International Journal of Advanced Multidisciplinary Scientific Research (IJAMSR) ISSN: 2581-4281
www.ijamsr.com CrossRef: https://doi.org/10.31426/ijamsr

MANAGING RISKS: CHALLENGES IN ADOPTION OF AGILE SOFTWARE DEVELOPMENT APPROACH

Thesis Submitted to

CMJ UNIVERSITY

In partial fulfillment for the award of the Degree of

DOCTOR OF PHILOSOPHY

IN

COMPUTER SCIENCE

By

SHABNAM ARORA

(Regd.No. 80187210105273)

Under the Research Supervision of

DR. Y. RAMA DEVI

Professor

DEPARTMENT OF COMPUTER SCIENCE

CMJ UNIVERSITY

SHILLONG, MEGHALAYA, INDIA-793003

DECEMBER – 2012

International Journal of Advanced Multidisciplinary Scientific Research (IJAMSR) ISSN: 2581-4281
www.ijamsr.com CrossRef: https://doi.org/10.31426/ijamsr

DECLARATION

This is to certify that the Thesis entitled **"MANAGING RISKS: CHALLENGES IN ADOPTION OF AGILE SOFTWARE DEVELOPMENT APPROACH"** submitted to **CMJ UNIVERSITY**, is a partial fulfillment for the award of the Degree of **Doctor of Philosophy in Computer Science** is my unique work under the supervision of **Dr. Y. Rama Devi, Professor, Department of Computer Science, CMJ University, Shillong, Meghalaya, India**. The thesis has not been submitted before for the award of any degree, diploma or similar title of this or any other University.

Place: Shillong (Meghalaya)

(SHABNAM ARORA)

(Regd.No. 80187210105273)

International Journal of Advanced Multidisciplinary Scientific Research (IJAMSR) ISSN: 2581-4281
www.ijamsr.com CrossRef: https://doi.org/10.31426/ijamsr

CERTIFICATE

This is to certify that the thesis entitled **"MANAGING RISKS: CHALLENGES IN ADOPTION OF AGILE SOFTWARE DEVELOPMENT APPROACH"** submitted to CMJ University, Shillong, Meghalaya, India by **Shabnam Arora,** for the award of **Doctor of Philosophy in Computer Science** is a record of research work done under my supervision and guidance. This thesis has reached the standards fulfilling the requirements of the regulations for the Degree and it was not previously formed the basis for any other degree or diploma and I additionally affirm that the Thesis speaks to an autonomous work with respect to the Candidate.

Place: Shillong (Meghalaya)

(DR. Y. RAMA DEVI)

Research Supervisor

International Journal of Advanced Multidisciplinary Scientific Research (IJAMSR) ISSN: 2581-4281
www.ijamsr.com CrossRef: https://doi.org/10.31426/ijamsr

Acknowledgements

Foremost, I would like to express my sincere and whole hearted thanks to my beloved Research Supervisor **Dr. Y. Rama Devi, Professor, Department of Computer Science, CMJ University, Shillong, Meghalaya, India** for her consistent consolation in the indictment of this investigation and in the preparation of this Thesis. I am extremely grateful to her for her sharp and individual intrigue, valuable feedback, and capable direction or more but for her convenient guidance with regard to mine exploration and the innovation of research paper in universal gathering.

I owe a commitment of gratefulness to all Teaching and Non-Teaching staff in the Department of Computer Science, CMJ University, for their moral help and encouragement and my grateful because of authorities of Libraries of CMJ University.

I would like to express my sincere and whole hearted thanks to **Dr Pradeep Kumar, Reader, Department of Computer Science & Engineering, Guru Jambeshwar University of Science & Technology Hissar, Haryana India** for their consistent consolation in the indictment of this investigation and in the preparation of this Thesis.

I would like to express my heartfelt gratitude to my husband **Sh. Raman Kwatra** for her tremendous support.

I wholeheartedly thank to my beloved Father- in -law **Sh. Nand Kishore** and my Mother- in -Law **Smt. Saroj** for giving me the strength that I required to pursue my dreams. I furthermore also thank to my Sister- in -law **Mrs Rachna Gulati** for their ethical help and support all through the work.

Finally, I obliged to the authorities of the **CMJ University** for providing me an opportunity and necessary facilities for carrying out this research work.

International Journal of Advanced Multidisciplinary Scientific Research (IJAMSR) ISSN: 2581-4281
www.ijamsr.com CrossRef: https://doi.org/10.31426/ijamsr

DEDICATED TO MY PARENTS

ANITA RANI

&

GORDHAN DASS

International Journal of Advanced Multidisciplinary Scientific Research (IJAMSR) ISSN: 2581-4281
www.ijamsr.com CrossRef: https://doi.org/10.31426/ijamsr

Table of Contents

International Journal of Advanced Multidisciplinary Scientific Research (IJAMSR) ISSN: 2581-4281
www.ijamsr.com CrossRef: https://doi.org/10.31426/ijamsr

MANAGING RISKS: CHALLENGES IN ADOPTION OF AGILE SOFTWARE DEVELOPMENT APPROACH

CHAPTER - 1

INTRODUCTION

International Journal of Advanced Multidisciplinary Scientific Research (IJAMSR) ISSN: 2581-4281
www.ijamsr.com CrossRef: https://doi.org/10.31426/ijamsr

MANAGING RISKS: CHALLENGES IN ADOPTION OF AGILE SOFTWARE DEVELOPMENT APPROACH

CHAPTER - 1

INTRODUCTION

1.1 Introduction

The usage of software has grown rapidly. It is playing a major role in all parts of life as it is being used in vat amount of products like cars, mobile phones, televisions, etc. In today's world, software development is a complex task because of frequently changing customer needs. In order to stay competitive, companies, should react according to changing requirements in a rapid and flexible manner. Due to this development, agile methods came into practice.

The roots of agile software development were formed in mid-1990s (e.g. Extreme programming, 1996). But, agile software development received much attention after the release of the agile manifesto in 2001. Agile methods are a set of light-weighted methods such as Extreme Programming (XP, Scrum, Crystal Clear, Adaptive Software Development (ASD), Feature Driven Development (FDD) and Dynamic Systems Development Method (DSDM). All agile methods are driven by values such as (1) "Individuals and interactions over processes and tools", (2) "Working software over comprehensive documentation". (3) "Customer collaboration over contract negotiation", and (4) "Responding to change over following a plan."

Due to agile driven values and principles, agile methods got popularity and most of the organizations are migrating to agile software development from plan-driven development. A number of studies have now been published describing the broad benefits that flow from adoption.

Unfortunately, as outlined in (Taylor 2007) " A cursory glance at some of the agile literature, or hearing a short talk on the subject, can give the mistaken belief that an agile development approach will be straight forward to adopt and will result in instant success." The adoption of agile methods, however, does come with risks. These risks must be carefully identified and managed within any potential agile- adopter, especially a small adopting organization that is less able to absorb the impact of any significant level of failed system development experimentation.

Risk management has become recognized as a best practice in the software industry (Wiegers, 1998). Controlling risks, improves essential software development features such as product quality, precision planning and cost efficiency (Englund, 1997) (Ropponen and Lyytinen, 2000). For this reason, the inclusion of risk management in software development is an important factor to consider if one wishes to achieve project success (Kontio, 1999).

Much research has been conducted in the software risk management field in the past decades. However, relatively little research has been performed to integrate risk management with development. The spiral model, based on a risk- driven and cyclic approach, is one suggestion for making software development more effective using risk management. However, despite the fact that it was already pioneered in 1988 (Boehm, 1988), it has been only partially realized.

Its cyclic character has been adapted by many current development approaches, such as iterative and agile development. It's risk-driven approach, on the other hand, has not been as influential. Still, development and risk management processes live somewhat isolated lives (Bohner and Coram, 2005) (Sliger, 2006).

Agile models claim to be risk-driven (Beck, 2004) (EPF, 2007) (Scrum, 2003). They state that their iterative approach enables continuous attention to risks and that risk can be reduced by practices such as continuous software integration and early testing (Beck, 2004). In reality, however the agile development models implement few risk management practices (Armenta and Gaono, 2008) (Bohner and Coram, 2005) (Sliger, 2006). Hence, there is clearly a gap well worth investigating bearing in mind the fact that risk management is considered best practice in contemporary Software engineering.

1.2 Research Problem

Despite the fact that risk management is of crucial importance for software project success, very few models have been found that explicitly relate risk management with agile development processes. One reason is the fact that extending the agile model with additional development practice concerns a highly controversial problem in itself. It is often argued that one of the most difficult impediments to extend the agile model concerns the conflict that emerges from trying to merge the agile process with any existing standards (Boehm and Turner, 2005). The question that arises is: “How can one merge agile, lightweight processes with standard industrial processes without killing agility?”

Although controversial, the demand for such solutions in the industry can no longer be neglected. This is also evident in recent publications addressing enterprise agility (Leffingwell, 2007) (Schwaber, 2007). Many large organizations show a genuine interest in using the agile model, but due the experienced scalability problems they are hesitant. Essentially, they are challenged by the lack of guidelines for building up the agile process according their needs, where one missing building block is risk management.

Solutions for introducing risk management in agile development have been proposed, for instance by Li at al. (2006) and Sliger (2006). Unfortunately, they are limited. For instance, there implement only a subset of risk management practices or focus only on risk management in selective phases of the overall development model. Risk management, however, is a continuous organization-wide process (SEI, 2008) (Williams et al., 1999). Hence, it needs to be addressed on an organization-wide level.

Considering the state of art, the research problem addressed in this thesis is multi-faceted. The problem not only concerns the lack of explicit risk management practices in agile development, but also the extent to which risk is managed continuously on an organization-wide bases. It also concerns the conflict that emerges from trying to merge the agile process with existing standard industrial processes without compromising agility.

1.3 Motivation for Research

The idea of being proactive to change is becoming increasingly important across all industries due to an increase in project failures. There are infinite examples across the literature of project failures

and media reports of "major engineering and development projects running late or exceeding their budget" (Coppendale 1995). Risk management has been described as the "activity of identifying and controlling undesired project outcomes proactively" (Smith and Merritt2002). One of the main reasons highlighted for the increasing failure in software projects is that "managers are not taking prudent measures to assess and manage the risks involved" in their projects (Keil, Cule et al. 1998).

According to Smith and Merritt (2002), a lack of proactiveness with regard to risk management is one of the main reasons for project delays, increased project running costs and/or eventual project failures, While there is extensive literature on risk management, research in relation to risk management in agile SD projects is non-existent. This is surprising considering how quickly agile methods are being adopted in SD.

