

project. It is suitable to any development in any field where agile methodologies as XP and SCRUM are applicable.

- The model was developed as a set of natural language pieces. Programming can extend the soft definition of concepts.

We find a possibility of prioritization in case of lack of recourses. User stories can be discarded cause of detection ranking or level of occurrence. Giving more time for complex valuable stories or adding more stories to iteration.

Standard agile rely on the iteration concept for solving problems. This implementation provides a complementary approach to track risks and failures.

9 Conclusion

9.1 Summary of dissertation

Reviewing the current state of agile risk management practices we have confirm that the agile approach lacks of formal implementation of risk management activities.

Common risk management practices in XP and SCRUM rely on the concept of incremental development.

As per the goals defined for this thesis, they were fulfilled as follows:

- The agile practices that ensure quality software projects were identified as follows:
    - Principles in SCRUM of Inspection and adaptation. Regular evaluation against expected results are part of the life cycle in a SCRUM project. The SCRUM team is developing under the principle that requirements may and will change at any moment.
    - Sprint retrospective is also present in SCRUM projects. It corresponds to an internal evaluation of the team performance in terms of processes and communication.

  - Mitigation of risks: This task is performed basically turning issues into new features to develop.
  - Real customer involvement in order to fulfill user requirements and share continuous feedback on the ongoing development.
  - Shared code responsibility among the team members. Collective ownership is translated in collective knowledge reducing the risk of truck factor.

  The risk management activities identified n agile methodologies do not follow formal implementation neither cover the basic three aspects of risk management (identification, quantification/evaluation and control/monitoring).

- A methodological approach for RM processes was proposed to be applied in projects developed using XP practices and/or SCRUM methodology. This approach uses the concepts of FMEA to identify, quantify and control risks.
- The methodological approach proposed by this thesis includes few and low effort activities to identify, track and measure risk. These activities are easily added to the normal life cycle of iteration in an agile project.
- The existing method for quality assurance FMEA we supported by several of the ideas compound in the agile approach. It can be considered by nature an agile method.

The work presented in this document provides a formal framework for agile teams to address risk management without jeopardizing the agile nature of the project development.

9.2 Suggestions for further research

There may be a possibility of a quantitative model but this was not considered during this thesis since we believe this goes against the agile approach.

Future work is towards validation of the methodological approach defined in a real case of study integrating the RM processes into an XP/SCRUM project.

Future project could be initiated as an extension of the presented work to be applicable in other agile techniques as Adaptive Software Development, Agile Unified Process, etc.

10 List of abbreviations

IS – Information System	

IS – Information System

RM – Risk Management

FMEA – Failure Mode Effect Analysis

SW – Software

XP – Extreme programming

ET – Engineering task

SQUARE – Software product Quality Requirements and Evaluation

GOPRR – Graphs Object Properties Roles Relationships

SEI – Software Engineering Institute

CMMI – Capability Mature Model Integration

UT – User Test

11 List of Tables

Table 1. Risk Severity Ranking……………………………………………………34

Table 2. Occurrence criteria……………………………………………………35

Table 3. Detection ranking……………………………………………………36

Table 4. FMEA form real example……………………………………………59

Table 5. Concept mapping……………………………………………………65

Table 6. Agile FMEA form……………………………………………………76

Table 7. Response action according to FMEA parameters…………………79

Table 8. Real sample of user stories from web job portal…………………82

Table 9. Flexibarcode Project Sample description……………………………84

Table 10. Project Sample Flexibarcode: User stories and engineering Tasks…..84

Table 11. Original user stories extract from JIRA repository………………90

Table 12. MIS Project description details...90

12 List of Figures

Fig. 1. Projec management triangle..15

Fig. 2. Organization of SQuaRE series of International Standards..................17

Fig. 3. Software quality model categories and attributes................................20

Fig. 4. The Zachman framework..27

Fig. 5. Project Quality Management Overview...28

Fig. 6. Project Risk Management Overview...30

Fig. 7. SCRUM Life Cycle...41

Fig. 8. XP Activity diagram of XP iteration...47

Fig. 9. ADIteration metamodel symbol..67

Fig. 10. ADEngineeringTask metamodel symbol..68

Fig. 11. ADUserStory metamodel symbol..69

Fig. 12. ADUserTest metamodel symbol...70

Fig. 13. RMAction metamodel symbol...71

Fig. 14. RMrisk metamodel symbol...72

Fig. 15. WorkUnit metamodel symbol..73

Fig. 16. Metamodel diagram for Agile Risk Management...............................74

Fig. 17. Agile iteration integrating RM activities..81

Fig. 18. Screenshot extracted from the project repository in GitHub................83

Fig. 19. FMEA agile project sample...92

13 Bibliography

Agile Alliance. (2013). *The Twelve Principles of Agile Software*. Retrieved March 6, 2013, from Agile Alliance: http://www.agilealliance.org/the-alliance/the-agile-manifesto/the-twelve-principles-of-agile-software/

Ambler, S. (2008). *The Agile System Development Life Cycle (SDLC)*. Retrieved March 2012, from Ambysoft: http://www.ambysoft.com/essays/agileLifecycle.html

Awad, M. (2005). *A Comparison between Agile and Traditional Software Development Methodologies*. Dissertation Thesis, University of Western Australia, School of Computer Science and software Engineering.