Many books on agile methods "have remarkably little to say about how a development team determines the risks it faces, prioritize them or takes action to negate their effects" (Smith and Pichler 2005). Essentially, agile methods must "tailor conventional risk management approaches meant for years- long projects into a risk driven agile iteration lasting only seven to thirty days" (Smith and Pichler 2005). How aile projects go about doing this remains unknown.

The primary objective of this research is thus to develop a better understanding of risk management practices in agile SD projects and the level of formality with which these practices are executed. Specifically, this research focuses on three main elements of risk management, namely risk identification, estimation and evaluation.

1.4 Research Objectives

Risk management is a key project success factor. Agile development models do not encompass risk management as required by many development organizations. Hence the main objectives of this thesis are as follows:

- To identify existing agile practices adopted for software development.
- To outline a model integrating the risk management and agile models on an organization-wide basis. The model is expected to:
- Provide software organizations practical guidance on how to integrate the risk management and agile development models.
- Provide a reference model for software organizations to examine their risk management practice and see how they compare to the reference model.
- Provide guidance for reasoning about agility with respect to risk management.

1.5 Theoretical Framework

There are mainly three development paradigms related to software development, namely plan-driven software development, agile software development and lean software development. This section provides a brief description about these three paradigms.

Plan- driven methods are heavy weighted methods. These are mainly focused on heavy documentation and the sequential execution of development activities. The best known plan-driven methods are Waterfall model and Rational unified process. These methods are

suitable for stable requirements. The present market is becoming more dynamic, so companies need to react with frequently changing requirements or customer needs. React to change quickly is difficult with plan driven software development. For example, the long lead-time to plan-driven projects lead to a high amount of requirements being discarded as the requirements become disused due to changes in customer needs. This is due to lack of end-user involvement, poor requirements, unrealistic schedules, lack of change management, late testing, inflexible and bloated processes. Managers and programmers have faced a lot of challenges during with the traditional way of software development.

In order to overcome all issues related to plan- driven approaches agile methodologies emerged as a strong solution in the late 1990s and early 2000s. Agile software development is different from plan-driven software development in many ways. In plan- driven development, the development process carried out by sequential phases i.e., the complete requirements specification is created in requirements specification phase followed by design, coding and testing. While coming to agile software development, it focuses on high-level plan for the overall development process with detailed plans only for the current iteration and all phases are conducted in the iteration.

Agile development contains several recurring themes like: simplicity, short iterations, close collaboration with customers and within the development team, frequent deliveries, refactoring, tight teaming, open communication, Proactive plan management and Continuous testing. Well known agile methodologies are extreme Programing (XP, Scrum, Crystal Clear, Adaptive Software Development (ASD), Feature Driven Development (FDD) and Dynamic Systems Development Method (DSDM). Each of these processes contains

several recurring themes called agile practices. The systematic review performed by Dyba and Dingsyr shows that agile software development has received much attention from the research community.

While agile software development became more and more popular, lean software development has received wide interest in practice with the publication of Poppendieck and Poppendieck. Lean is basically manufacturing and product development approach. It is a Toyota. Production System approach, lean manufacturing led to enormous performance improvement in manufacturing cars context at Toyota. Lean is a continuous process through systematic analysis that focuses on waste identification; waste is everything that does not lead to customer value. Poppendiecks were inspired by lean development and translated the lean principles and practices to software development. The research work of Mary and Tom Poppendieck are the main sources to learn about lean software development.

The software development paradigms agile and lean seem similar in their goals that are focused on the customers and respond to their needs in rapid manner. The recent work of Kai Petersen proved that Lean is agile because it includes all the principle of agile, but agile is not lean because it does not emphasize the E2E focus on its principles.

This thesis only focuses on the agile software development and its practices, due to scope and time constraints. However, given that agile is inspired by the ideas of lean product development, both agile and lean development share similarities.

1.5.1 Agile Methodology

To respond to the software crisis by the "heavy-weight," process-focused waterfall approach, some software professional summarized their best practices and proposed "lightweight" change-tolerant software development methods such as extreme Programming (XP), scrum, FDD (Feature Driven Development), ASD (Adaptive Software Development). These methods share many similarities and their founders agreed to label them as agile software development methodology. Some agile method founders an agile alliance and published the manifesto of agile software development. The manifesto contains a set of agile values and guiding principles that describe and characterize agile methodology.

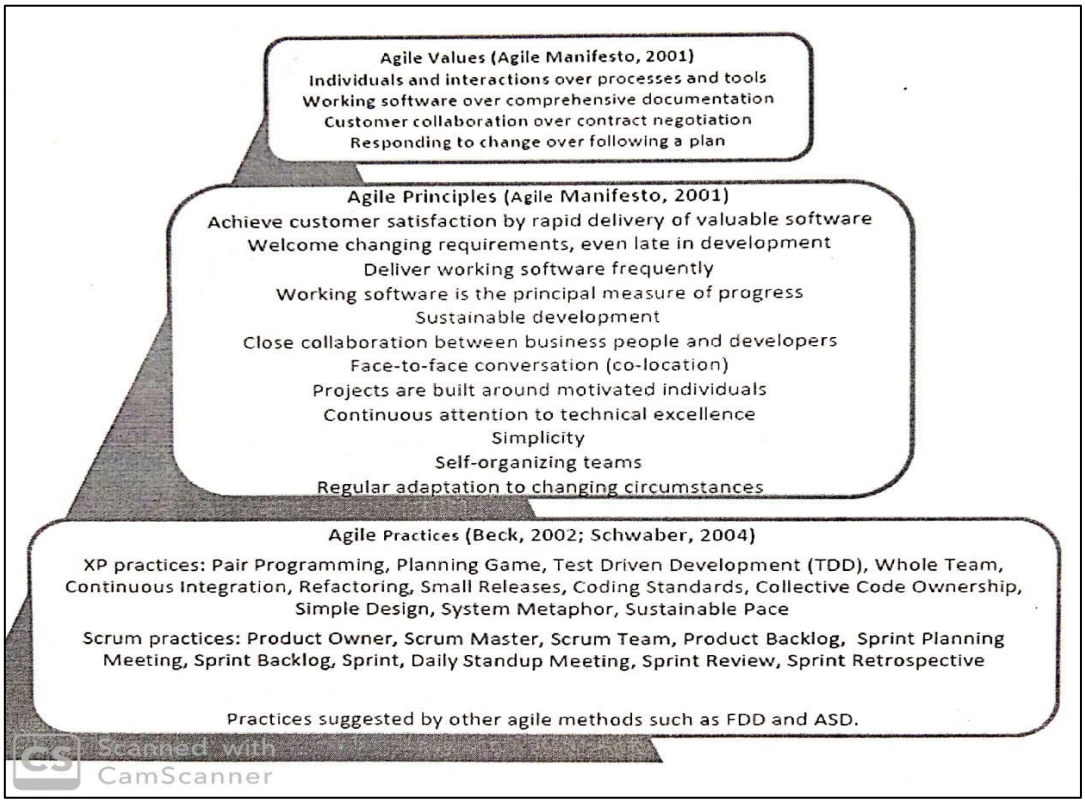


FIG.1.1: AGILE VALUES, PRINCIPLES, AND PRACTICES

Agile Values:

Agile reveal the philosophy behind the agile software development approach. Instead of viewing software development as an engineering process where optimization and control are required, agile methodologists see software development as a process of learning and innovation where responsiveness and Flexibility is a must in order to cope with uncertainty and changes (Highsmith and Cockburn, 2001; Nerur and VenuGopal, 2007). The values of agile methodology are expressed as follows in the Agile Manifesto:

We are uncovering better ways of developing software
by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions	*over*	*processes and tools*
Working software	*over*	*comprehensive documentation*
Customer collaboration	*over*	*contract negotiation*
Responding to change	*over*	*following a plan*

That is, while there is value in the items on the right,
we value the items on the left more[53].

It does not mean that there is no documentation done on agile projects, but it does mean that if a customer were to choose between documentation and another feature in the software the customer would most likely chose the feature, and as such, agile methodologies say that features should be valued as more important than the documentation.

The values may give rise to questions such as what does it mean and it can mean everything from how I is interpreted. Martin Fowler and Jim Highsmith (co-authors of the Agile Manifesto) commented on the values in an article in 2001. They pointed out that it was as much as 17 experienced and recognizes software development "gurus" that agreed with the statement in the first place which may be the first aspect that should be noted with it. The word uncovering was selected to indicate that the members of the Agile Alliance do not have all the answers, nor do they subscribe to the silver-bullet theory. By doing it indicates that the members practice these methods themselves in their own work and by helping others do it show that the idea is to further the own knowledge through the helping of others.

Ken Schwaber (a proponent of SCRUM) told of his days of selling tools to automate comprehensive "heavy" methodologies. Impressed by the responsiveness of Ken's company, Jeff Sutherland (Scrum) asked him which of these heavy methodologies he used internally for development.

"I still remember the look on Jeff's face," Ken remarked, "when I told him, 'None—if we used any of them, we'd be out of business!".

Each bullet point states a preference in the first segment and is followed by something that is of lesser importance. The distinction between them is where the heart is agility lies. The latter segment is however not without importance.

"Yes, we value planning, comprehensive documentation, processes and tools. That is easy to say. The hard thing is to ask 'what do you value more?!" Roy Singham at Thought Works, about that it is the edge cases, the hard choices, which interest him.

It is recognized by the Alliance that processes and tools are important, but it is also recognized that the interaction between skilled individuals in a project is of even greater importance. Comprehensive documentation may be important in some projects, but it can never be more important that the final product and delivering working software. Every team should decide their specific project.

Internal project charters or external legal contracts are not believed to be the best way to create an understanding of each other and to understand and deliver what the customer wants. Only through on-going collaboration can any real understanding be created and therefore is contract negotiation said to be insufficient for the purpose.

Even successful projects very rarely deliver what was planned in the beginning. Instead, they may be considered to be successful because they were agile enough to respond to the changing requirements throughout the process. A fixed plan may even become counterproductive for a project if it is not allowed to change to respond to external changes.

The values have since been altered numerous times and interpreted differently by various practitioners to suit needs of individual projects which is something agile methodologies encourage since an understanding that no two projects are exactly the same is essential. The fundamental ideas, however remain the same as those in the Agile Manifesto as the work continues to evolve in the software development community.

Agile Principles:

The agile principles are more specific rules that reflect and support the agile values and govern the software development activities and practices. These rules are specific enough to provide guidance and general enough to give software professionals leeway to choose and tailor software practices to fit their needs.

Principles of Agile Methodology According to the Agile Manifesto:

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face- to – face conversation.
- Working software is the primary measure of progress. Agile processes promote sustainable development.
- The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

- Continuous attention to technical excellence and good design enhances agility.
- Simplicity—the art of maximizing the amount of work not done—is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly

Agile Practices:

The manifesto doesn't specify a set of agile practices because doing so will be against the core values of agile philosophy. Agile methods such as extreme Programming (XP), Scrum, Lean, and Feature Driven Development (FDD) generally specify a set of agile practices that have a certain synergy. There has been a debate over whether agile methods can be tailored. Some methodologies insist that their methods have to be used as a coherent whole in order to achieve agility (Beck and Andres, 2004; Schwaber and Beedle, 2002), while others suggest that agile methods can be tailored or combined to address development needs (Fitzgerald, Hartnett and Conboy, 2006; McBreen, 2003).

A common practice is to combine XP and Scrum because XP practices such as pair programming and test-driven development specifies the development processes well, while Scrum activities such as sprint and daily standup meetings contribute to project management. Systems development methodologies (SDM) are rarely used in totality without alteration; instead, software practitioners have tailored software development methodologies in a variety of

ways to fit the special needs or different organizations and different projects (Fitzgerald, 1996, 1998 and 2000; Fitzgerald, Russo and O'Kane, 2003; Russo and Wynekoop, 1995). Method tailoring has been a common practice.

Fitzgerald (2000) reported that practitioners did not adopt formalized methodologies in their prescribed form and only 6% of surveyed organizations followed a methodology rigorously; practitioners customized system development methodology in a pragmatic way and methodology-in-action was uniquely enacted foe each development project. Similarly, Russo (1995) found that 85% of the surveyed organizations adapted the system development methodology on a project by project basis; software professionals view methodology as a general framework of phases pr activities and the decisions regarding what development activities to perform is typically made at the project-team level.

Agile methods are no expectations with regard to method tailoring and customization. Even though some argue that agile methods have to be used in entirety to achieve promised benefits, the more popular belief is that agile methods can be applied a la carte or uniquely combined to address the needs of the development context (Fitzgerald, Hartnett and Conboy, 2006).

Boehm and Turner (2003 and 2004) suggested that developers should use risk assessment to find the: sweet spot" for balancing agile and plan-driven methods; an example of this type of method tailoring is the combination of the ISO 9000 with the XP. Another way of method tailoring is to select and combine agile practices suggested by the different agile methods. An example is the selection and combination of XP and scrum practices to improve both the

development processes and to strengthen the project management at Intel Shannon, in Ireland (Fitzgerald, Hartnett and Conboy, 2006).

1.5.2 Criticisms and Risks with Agile Methods

Even the fiercest proponents of agile methodologies say that "using agile is not for everyone" and that it is no silver-bullet theory. Various types of criticism from different directions, targeting different aspects of the agile approach have emerged as the approach has been renowned and more and more widely used, The proponents of agile say that in many cases it is most often a lack of understanding agile that is creating the problems and that anyone contemplating to try an agile approach must understand that it is no quick fix that is appropriate in any type of situation.

Gain The Support Of Upper Management

It is of utmost importance before embarking on any agile project methods to have the understanding and support of high level management. Without the support of higher management the likelihood of succeeding with a project is dramatically decreased. However, the agile approach represents a fundamental culture change for high level, and executive, management with, for example, one or more of the following; less documentation, possibly no set release dates and no set features for every release as well as no predictive, detailed, planning – since this goes against the agile way of thinking- and that in turn goes against most management approaches, which makes it complicated. Without the understanding and support of high level management it is not likely to succeed with any agile project.

The Standish Group, which produced the CHAOS Report, outlines the top criteria for successful projects and executive support is stated as the most important criteria to have. The large companies ABB, Daimler Chrysler, Motorola and Nokia through collaboration compared their experiences with agile pilot projects and this aspect of trust and support from upper management's one issue that was highlighted. Although all the companies knew of reports of many agile success stories they needed to know whether agile practices actually would work for them and whether a large company with established standard processes could at all adopt agile methods to develop large, complex, safety- critical systems that would be maintained for decades.

The studies did show similar success rates from all four companies; that agile practices could help to bring agility to large companies with traditional development processes, there were reports that final products exhibited higher quality than it had previously, that flexibility improved in responding to changing requirements quicker, implementation was finished quicker alongside other positive aspects. However, it was pointed out that it had to be done by assessing the situation and chosen projects well beforehand and well understand the different work practices and the changes they would bring.

Surveys with the results among some of the developers showed that +80 percent of the respondents/team members thought the team morale had increased, that they felt more confident with the work from the XP-influenced pair programming (than had they done the same work alone), that the learning curve for new engineers were dramatically shortened, there was a general higher confidence in the quality of the design and the code. All those pilot projects were said

to have succeeded in improving one or more of the following; customer satisfaction, quality, productivity, and cost.

However, important observations were made that in several cases did agile methods indeed clash with the traditional established methods and values, which in some cases resulted in double work being done (for example double documentation to fulfil different level of management requests, double quality management tests), the pilot projects also reported clashes with the traditional Change Control Board (CCB) who had decided to make a decision on a change request and by the time the decision had been made by the CCB the design of the product had already changed to the extent that more extensive changes had to be made in order to accommodate the change decided upon by the CCB (thus simultaneously decreasing the agility for the project), and conclusions from this were that more work had to be put into "integrating the agile approach into the existing ones" since the integration issue seemed to be more a cause for problems rather than the agile influenced approach itself.

The importance of trying out an agile approach on the right type of project is essential in order to gain success and also understanding of higher management. If not suitable project is chosen for a pilot project, it may not be a good idea to try a new agile approach at all according to Martin Fowler. He suggests a smaller project, preferably of not a too high critical factor, but still slightly more critical "than what is comfortable" or no one will care whether the project succeeds or not, which also eliminates the incentive to try and change existing processes for new agile methods.

Communication Issues in The Team

XP and Scrum do not specifically emphasis the often difficult task of facing the on-site customer, prioritizing the requirements and features not the gathering of requirements, other methods may be needed to assist and complement those aspects of possible problems.

Communication between Teams

One aspect of an agile approach the pilot projects at ABB, Daimler Chrylser, Motorola and Nokia discussed above was that larger companies per definition often have larger projects with more project members often based on various different physical locations. This was concluded as one problem that remained to be solved when working with agile project practices that emphasize real time, direct, face-to-face communication in preferably small teams. The way to solve this was partly by minimizing the need for inter team communication as much as possible and partly by organizing small workshops with representatives from all smaller teams at the beginning of the project, and later at regular intervals, to bring everyone up to speed with the iterative progress everyone (every small team) was at.

This is also an approach Jim Highsmith suggests for large projects separated into smaller teams and/or situated in multiple physical locations which often can be the case for large or global companies. However Highsmith writes that this should be an exception rather than the rule as face-to-face communication and working physically together in the same area cannot be replaced with, however many, workshops. As concluded in the pilot projects at ABB, Daimler Chrysler, Motorola and Nokia is this a problem, of inter team communication that remains to be solved, although there are agile

projects that are reportedly effective in large projects of +100 team members over several continents (one example mentioned is ThoughtWorks).

No Detailed Planning

The workshop's (mentioned above) agendas should not be set beforehand in a predictive way, but should only be addressing the iteration at hand and plan for the next iteration to follow, in order to support an agile approach for every team. Again, this poses a cultural change to most management theories -- to not have a set plan for every stage ahead in a project; it directly goes against the idea of the traditional approach with detailed plans for every gate he project reaches. This can cause hesitation among various management levels to use an agile approach.

Staff Turnover

In large companies with large projects that may run over a period of several years the staff turnover is inevitably a serious problem for agile projects. Agile methods rely on the involved individuals to form a project team with its collective knowledge and that the most effective way to spread information from one individual to another individual is to have those individuals based physically together, and the aim is to minimize ineffective work by drastically cutting back on the amount of unnecessary documentation that no one ever reads in order to rather spend that time more effectively improving the product.

Thus can the agile approach be counterproductive for itself in a large project in this sense, but there are ways to lessen the effects of the loss of an important project member although it can of course not be

minimised completely. One way is to use team rotation which also generally increases knowledge and understanding throughout the team for all the various work assignments in the team, this also increases individual development and skills among the team members which makes it easier for someone to take over somebody else's job duties should it be necessary.

Another way is to make sure the produced code is kept 'clean' and easy to comprehend and maintain which would make it easier for any new member in the team to quicker come up to speed with previous progress. However these approaches would not have an affect should there be a loss of customer or a customer representative that has been a team member. One approach can instead be to tailor the approach to the individual project (which agile encourages) and when needed keep some of the traditional practices of detailed documentation, though keeping in mind that it should be kept to a minimum if the agility of the processes are not to be too effected.

Hostile Towards Problem Solvers

Skowronski, V. (2004) speaks of how agile methods can "marginalize problem solvers" and that agile methods may be hostile towards the best programmers by encouraging continuous team-work, saying that such working practices does not allow for individual thinking process – which requires time to consciously and subconsciously work and think about a problem. Skowronski does the comparison with great thinkers of our time (Isaac Newton and more) that much of their best work came out of working and thinking in solitude and also pointing out that difficult problems may not be possible to solve through brainstorming sessions in groups but may need experience and knowledge from outside the appointed

project group. The agile proponents on the other hand argues that people working together under conditions of good and effective communication and interaction can operate at higher levels and be more effective than they had had they only been using their talents individually, and that this is proven over and over in brainstorming and problem-solving sessions.

Risks with Direct Feedback

Skowronski (2004) also speak of the risk with direct feedback - something agile methods encourage between projects members, developers and customers; that it needs to be carefully controlled, and even when it is controlled it may still pose a risk for killing new ideas prematurely if not supported correctly.

Developers "Should Not Be 'People-People'"

Skowronski comments on the 'people-centred' aspects of agile methods saying that software engineering people primarly solve technical problems and not problems related to people, and because of that focus on the technical aspects is much more efficient. Also saying that too much interaction with other people may hamper or interfere with these people's (the developers) focus on 'things' and technicalities, thus arguing against the fundamental values of agile.
The agile values states that software development is far from a solely technical activity, and also does Cockburn and Highsmith conclude that agile individuals have a difficult time to function well in a rigid organization with traditional methodologies, and vice versa. The agile approach tends to grow and span teams, organisations and other form of working relationships as well.