Banerjee, N. (1995). Utilization of FMEA concept in software lifecycle management. *Software Quality Management. 1*, pp. 219-230. Southampton: Computational Mechanics Publications.

Beck, K. (2004). *Extreme Programming Explained: Embrace Change* (2nd Edition ed.). Addison Wesley.

Beck, K., Grenning, J., Martin, R., & Beedle, M. (2001). *agilemanifesto*. Retrieved 2008, from Manifesto for Agile Software Development: http://agilemanifesto.org/

Bicchierai , I., Bucci, G., Nocentini, C., & Vicario, E. (2012). An Ontological Approach to Systematization of SW-FMEA. In F. Ortmeier, & P. Daniel (Eds.), *Computer Safety, Reliability, and Security* (Vol. 7612, pp. 173-184). Berlin: Springer Berlin Heidelberg.

Boehm, B. (2002). Get ready for Agile Methods, with Care. *Computer , 35* (1), 64-69.

Boehm, B. (1991). Software risk management: principles and practices. *Software, IEEE , 8* (1), 32-41.

Buchalcevová, A. (2009). Research of the Use of Agile Methodologies in the Czech Republic. In *Information Systems Development* (pp. 51-64). New York: Springer US.

Chang, K. (2013). A more general risk assessment methodology using a soft set-based ranking technique. *Soft Computing , 18* (1), 169-183.

Chow T., C. D. (2008). A Survey study of critical success factors in agile software projects. *Journal of Systems and Software , 81* (6), 961-971.

CMMI Product Team . (2010). CMMI® for Development, Version 1.3 . Carnegie Mellon University. . Retrieved from http://www.sei.cmu.edu/reports/10tr033.pdf

Ern, B., Nguyen, V., & Noll, T. Characterization of Failure Effects on AADL Models. *LNCS 8153* (pp. 241-252). Berlin: Springer Verlag.

Fitsilis, P. (2008). Comparing PMBOK and Agile Project Management Software Development Process. *Advances in Computer and Information Sciences and Engineering* , 378-383.

Ford Motor Company. (2008). *FMEA Handbook version 4.1.* Design Institute. Dearborn: Ford Company.

Gavin Robbins Ltd. (2013, March). *Process FMEA example.* Retrieved June 11, 2014, from Professional FMEA Training, Facilitation & Software: http://www.fmea.co.uk/example/index.html

Glazer, H. D. (2008). *CMMI or Agile: Why Not Embrace Both! (CMU/SEI-2008-TN-003).* Carnegie Mellon University. Software Engineering Institute.

Glazer, H., Dalton, J., Anderson, D., Konrad, M. D., & Shrum, S. (2008). *CMMI or Agile: Why Not Embrace Both!* Carnegie Mellon University. Pittsburgh: Software Engineering Institute.

Higuera, R., & Haimes, Y. (1996). *Software Risk Management Technical Report.* Carnegie Mellon University. Pittsburgh: Software Engineering Institute.

International Standard Organization (ISO). (2011). *Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models.* ISO.

Jamieson, J. M., & Fallah, M. H. (2012). Agile Quality Management Techniques. *Software Quality Professional , 14* (2), 12.

Jensen, B., & Zilmer, A. (2003). Cross-Continent Development Using Scrum and XP. *4th International Conference, XP 2003* (pp. 146-153). Berlin: Springer Berlin Heidelberg.

Kelly, S. (1997). *Towards a Comprehensive MetaCASE and CAME Environment: Conceptual, Architectural, Functional and Usability Advances in MetaEdit+*. Ph.D. Thesis, University of Jyvaskyla.

Kelly, S., & Pohjonen, R. (2013). Dynamic Symbol Templates and Ports in MetaEdit+. *SPLASH Workshop on Domain-Specific Modeling*. Indianapolis,.

Kiniberg, H. (2007). *Scrum and XP from the Trenches.* Retrieved 2014, from InfoQ: http://wwwis.win.tue.nl/2R690/doc/ScrumAndXpFromTheTrenchesonline07-31.pdf

Lauritsen, T., & Stålhane, T. (2005). Safety methods in software process improvement. *12th European conference on Software Process Improvement* (pp. 95-105). Berlin: Springer-Verlag Berlin.

Levine, L. (2005, May). *Carnegie Mellon Software Engineering Institute*. Retrieved 2012, from Reflections on Software Agility and Agile Methods: Challenges, Dilemmas, and the Way Ahead.: http://www.sei.cmu.edu/library/assets/reflections.pdf

Lindvall, M., Basili, V., Boehm, B., Costa, P., Dangle, K., & Shull, F. Empirical Findings in Agile Methods. *Second XP Universe and First Agile Universe Conference Chicago,* (pp. 197-207). Berlin: Springer Berlin Heidelberg.