Costumer Involvement

Agile principles and values dictate bringing the customer out of the traditional customer role where he/she/they are only present at the beginning of a project to set up the requirements, and later when acceptance testing and release are planned. The agile approach says that it is not possible to understand each other fully from the beginning, even more so when systems become increasingly complex, and the traditional methodology of meeting up, producing a product, deliver product, and then after that go back to implement change requests; is time-consuming and inefficient. In addition, the requirements are changing as the environment for a company/system is constantly changing and thus requirements will be applied.

The fundamental idea behind is to use work practices that welcome change and deals with it effectively and flexibly, and this, it is argued, requires a customer to be present and preferably involved in the work as a team member throughout the project. Additionally, it should not be just any customer representative but a customer present who is; committed, knowledge able, collaborative, representative, empowered, and know and understand what is required for the end user. If the customer has no power to make decisions on features in the product development it may still cause delays and hamper agility for the project. However, that type of customer may not be available or even willing to take the 'agile' role of involvement in a software Project; hence the agile approach may not be appropriate or desirable for the project at hand.

"If customers do not collaborate, then you will not see the full advantages of an adaptive process. Having said that we have found on several occasions that we have worked with customers who did not want to collaborate, but changed their mind over the first few months as they begun to understand the agile approach". **Martin Fowler**

1.5.4 Risk Management

Risk in itself is not bad; risk is essential to progress, and failure is often a key part of learning. But we must learn to balance the possible negative consequences of risk against the potential benefits of its associated opportunity. **(Van Scoy, 1992)**

A risk is a potential future harm that may arise from some present action (Wikipedia, 2004), such as, a schedule slip or a cost overrun. The loss is often considered in terms of credibility, future business, and loss of property or life.

Risk management is a series of steps whose objectives are to identify, address, and eliminate software risk items before they become either threats to successful software operation or a major source of expensive rework. (Boehm, 1989) The software industry is fraught with failed and delayed projects, most of which far exceed their original budget. The Standish Group reported that only 28 percent of software projects are completed on time and on budget. Over 23 percent of software projects are cancelled before they ever get completed, and 49 percent of projects cost 145 percent of their original estimates. (Standish, 1995)

In hindsight, many of these companies indicated and their problems could have been avoided or strongly reduced if there had been an explicit early warning of the high-risk elements of the project. Many projects fail either because simple problems were reported too late or because the wrong problem was addressed. (Bruegge and Dutoit, 2000) Problems happen. Teams can choose to be reactive or proactive about these problems.

Reactive teams fly into action to correct the problem rapidly in a crisis-driven, firefighting mode. Without proper planning, problems often occur late in the schedule. At this point, resolving any serious problems can require extensive modification, leading to big delays.

Proactive teams begin thinking about risks even before technical work is initiated. Their objective is to be able to avoid risk whenever possible, to solve problems before they manifest themselves and to respond to problems that do happen in a controlled and effective manner.

The Risk Management Practice

The risk management process can be broken down into two interrelated phases, risk assessment and risk control, as outlined in Figure 1.2. These phases are further broken down. Risk assessment involves risk identification, risk analysis, and risk prioritization.

Risk control involves risk planning, risk mitigation, and risk monitoring. (Boehm, 1989)

It is essential that risk management be done iteratively, throughout the project, as a part of the team's project management routine.

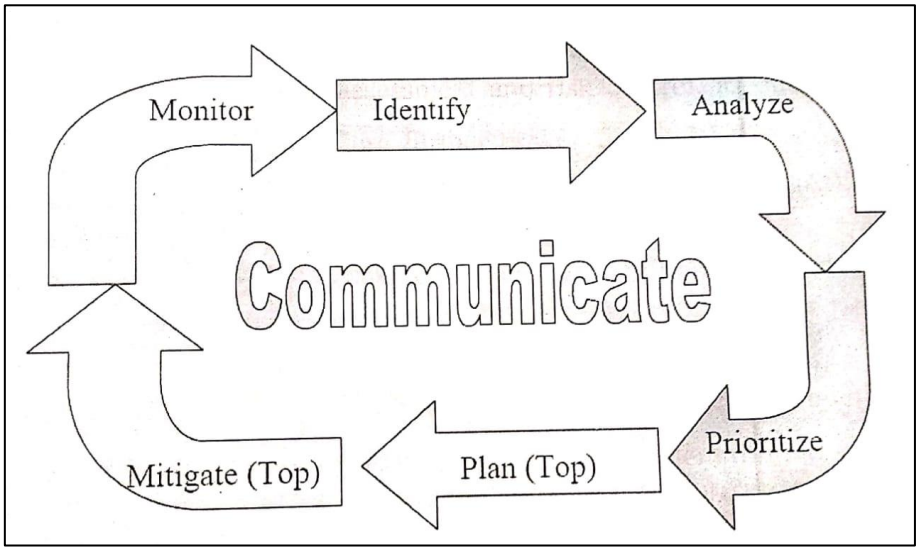


FIG. 1.2: THE RISK MANAGEMENT CYCLE

1. Risk Identification

In the risk identification step, the team systematically enumerates as many project risks as possible to make them explicit before they Become problems. There are several ways to look at the kinds of software project risks, as shown in Table 1.1. It is helpful to understand the different types of risks so that a team can explore the possibilities of each of them. Each of these types risk is described below.

TABLE 1.1: GENERAL CATEGORIES OF RISK

Generic Risks		Product-Specific Risks
Project Risks	Product Risks	Business Risks
Factors to consider: People, size, process, technology, tools, organizational, managerial, customer, estimation, sales, support		

Generic risks are potential threats to every software project. Some examples of generic risks are changing requirements, losing key personnel, or bankruptcy of the software company or of the customer. It is advisable for a development organization to keep a checklist of these types of risks. Teams can then assess the extent to which these risks are a factor for their project based upon the known set of programmers, managers, customers, and policies.

Product- specific risks can be distinguished from generic risks because they can only be identified by those with a clear understanding of the technology, the people, and the environment of the specific product. An example of a product-specific risk is the availability of a complex network necessary for testing.

Generic and product-specific risks can be further divided into project, product, and business risks. Project risks are those that affect the project schedule or the resources (personnel or budgets) dedicated to the project. Product risks are those that affect the quality or performance of the software being developed. Finally, business risks are those that threaten the viability of the software, such as building

an excellent product no one wants or building a product that no longer fits into the overall business strategy of the company

There are some specific factors to consider when examining project, product, and business risks. Some examples of these factors are listed here, although this list meant to stimulate your thinking rather than to be an all-inclusive list.

- People risks are associated with the availability, skill level, and retention of the people on the development team.
- Size risks are associated with the magnitude of the product and the product team. Larger products are generally more complex with more interactions. Larger teams are harder to coordinate.
- Process risks are related to whether the team uses a defined, appropriate software development process and to whether the team members actually follow the process.
- Technology risks are derived from the software or hardware technologies that are being used as part of the system being developed. Using new or emerging or complex technology increases the overall risk.
- Tools risks, similar to technology risks, relate to the use, availability, and reliability of support software used by the development team, such as development environments and other Computer-Aided Software Engineering (CASE) tools.
- Organizational and managerial risks are derived from the environment where the software is being developed. Some examples are the financial stability of the company and threats of company reorganization and the potential of the resultant loss of support by management due to a change in focus or a change in people.

- Customer risks are derived from changes to the customer requirements, customer' lack of understanding of the impact of these changes, the process of managing these requirements changes, and the ability of the customer to communicate effectively with the team and to accurately convey the attributes of the desired product.
- Estimation risks are derived from inaccuracies in estimating the resources and the time required to build the product properly.
- Sales and support risks involve the chances that the team builds a product that the sales force does not understand how to sell or that is difficult to correct, adapt, or enhance.

Spontaneous and sporadic risk identification is usually not sufficient There are various risk elicitation techniques the team can use to systematically and proactively surface risks:

- Meeting. The team, including the development team and the marketing and customer representatives if possible, gathers together. The group brainstorms; each participant spontaneously contributes as many risks as they can possibly think of.
- Checklists/ Taxonomy. The risk elicitors are aided in their risk identification by the use of checklists and/or taxonomies (in other words, a defined, orderly classification of potential risks) that focuses on some subset of known and predictable risks.

Checklists and taxonomies based upon past projects are especially beneficial. These artifacts should be used to interview project participants, such as the client, the developers, and the manager.

- Comparison with the projects. The risk elicitors examine the risk management artifacts of previous projects. They consider whether these same risks are present in the new project.
- Decomposition. Large, unwieldy, unmanageable risks that are identified are further broken down into small risks that are more likely to be managed. Additionally, by decomposing the development process into small pieces, you may be able to identify other potential problems.

Project participants can be reluctant to communicate potential failures or shortcomings and can be too optimistic about the future. It is essential that all participants are encouraged to report risks so they can be monitored and managed. Participants should be rewarded for identifying risks and problems as early as possible.

It is recommended that risks should be stated using the condition-transition-consequence (CTC) format (Gluch, 1994):

Given that <condition> then there is a concern that (possibly) <transition> <consequence>.

- Condition is a description of the current conditions prompting concern.
- Transition is the part that involves change (time).
- Consequence is a description of the potential outcome.
- For example, given that no one in our team has ever developed a product in Prolog, then there is a concern that (possibly) the project will take two months longer than has been estimated.

II. Analyze

After risks have been identified and enumerated, the next step is risk analysis. Through risk analysis, we transform the risks that were identified into decision-making information. In turn, each risk is considered and a judgment made about the probability and the seriousness of the risk. For each risk, the team must do the following:

- Assess the probability of loss occurring. Some risks are very likely to occur. Others are very unlikely. Establish and utilize a scale that reflects the perceived likelihood of a risk. Depending upon the degree of detail desired and/or possible, the scale can be numeric, based on percentage scale, such as "10 percent likely to lose a key team member" or based on categories, such as: very improbable, improbable, probable, or frequent. In the case that a categorical assignment is used, the team should establish a set numerical probability for each qualitative value (e.g. very improbable= 10 percent, improbable= 25 percent).

    - Assess the impact of the loss if the loss were to occur, Delineate the consequences of the risk, and estimate the impact of the risk on the project and the product. Similar to the probability discussion above, the team can choose to assign numerical monetary values to the magnitude of loss, such as $10,000 for a two-week delay in schedule.

Alternately, categories may be used and assigned values, such as 1=negligible, 2-marginal, 3=critical, or 4=catastrophic.

Determining the probability and the magnitude of the risk can be difficult and can seem to be arbitrarily chosen. One means of determining the risk probability is for each team member to estimate each of these values individually. Then, the input of individual team members is collected in a round robin fashion and reported to the group. Sometimes the collection and reporting is done anonymously. Team members debate the logic behind the submitted estimates. The

Individuals then re-estimate and iterate on the estimate until assessment of risk probability and impact begins to coverage. This means of converging on the probability and estimate is called the Delphi Technique (Gupta and Clarke, 1996). The Delphi Technique is a group consensus method that is often used when the factors under consideration are subjective.

The analyzed risks are organized into a risk table. The template for a risk table is shown in Table 1.2.

TABLE 1.2 RISK TABLE TEMPLATE

Rank	Risk	Probability	Impact	Rank Last Week/ Weeks on list	Action

III. Prioritize

After the risks have been organized into a risk table, the team prioritzes the risks by ranking them. It is too costly and perhaps even unnecessary to take action on every identified risk. Some of them have a very low impact or a very low probability of occurring – or both. Through the prioritization process, the team determines which risks it will take action on.

The team sorts the list so that the high probability, high impact risks percolate to the top of the table and the low-probability, low impact risks drop to the bottom. If the team used categorical values for probability (e.g. very improbable, improbable, probable, or frequent) and/or impact (e.g. negligible, marginal, critical, or catastrophic), group consensus techniques may need to be used to produce the risk ranking.

If numerical values were given for probability (percentage) and impact (monetary), the risk exposure can be calculated. Risk exposure is calculated as follows (Boehm, 1989):

Risk Exposure (RE) = P x C

Where P = probability of occurrence for a risk and C is impact of the loss to the product should the risk occur. For example, if the probability of a risk is 10 percent and the impact of the risk is $10,000 the risk exposure = (0.1)($10,000) = $1,000. If RE is calculated for each risk, the prioritization is based upon a numerical ranking of the risk exposures.

After he risks are prioritized, the team, led by the project manager, defines a cut off line so that only the risks above the line are given further attention. The activities of this "further attention" are to plan, mitigate, monitor, and communicate. The lower ranked risks stay on the table for the time being with no action other than monitoring.

IV. Plan

Risk management plans should be developed for each of the "above the line" prioritized risks so that proactive action can take place. These actions are documented in the Action column of the Risk Table (Table 1.2). Following are some examples of the kinds of risk planning actions that can take place:

- Information buying. Perceived risk can be reduced by obtaining more information through investigation. For example, in a project in which the use of a new technology has created risk, the team can invest some money to learn about the technology.

Throw-away prototypes can be developed using the new technology to educate some of the new technology and to assess the fit of the new technology for the product.

- Contingency plans. A contingency plan is a plan that described what to do if certain risks materialize. By planning ahead with such a plan, you are prepared and have a strategy in place do deal with the issue.
- Risk reduction. For example. If the team in concerned that the use of a new programming language may cause a schedule delay, the budget might contain a line item entitled "potential schedule" to cover a potential schedule slip, the financial risk to the organization is reduced. Alternately, the team can plan to employ inspections to reduce the risk of quality problems.

- Risk acceptance. Sometimes the organization consciously choose to live with the consequences of the risk (Hall, 1998) and the results of the potential loss. In this case, no action is planned.

V. Mitigate

Related to risk planning, through risk mitigation, the team develops strategies to reduce the possibility or the loss impact of a risk. Risk mitigation produces a situation in which the risk items are eliminated or otherwise resolved. These actions are documented in the Action column of the Risk Table (Table 1.2). Some examples of risk mitigation strategies are as follows:

- Risk avoidance. When a lose-lose strategy is likely (Hall, 1998), the team can opt to eliminate the risk. An example of risk avoidance strategy is the team opting not to develop a product or a particularly risky feature.
- Risk protection. The organization can buy insurance to cover any financial loss should the risk become reality. Alternately, a team can employ fault-tolerance strategies, such as parallel processors, to provide reliability insurance.

Risk planning and risk mitigation actions often come with an associated cost. The team must do a cost/benefit analysis to decide whether the benefits accrued by the risk management steps outweigh the costs associated with implementing them. This calculation can involve the calculation of risk leverage (Pfleeger, 1998).

Risk Leverage = (risk exposure before reduction – risk exposure after reduction)/cost of risk reduction

If risk leverage value, rl, is <1, clearly the benefit of applying risk reduction is not worth its cost. If rl is only slightly > 1, still the benefit is very questionable, because these computations are based on probabilistic estimates and not on actual data. Therefore, rl is usually multiplied by a risk discount factor p <1. If p rl >1, then the benefit of applying risk reduction is considered worth its cost. If the discounted leveraged valued is not high enough to justify the action, the team should look for other, less costly or more effective, reduction techniques.

VI. Monitor

After risks are identified, analyzed, and prioritized and actions are established, it is essential that the team regularly monitor the progress of the product and the resolution of the risk items, taking corrective action when necessary. This monitoring can be done as part of the team project management activities or via explicit risk management activities.

Often Teams Regularly Monitor Their "Top 10risks,"

Risks need to be revisited intervals for the team to reevaluate each risk to determine when new circumstances caused its probability and/ or impact to change. At each interval, some risks may be added to the list and others taken away. Risks need to be

Reprioritized to see which are moved "above the line" and need to have action plans and which move "below the line" and no longer need action plans. A key to successful risk management is that proactive actions are owned by individuals and are monitored. (Larman, 2004)

As time passes and more is learned about the project, the information gained over time may alter the risk profile considerably. Additionally, time may make it possible to refine the risk into a set of more detailed risks. These refined risks may be easier to mitigate, monitor, and manage.

VII. Communicate

On-going and effective communication between management, the development team, marketing, and customer representatives about project risks is essential for effective risk management. This communication enables the sharing of all information and is the cornerstone of effective risk management.

The Stakeholder of Risk Management

The three stakeholders are involved in risk management.

- The developer must systematically and continually enumerate all the possible risks related to technical capability and making the schedule.
- The manager must lead the team to follow the risk management process to proactively manage the project risks. The manager must also allocate resources for proactive risks management.

- The customer must participate in the continual identification of risks.

None of these stakeholders is empowered to manage business risks, i.e. what we called organization and managerial risks, and sales and support risks in the "Risk Identification" section above. The kind of risk must be managed by upper management and marketing department of the firm.

International Journal of Advanced Multidisciplinary Scientific Research (IJAMSR) ISSN: 2581-4281
www.ijamsr.com CrossRef: https://doi.org/10.31426/ijamsr

MANAGING RISKS: CHALLENGES IN ADOPTION OF AGILE SOFTWARE DEVELOPMENT APPROACH

CHAPTER - 2

REVIEW OF LITERATURE

The experience reports from the use of agile methods, often written by practitioners or consultants, have been predominantly positive. However, academic research on the subject is still scarce (Abrahamsson et al., 2002). Therefore there is little scientific proof that would support many of the claims made by the agile community (Mcbreen 2003, cited in Dyba & Dingsoyr, 2008).

Laanti et al. (2011) suggest, that the lack of research on the subject stems from companies rarely having comparable data to explain the impacts of agile methods before and after their adoption. Also, the majority of existing research consists of qualitative case studies of individual projects, and therefore the results are not very applicable to wider contexts, and they also lack a broader scope (Laanti et al., 2011).

Until 2008, Dyba and Dingsoyr had identified altogether 1996 studies on the subject of agile software development. Only 3 of these studies were empirical research that they measured to be acceptable enough in terms of relevance, rigour and credibility. It seems as one of the research gaps in agile software development research is also, that majority of empirical studies of sufficient quality focus on evaluating a single process model, in most cases Xp.

According to Dyba & Dingsoyr's (2008) review, (76%) of the studies were done in organizations using XP as their agile methodology. Studies on agility in general seem to be the second popular theme in literature with 15% of the studies, at least in academic context (Dyba & Dingsoyr, 2008). Despite the popularity of Scrum, the authors found only one case study prior to 2006 researching that method.

After reviewing the existing literature on agile software development, Dyba and Dingsoyr (2008) consider that the state of research on agile software development being currently on a nascent state, which suggests a need for exploratory qualitative studies.

A common argument is that agile methodologies are not suitable for large organizational settings and in large teams (Cohen et al., 2004, cited in Dyba & Dingsoyr, 2008). However, some studies have suggested that agile can be very successful in large organizational setting as well, as long as the organizational environment supports agile adoption.

Lindvall et al. (2004) claim that failure to adopt agile successfully in large organizations is related to their complexity and rigid organizational structure. Laanti et al. (2011) note that agile adopters are not often aware of what agility actually means, and how broad of a change is actually required. Also, in large organizational settings where complex software is produced, a holistic view on agility may be needed; it is not enough to focus on team and project-level dimensions only, as in a typical application of agile software methods (Kettunen & Laanti, 2008).

Lindvall et al. (2004) note that the challenge for large organizations doesn't necessarily lie in applying agile practices into a project, but in integrating the agile project into it environment. There is a possibility for conflict and double work, when agile practices interact with traditional ones (Lindvall et al., 2004). Agile does not usually thrive in large teams, mostly because it makes face-to-face communication challenging (Lindvall et al., 2004).

The main conclusion of Svensson Host's (2005, cited in Dyba & Dingsoyr, 2008) study was that the introduction process of agile development was difficult due to the complexity of the organization. The authors studied a large software development company that introduced the XP development process to a pilot team, during a period of eight months. As a result, they advise companies that are introducing the XP process into the organization to set clear goals on what to introduce, and communicate this clearly with the rest of the company. Also, the companies should bear in mind, that the adoption process should not be underestimated, because it takes time and effort (Svensson & Host, 2005, cited in Dyba & Dingsoyr, 2008).

Lindvall et al. (2004) studied the experiences of agile software development methods in large organizations. The authors conclude, that based on the experiences in large organizations, agile methods can be successful and especially in small, collocated teams (Lindvall et al., 2004). The authors based their analysis on the experiences shared by the Software Experience Center (SEC) member companies, including ABB, Daimler- Chrysler, Motorola, and Nokia. The representatives of Nokia noted that small software development teams are more productive than large ones (Lindvall et al., 2004). The notion of a small size team is also present I agile philosophy. For example in Scrum, the maximum suggested team

size is 9 individuals. This is also in order to ensure effective face-to-face communication, which is also part of agile philosophy, defined in the Agile Manifesto.

According to Boehm (2002), the scalability of the agile process seems to be one of the limiting factors in the use of agile methods. As the team size gets bigger, communication through documents becomes easier than explaining everything, most probably various times, to a larger number of people by face-to-face communication.