Lindvall, M., Basili, V., Boehm, B., Costa, P., Dangle, K., & Shull, F. (2002). Empirical Findings in Agile Methods. *Second XP Universe and First Agile Universe Conference Chicago* (pp. 197-207). Berlin: Springer Berlin Heidelberg.

Marcal, A., De Freitas, B., Soares, F., & Belchior, A. (2007). Mapping CMMI Project Management Process Areas to SCRUM Practices. *Software Engineering Workshop, 2007. SEW 2007. 31st IEEE* (pp. 13-22). Columbia: IEEE.

Martisson, J. (2003). Maturing XP trought the CMM. *Extreme Programming and Agile Processes in Software Engineering. 2675*, pp. 80-87. Genova: Springer Berlin Heidelberg.

Merunka, V. (2009). Software Engineering. Prague.

MetaCase. (2014). *MetaEdit+ 4.0 documentation updates*. Retrieved January 2014, from MetaCase: http://www.metacase.com/support/40/manuals/

Moran, A. (2014). *Agiles Risk Management*. Zurich: Springer International Publishing.

Nelson, C., Taran, G., & Hinojosa, L. (2008). Explicit Risk Management in Agile Processes. *Agile Processes in Software Engineering and Extreme Programming. 9*, pp. 190-201. Limerick: Springer Berlin Heidelberg.

Nyfjord, J. (2007). Commonalities in risk management and agile process models. *Proceedings of the 2nd International Conference on Software Engineering Advances (ICSEA'07)* (p. 18). France: IEEE Computer Society Press.

Nyfjord, J. (2008). Integrating Risk Management with Software Development: State of Practice. *Proceedings of The International MultiConference of Engineers and Computer Scientists . 1*, pp. 878-884. Hong Kong: Newswood Limited.

Nyfjord, J. (2008). *Towards integrating agile development and risk management.* Stockholm University, Faculty of Social Sciences, Department of Computer and Systems Sciences. Kista: Institutionen för data- och systemvetenskap (tills m KTH).

Přı́brský́, M. (2012). *Kvantifikovaný́ přístup k jakosti informačního zabezpečení pro podporu evaluace informačních technologií.* PhD Thesis, University of Life Sciences in Prague, Department of Information Engieering, Prague.

Pekka, A., Outi, S., Jussi, R., & Juhani, W. (2010). Agile Software Development Methods: A Comparative Review. In *Agile Software Development* (pp. 31-59). Finland: Springer Berlin Heidelberg.

Picka, M. (2004). Metamodeling and development of information systems. *Agricultural Economics , 50*, 65-70.

Project Management Institute PMI. (2013). *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)* (5th Edition ed.). Project Management Institute.

Raspotnig, C., & Opdahl, A. (2012). Supporting Failure Mode and Effect Analysis: A Case Study with Failure Sequence Diagrams. *Requirements Engineering: Foundation for Software Quality. 7195*, pp. 117-131. Essen: Springer Berlin Heidelberg.

Santana, C. G. (2009). Agile Software Development and CMMI: What We Do Not Know about Dancing with Elephants. *10th International Conference, XP 2009, Pula, Sardinia, Italy, May 25-29, 2009. Proceedings* (pp. 124-129). Springer Berlin Heidelberg.

Stamatis, D. (2003). *Failure Mode and Effect Analysis: FMEA from Theory to Execution* (2nd Edition ed.). Milwaukee: Amer Society for Quality.

Standish Group. (2013). *CHAOS MANIFESTO 2013 - VersionOne.* Retrieved 05 17, 2014, from VersionOne: http://www.versionone.com/assets/img/files/CHAOSManifesto2013.pdf

Stoll, P., & Wall, A. (2009). *Software engineering feauturing the Zachman taxonomy.* Mälardalen University, Computer Science and Electronics, Västerås.

Sutherland, J., & Schwaber, K. (2011, October). *The SCRUM guide. The Definitive Guide to Scrum: The rule sof the game.* Retrieved March 2012, from Scrum.org: https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum_Guide.pdf

Torchiano, M., Ricca, F., & Marchetoo, A. (2011). Is my project's truck factor low?: theoretical and empirical considerations about the truck factor threshold. *2nd International Workshop on Emerging Trends in Software Metrics* (pp. 12-18). New York: ACM.

Turk, D., France, R., & Rumpe, B. (2002). Limitations of Agile Software Processes. *Third International Conference on eXtreme Programming and Agile Processes in Software Engineering.*

Von Scoy, R. (1992). *Software Development Risk: Opportunity, Not Problem.* Carnegie Mellon University. Pittsburgh: Software Engineering Institute.

Wells, D. (2013, October). *Extreme Programming*. Retrieved 2014, from Extreme Programming: A gentle introduction: http://www.extremeprogramming.org/

Zachman, J. (2008). *About The Zachman Framework™*. Retrieved 2014, from Zachman International®: https://www.zachman.com/about-the-zachman-framework

Zachman, J. (2011). *The Zachman Framework Evolution*. Retrieved 2014, from Zachman International®: https://www.zachman.com/ea-articles-reference/54-the-zachman-framework-evolution