Bahli and zeid (2005) studied a Canadian Organization shifting from a Waterfall process to XP, and got quite positive results in their study. The developers found XP easy to use already after one week of training. However, a development manager stated that the adoption process itself wasn't easy, since none of the developers had prior experience on XP and it will take more work to master it completely. The developers in this study also stated that they prefer XP over Waterfall, calling the latter and unpleasant experience", while XP was described to be "beneficial and a good move from the Management". However, the size of the organization is unknown, since the company wanted to remain anonymous (Bahli & Zeid, 2005).

A study by Robinson and Sharp (2004, cited in Dyba & Dingsoyr, 2008) found that XP has the ability to thrive in very different kinds of organizational settings. The study consisted of three case companies, and factors of organizational type, size and structure as well as physical and temporal settings varied significantly between these companies. The authors found out that despite the diversities in these factors, XP was working well (Robinson & Sharp, 2004, cited in Dyba & Dingsoyr, 2008).

Dyba and Dingsoyr (2008) conclude, that according to research they analyzed in their review, XP seems to thrive in very different organizational settings, for example in organizations that varied from having a hierarchical structure to those that there was little or no central control. Also customer involvement and physical settings varied between the successful XP teams studied. In terms of the adoption process, XP is found to difficult to introduce in a complex organization, but seemingly easy in other types of organizations. Also, many suitable for small teams than in larger ones (Dyba & Dingsoyr, 2008).

Laanti et al. (2011) conducted a quantitative study on developer perceptions about agile methodologies at Nokia. The data was collected using a questionnaire, and more than a 1000 respondents from three different continents took part in the survey. Nine statements on agile development were presented (see Table 2, below), and a scale from 1 to 7 was used to collect the responses: (1 = totally disagree, 4 = neutral and 7 = totally aggress).

According to Laanti et al.'s (2011) research, responses to only one of the nine statements were below neutral (making work less hectic), while all the other statements received positive responses on average. The results revealed that most of the respondents agreed on all accounts with the generally claimed benefits of agile, which include higher satisfaction, feeling of effectiveness, increased quality and transparency, increased autonomy and happiness and earlier detection of defects (Laanti et al., 2011).

Laanti et al. (2011) also addressed the question of how much a person's background, i.e. the length of experience on agile and traditional methods, affects the perceptions on agile methodologies and overall satisfaction of them. The second motivation for the study was to figure out, how useful of a software development method agile is from the practitioner's point of view.

Laanti et al. (2011) identified respondents falling into groups for and against agility. 65% of respondents would like to stay in the agile development mode, as 6% would like to go back to a traditional way of working instead. The rest of the respondents (27%) were neutral, mostly because some of the population answering the survey did not yet have experience on working in agile mode.

Laanti (2012) also studied the wellbeing in agile tams applying mainly Scrum as their development methodology. The research was conducted at Nokia, where 466 employees responded to an online survey sent to 10000 employees, which was analyzed by using quantitative methods. According to this study, 55% of respondents were happy to work in an agile way, and only 12% would like to go back to traditional way of working. Further, the study indicates that agile practices might relieve people's stress. 27% of respondents stated that their stress level is better because of agile, while 15% said that they feel worse. Rest of the respondents did not notice a direct link between work mode and stress level, or felt that it has not changed after the adoption of agile methodologies.

Mann and Maurer's (2005) study on introducing Scrum into an organizations detected that developer perceptions on agile development were very positive as well. They perceived the Scrum process being very beneficial, and in a questionnaire, every developer

inside a Scrum team would recommend using Scrum in the future. The closer relationship with the customer's, as well as increased communication with them was seen as a very positive thing and giving confidence on being able to meet the customer's needs better. In addition, the developers were more satisfied with the quality of the product created than in previous projects.

There has been some research on whether a person's background or personality traits have an impact on perceptions about agile. Laanti et al. (2011) found out, that more than three years of experience on traditional methods affected somewhat negatively towards perceiving agile methods, and those people also found more difficult seeing the benefits of it.

Korhonen (2010) also found a positive correlation between the engagement and contentment of agility and being able to perceive improved quality of the products created.

Melnik and Maurer (2006) got similar results finding moderate positive correlation between the level of experience with agile methods and the overall job satisfaction. The study was a comparative survey of job satisfaction. The study was a comparative survey of job satisfaction among 448 IT employees, using both agile and traditional development methods. The data was analyzed using quantitative methods and respondents were recruited via active newsgroups, mailing lists and Wikis specialized in software engineering. The results also indicated that in there are twice as many members working in agile teams who satisfied with their jobs, satisfiers were ability to influence decisions that affect you, the opportunity to work on interesting projects and the relationships with the users (customers).

Young et al. (2005, cited in Dyba & Dingsoyr, 2008) studied the role of personality traits in agile software development teams. They used a technique called "repertory grid analysis" to identify good (and bad) characteristics for members in different roles in an XP development team. They defined the characteristics of a "good" XP team member as "analytical, with good interpersonal skills and a passion for extending his knowledge base and passing this on to others" (Young et al., 2005, cited in Dyba & Dingsoy, 2008). In regards to job satisfaction and wellbeing in agile development teams, studies have yielded mainly positive results.

Mannaro et al. (2004) studied the job satisfaction among employees in software companies, which used XP as an agile methodology to those who didn't apply agile development methods. The research was conducted with a web bases questionnaire of 122 participants. The results were very complementary towards XO and not that much towards non-agile development. 95% of employees using XP were satisfied with the current development methods and wanted to keep using XP. In comparison, the satisfaction rate among employees that were not applying agile methodologies was only 40%.

Mannaro et al. (2004) found out, that the employees who apply XP practices have greater job satisfaction in a way that they feel job environment is more comfortable and that their productivity is higher compared to those using non-agile development process.

Syed-Abdullah et al. (2006) studied, if agile methods have any distinct effects on wellbeing amongst people in agile software development teams. The subjects of research were software engineering students, with a total population of 75 people, forming

altogether 17 teams. Half of the teams developed software with an agile XP methodology, and the other half with a plan-driven methodology. The research methods used in the study were quantitative and qualitative in nature, consisting of participative observation, focus group interviews, close-ended questionnaires and simple statistical test (Syed- Abdullah et al., 2006). Wellbeing was conceptualized in this study by examining factors of job related anxiety, depression, contentment and enthusiasm. These factors were measured by using a 12-items anxiety-contentment and depression-enthusiasm scales developed by Warr (1990, cited in Syed-Abdullah et al., 2006). The results of the study showed statistically no significant difference between agile and plan-driven development teams, except with the level of enthusiasm. The agile methods (XP) had a positive effect on the level of enthusiasm among developers, meaning the feelings of enthusiasm, optimism and cheerfulness towards the project being developed (Syed-Abdullah et al., 2006).

After doing a comprehensive research of existing research on agile development, Dyba and Dingsoyr (2008) conclude that developers are prevalently satisfied with agile methods, and the software developers in companies that use XP as an agile method have reported to be more satisfied with their job and with the products developed as well (Dyba & Dingsoyr, 2008). The authors continue, that whilst the effect on work practices and job satisfaction from the use of agile or traditional methods is not thorough, some studies suggest that the standardized work practices in agile development, lead to greater job satisfaction (Dyba & Dingsyor, 2008). Person's background seems to have an impact on the perceptions and contentment on agile methodologies (Laanti et al., 2011; Melnik & Maurer, 2006). Contentment in agile seems to have an impact on

being able to see the benfits of the methodologies as well (Korhonen 2010; Laanti et al., 2011).

The research on agile teams and productivity is still scarce, and therefore it is difficult to draw conclusions, whether agile increases productivity or not. Dyba and Dingsyor (2008) could identify only four studies in their review, in which productivity of agile teams was measured in quantitative methods. However, these studies had all an inappropriate recruitment strategy, so unbiased comparison is not ensured (Dyba & Dingsoyr 2008).

Ilieva et al. (2004,cited in Dyba & Dingsoyr, 2008) compared the productivity of two similar projects, of which one used traditional software development methods and the other XP. They measured the productivity of three iterations in each project and the results indicated a 42% increase in productivity for the agile team.

A case study by Layman et al. (2004) compared an old product release developed with traditional methods to a new release developed with agile methods. They discovered a 46% increase in productivity for the new agile release compared to the old one. However, in this case, the agile team members had more expertise in software engineering and project management experience than the traditional project team members (Dyba & Dingsoyr, 2008).

There are also number of studies that suggest that the subject of the research themselves believe that agile methods lead to increasing productivity (Dyba & Dingsyor, 2008). According to Melnik and Maurer's (2005, cited in Dyba & Dingsyor 2008) study of student perceptions on agile, 78% of respondents believed strongly that using XP improves productivity in small teams.

This was also the case in Laanti's (2012) research, where 64% out of 466 respondents felt that their performance had increased after taking agile methods into use. The main finding of the study was that developing software at a sustainable pace leads to better performance. Sustainable pace in development is one of the main principles of the agile: Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely. In practice, this means working in a constant flow and focusing on top priorities. This principle is opposing the Waterfall-like development process where working overtime near deadline and resting after it is over, is usual (Laanti, 2012).

Contrary to these findings, a study by Wellington et at. (2005) analyzing XP in the use of students, reported a 44% decrease in productivity compared to a plan-driven team. Furthermore, Svensson and Host (2005, cited in Dyba & Dingsoyr, 2008) found no evidence of change in productivity after agile processes were introduced to a company.

Some studies indicate that some agile practices require skilled individuals, in order for them to increase productivity, Melnik and Maurer (2002, cited in Dyba & Dingsoyr, 2008) studied student perceptions on XP. While most of the students noted that the skill differences between him and his pair were quite significant, which resulted in decreased productivity. Also, test driven development was found to be difficult bt many students. The authors believe that this is because writing the tests before designing, forces the students to make design decisions early (Melnik and Maurer, 2002 cited in Dyba & Dingsoyr, 2008),

A study in Tessem (2003, cited in Dyba % Dingsoyr, 2008) of the XP process, Indicates that it takes time to learn correct estimations. At the beginning of the project, only one third of the user story estimations were correct. Also, several study participants felt that there was not enough discussion on design and architecture throughout the project (tessem 2003, cited in Dyba & Dingsoyr, 2008).

Dyba and Dingsoyr (2008) conclude these studies related to the productivity of agile versus traditional teams, that three out of four show that using XP results in increased productivity in terms of LOC/h (lines of code per hour). Also developer's perceptions on the impacts agile has on productivity and performance seem to be mainly positive.

A common statement made by the advocates of agile development is, that agile processes lead to better quality in many ways. The arguments supporting this are often related to the continuous testing and integration of the code, instead of doing all of that at the end of the project like in plan-driven methodologies (Nerur et al., 2005; Schwaber et al., 2007; Leffingwell, 2011).

A comparative study by Huo et al. (2004) of quality assurance in a Waterfall process compared to and agile process suggests, that the frequency of quality assurance (QA) practices are higher in agile processes. The conclusion could be drawn, therefore, that as code is tested more often, it would lea to better quality as well.

Layman et al. (2004) conducted a longitudinal case study at an airline company inside one software development team, and measured the product quality before and after the XP methodology was adopted. After the adoption process, a 65% improvement in prerelease quality

and a 35% improvement in post-release quality were reported. In a comparison of two similar projects, Ilieva et al. (2004, cited in Dyba & Dingsoyr, 2008) found 13% fewer defects in the product (reported by the customer or by the quality assurance team) in an XP project that in a non-XP project.

Wellington et al. (2005) conducted a case study among university students, majoring in software engineering. They offered two courses, one based on plan-driven methodology and one on agile development, XP in particular. They found out, that compared to a traditional team, the XP team's code scored consistently better on the quality metrics and that the quality of the code delivered was significantly greater than that of a traditional team. However, in the same study, both teams perceived that the plan-driven team had created a better user interface in the product. Macias et al. (2003, cited in Dyba & Dingsoyr, 2008) measured the quality of products developed by 10 plan-driven and 10 XP teams and they found no significant difference in results.

In addition to proving the increased quality in quantitative metrics, according to some studies, developers perceive that quality has improved after the adoption of agile methods.

Laanti et al., (2011) conducted an extensive quantitative survey study across Nokia, with over 1000 respondents and a response rate of 33%. The aim of the study was to get an overview on practitioner's perceptions on agile development. The study reports positive results on many accounts, including increased product quality (Laanti et al., 2011).

Fitzgerald (2000) reported that practitioners did not adopt formalized methodologies in their in their prescribed form and only 6% of surveyed organizations followed a methodology rigorously; practitioners customized system development methodology in a pragmatic way and methodology-in-action was uniquely enacted for each development project. Similarly, Russo (1995) found that 85% of the surveyed organizations adapted the system development methodology on a project by project basis; software professionals view methodology as a general framework of phases or activities and the decisions regarding what development activities to perform is typically made at the project-team level.

Boehm and Turner (2003 and 2004) suggested that developers should use risk assessment to find the "seep spot" for balancing agile and plan-driven methods; an example of this type of method tailoring is the combination of the ISO 9000 with the XP. Another way of method tailoring is to select and combine agile practices suggested by different agile methods. An example is the selection and combination of XP and scrum practices to improve both the development processes and to strengthen the project management at Intel Shannon, in Ireland (Fitzgerald, Hartnett and Conboy, 2006).

Mannaro et al. (2004) conducted a web-based questionnaire of 122 respondents, among software companies that used XP method and in companies that did not apply agile methods. The study reports that 76% of the people, who had applied XP in their work, believed that XP has improved the quality of code (Mannaro et al., 2004).

Korhonen (2010) studied, if people's perceptions impacts of the agile practices havinh on code quality were realistic, and the results were quite surprising. The survey was conducted in a large, globally disturbed telecommunications organizations, migrations from traditional (Waterfall) development mode into agile. As survey was conducted after 6 months of agile adoption, and it received 78 responses. Only 25% of agile adoption, and it received 78 responses. Only 25% of agile adopters believed that quality had increased, in reality, the quality had improved and was visible in defect data. Further, the study revealed that a realistic perception of the positive changes in the defect data coincided with positive emotional engagement in agile transformation.

Robinson and Sharp's (2004) study seems strengthens the proof for the previous statements. The authors drew a conclusion from their study of three case organizations using XP, that agile development teams have faith in their own abilities, show respect and responsibility, establish trust and preserve the quality of working life. Further, Laanti's (2012) quantitative research study at Nokia indicated that the majority (71%) of respondents working in agile teams feel their team is empowered.

Whitworth & Biddle's (2007) study of the motivation and cohesion in agile teams brought up positive results as well. The research was conducted by analyzing qualitative data, based on semi-structured interviews with 22 participants working in different agile teams. All but two had experience in working in non-agile teams before agile. The participants were recruited through networking with members of the agile software development community. They operated in variety of

roles, including developers, designers, 44 project managers, coaches and specialists. The researchers examined the results by trying to point out characteristics that are related to team cohesion. They found out, that the main value of agile methods in supporting team cohesion and motivation was a collective team culture. As a result of the agile planning in the beginning of a sprint, as well as the iterative nature of agile software development, the process was seen as stabile but at the same time complex, creative and social problem-solving activity (Whitworth & Biddle 2007).

This kind of agile development process was seen to support and even require the development of collective culture and team-wide communication, including effective feedback mechanisms in order for it to work properly (Whitworth & Biddle 2007). The results also showed that self-efficacy was experienced highly by agile team members. Self-efficacy means things such as effort or skill put into the work and controllability or modifiability of one's environment (Eby & Dobbins 1997, cited in Whitworth & Biddle 2007). Agile practices, such as daily meetings, feedback, negotiation of a flexible plan, continuous integration and testing were seen to increase perceptions of self-efficacy and control within the team (Whitworth & Biddle 2007). Being involved and aware of the project activities supported the feelings of self-efficacy, whilst the team members that were not clued to day-to-day activities experienced discomfort, dissatisfaction and the absence of self-efficacy (Whitworth & Biddle 2007)

Therefore, although a team can be seemingly agile, by failing to involve people in close communication can remove the cohesion of agile teams.

Despite these positive results, Wellington et al., (2005) study of team cohesion in XP and plan-driven teams yielded equal or higher scores for every aspect of cohesion for the non-agile teams. The study was conducted among students of two engineering courses, one teaching XP and the other a plan-driven methodology. Both of the courses formed a team of 14 to 16 students and they were given the same problem statement to solve with a given methods. All in all, this study of team cohesion did not find any improvement of cohesion in the XP team (Wellington et al., 2005).

From their case study of three large companies, Karlstrom and Runeson (2005, cited in Dyba & Dingsoyr, 2008) found out that XP teams experienced improved communication within the team, but that they were perceived by other teams as more isolated. Bahli and Zeid (2005) examined knowledge sharing in XP team and a traditional team, and found out that the creation of tacit knowledge was improved as a result of frequent contacts and communication.

Lindvall et al., (2004) suggest that adding support for cross team communication presents an important need for improvement in large organizations, particularly prominent at Nokia. Large organizations often distribute teams across several physical locations, which can bring up challenges in close and effective communication, which is a crucial part of a successful agile team (Lindvall et al,. 2004).

Studies of agile team dynamics, characteristics and communication indicate that the successful agile teams are able to balance a high level of individual autonomy with a high level of team autonomy and corporate responsibility. Team members in successful agile teams have faith in their abilities and preserve the quality of their working lives.

Also good interpersonal skills and trust were found to be crucial factors for a well functioning agile team (Dyba & Dingsoyr, 2008). An agile teams usually experiences improved communication, but might be isolated from other development teams.

Ceschi et al. (2005) conducted a survey study for 20 project managers in "software companies that were using plan-driven and agile methods. They found out, that agile methods improved the management of the development process as well as relationships with the customer. The results indicate that agility creates increased customer contact, which makes a high quality link between the development team and the customer. Agile companies organize their work in more releases and pay more attention to activity planning by prioritizing essential work in each iteration. Managers in agile companies are more satisfied with the way their projects are organized than those in plan-driven companies (Ceschi et al., 2005).

Mann and Maurer (2005) did a case study, where they assessed the variables of customer satisfaction and working overtime before and after Scrum was introduced into a development team. The quantitative results of measuring working overtime indicated that after the adoption of Scrum, working overtime had decreased significantly. This means that the team was able to work fewer hours and most probably at a sustainable pace. At the same time, customer satisfaction had increased as well. All of the three customers said they would recommend using Scrum in the future, and were happy to be part of the development process. The customers liked the fact that they were

involved in Scrum meetings and found them to be beneficial. They also felt that their respect towards the developers raised as well. Before Scrum, they did not have an active role throughout the process, and sometimes were not satisfied with what was produced.

Dagnino et al. (2004, cited in Dyba & Dingsoyr, 2008) compared the use of an agile approach to a traditional one in two different development projects. They noticed a higher customer satisfaction with the agile team, because it was able to demonstrate business value more quickly and more often than a traditional team. Also, the customer was giving feedback throughout the agile development process, and the agile team was able to incorporate changes more easily because of the incremental development in short cycles.

Sillitti et al. (2005) also found similar results regarding customer collaboration and relationships in agile companies. They interviewed eight project managers from document driven companies and eight working in agile ones. Sillitti et al. (2005) draw conclusions of the results, that agile companies are more customer-centric and flexible than document- driven ones. Agile companies also seem to have a better relationship with the customer than document-driven companies (Sillitti et al. 2005).

However, a case study by Martin et al. (2004, cited in Dyba & Dingsoyr, 2008) of three XP projects with on-site customers indicated that in all three cases the customers were under stress. They had to commit working long hours on the project even though they were supported by various technical advisors and other personnel inside the company.

While there is Extensive literature on risk management, research in relation to risk management in agile SD projects is non-existent. This is surprising considering how quickly agile methods are being adopted in SD. Many books on agile methods "have remarkably little to say about how a development team determines the risks it faces, prioritises them or takes action to negate their effects" (Smith and Pichler 2005).

Essentially, agile methods must "tailor conventional risk management approaches meant for years-long projects into a risk driven agile iteration lasting only seven to thirty days" (Smith and Pichler 2005). How agile projects go about doing this remains unknown.

Agile models claim to be risk-driven (Beck, 2004) (EPF, 2007) (Scrum, 2003). They state that their iterative approach enables continuous attention to risks and that risks can be reduced by practices such as continuous software integration and early testing (Beck, 2004). In reality, however the agile development models implement few risk management practices (Armenta and Gaono, 2008) (Bohner and Coram, 2005) (Sliger, 2006). Hence, there is clearly a gap well worth investigating bearing in mind the fact that risk management is considered best practice in contemporary software engineering.

International Journal of Advanced Multidisciplinary Scientific Research (IJAMSR) ISSN: 2581-4281
www.ijamsr.com CrossRef: https://doi.org/10.31426/ijamsr

MANAGING RISKS: CHALLENGES IN ADOPTION OF AGILE SOFTWARE DEVELOPMENT APPROACH

CHAPTER - 3

RESEARCH METHODOLOGY

The purpose of this chapter is to identify an appropriate research methodology for our research objectives. Before going to the selection of research methodology for individual research objectives, we first provide a brief description of available research methods for research projects in a software engineering setting (as presented by Wohlin, and Creswell).

3.1 Research Methods

Most commonly used research methods in software engineering are Literature review, survey, case study and experiment. Each research method is used far different purposes according to researchers needs.

Literature Study:

The literature review is a research study which fulfills several purposes. It helps in identifying others studies that are closely related to study being reported, identify the gaps in present research work, and in filling gaps and extending prior studies. It provides the framework to search and identify the research study related to particular topic. It also provides the benchmark for comparing the results of the study with other studies.

Survey

Surveys are typically used to perform backward- and/or a forward-looking investigation. Surveys help in obtaining general findings for a population by obtaining answered questionnaires from a representative and unbiased sample. The data from surveys are gathered from either interview and/or through questionnaires. Online questionnaire is considered as the most efficient way of gathering large data sets in short time, on the other hand interviews consume time and money. Moreover, online questionnaires are able to reach more potential respondents, given that the questionnaires can be distributed through many channels (e.g. e-mail, forums, and communities). For example, Gorschek et al., they received more than 3000 answers for their questionnaire.

Case Studies:

Case studies are typically used to perform in-depth investigations of a phenomena concentrating on a particular area, often conducted in an industrial setting. The results of case studies are harder to generalize, often context dependent and also can be used for comparisons. The main advantage of case studies is that they provide very rich and deep information. However, it is difficult to generalize the findings to other settings and situations.

Experiments:

Experiments are mainly carried when the investigations state variables are to be changed to make controlled study. The main difference between an experiment and a case study is the definition of state variables, i.e., the state variables in a case study cannot be changed where as it is possible to manipulate the state variables in an

experiment, which makes the experiment more suitable to investigate different set of state variables. That is, we are able to control variables given that experiments take place in a laboratory setting. This makes it possible to reduce the threat that other unknown variables affect the outcome variables, other than the treatment variables.

3.2 Research Method Selection

After performing a careful review of all the available research methods, we have selected the following research methods for our research objectives.

For research objective 1, we have selected **Literature review.** Literature reviews are particularly useful to identify relevant information with respect to a research question, avoiding to reinvent the wheel and to be able to further built upon previous work. Given that there are many literature reviews on agile practices available, we conduct a tertiary literature review, i.e. a review of literature reviews. With, that we identify the relevant studies that investigated agile practices, and with that also have access to all the primary studies on the topic.

The integrated model (as mentioned in research objective 2), has been developed according to the empirical research process following the inductive reasoning method. As depicted in Figure 3.1, this process consists of the following four phases, *Observation, Pattern, Tentative hypothesis* and *Theory* (Trochim, 2006).

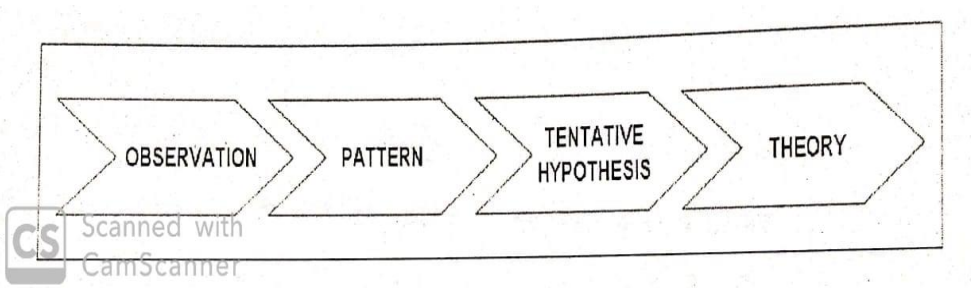


FIG. 3.1: EMPIRICAL

Research Phases

In the *Observation* phase the emphasis is on the collection of empirical facts. When following the inductive approach this means that one examines specific data, perhaps many pieces of specific data, as a first step towards the creation of a general principle or theory within the domain of study. In the next phase, *Pattern,* one identifies patterns in the collected data to be further explored. It involves identifying regularities or relationships between the facts observed in the *Observation* phase. The *Tentative Hypothesis* phase involves further exploration of the identified patterns by studying and verifying them in new contexts. The aim is to specify a tentative hypothesis that allows one to examine the premises for formulating a theory or drawing general conclusions based on the findings made so far. In the final phase, called *Theory,* one collects new empirical data and examines whether the conclusions and theories as predicted by the tentative hypothesis can be supported in the new data. One then interprets the results to generate a theory or the statements to be generalized, or identifies ideas for new hypotheses or research if needed. Another round of the empirical process may therefore start again to further develop and improve the results of this round.

In addition to using an empirical research process based on inductive reasoning, this study uses an explorative research approach. When preparing a research project and deciding on the research approach, one must also determine the epistemological status of the area of research (Karlstrom, 2003).

If there is a low level of previous knowledge, an exploratory study is probably the best selection. If a little more is known and a more detailed result is required, a descriptive methodology might be the better selection. If much is known and relationships are to be confirmed, then an explanatory methodology is probably the best selection.

What Karlstrom (2003) describes here is a research method staircase. As the status of knowledge in a research area increases, one steps up the staircase. Low levels of knowledge imply that the research community must strive to explore and describe the fundamentals of the area of study before examining the exact relationships between elements of knowledge within it.

Considering the current status of knowledge about the integration of risk management and agile models, suggests that our research be initiated at the lower levels of the staircase.

From the point of view of the empirical research process, explorative research means that variables and relationships among variables describing the problem cannot be determined beforehand. Hence, the research question put forward in this thesis is not aimed at testing a hypothesis as prescribed by the explanatory research approaches using deductive reasoning (Trochim, 2006).

On the other hand, on the basis of the results of exploratory or descriptive research, a hypothesis could be specified afterwards which can then be tested in another round of research. Descriptive research stresses the description of an existing variable whereby the relationships between variables are subordinate, whereas in explorative research it is even more relaxed since the researcher has only suppositions about the nature of the problem and therefore starts with identifying the variables. Both explorative and descriptive research therefore addresses understanding the variables, before they and their relationships can be explained (Karlstrom, 2003).

In other words, exploratory research can be considered a first step in identifying and validating fundamental concepts. Hence, when initiating new rounds of the empirical process, it may use either the inductive or deductive approach depending on the status of the results from the previous roun

3.3 Research Process

Our research process, its phases and their inherent steps are illustrated in Figure 3.2.

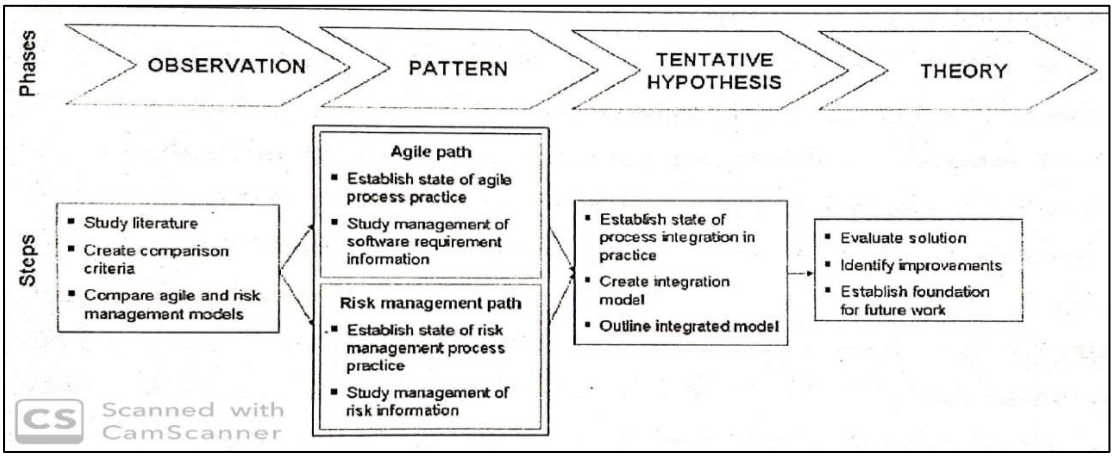


FIG. 3.2: RESEARCH PROCESS

3.3.1 Observation phase

According to our empirical-inductive research approach, the *Observation* phase involves the collection and examination of specific empirical facts within the domain of study. Hence, we conducted a literature survey of the state of art and practice within risk management and agile development. The goal was two-fold: (a) to understand the fundamentals of each discipline, and (b) to investigate the agile process from a risk management perspective.

As listed in the box of the *Observation* phase in Figure 3.2, it consisted of three steps: (1) *Study literature*, (2) *Create comparison criteria*, and (3) *Compare agile and risk management models.*

In the first step, we studied both disciplines. To achieve both breadth and depth of the risk management discipline, we chose publications of renowned industrial and academic institutions, including: (1) international or organizational standards, such as the AS/NZS 4360 (Standards Australia, 2004), IEEE 1540 (IEEE 1540, 2001) and the Project Management Body of Knowledge (PMI, 2004), (2) academic and/or industrial models, for instance proposed by Boehm (1991), Carr et al. (199,3) and Charette (1989), and (3) various investigations made by individual practitioners, researchers or research groups, for instance by Boehm and Turner (2005), Demarco (2004), Englund (1997), Hulett (2001), Kontio (1999), Ropponen and Lyytinen (2000), Westfall (2001) and Wiegers (1998). We did the same with the agile discipline, where we studied well-known agile models, such as eXtreme Programming (Beck, 2004), Lean Software Development (Poppendieck, 2003), OpenUP (EPF, 2007) and Scrum (Schwaber and Beedle, 2001) and their state of practice in depth.

This literature survey helped us identify and establish fundamental aspects of the two disciplines. We used the risk management aspects as criteria for comparing the agile and risk management models in the second step, *Create comparison criteria.*

Finally, in the third step, *Compare agile and risk management models,* we analyzed the risk management and agile process models using the fundamental risk management aspects as comparison criteria. This helped us to identify gaps between the two disciplines studied which recognized future research challenges. The specific observations made regarding these gaps laid the basis not only for further exploration of patterns in the next research phase, but also for the entire thesis work.

3.3.2 Pattern phase

The second research phase, *Pattern,* involves identifying regularities in the data collected and observed in the *Observation* phase to be further explore. Hence, we moved from theoretical to industrial studies by examining the problems identified in the literature in industrial settings. The goal was to identify patterns describing a typical manner of conducting agile development and risk management in industry.

As depicted in Figure 3.2, the research in the *Pattern* phase was carried out in two parallel paths, the *Agile path* and *Risk management path.* We studied the agile and risk management process models separately to establish their status in industry. It also included a minor study of the information managed in both processes. The two paths were however merged into one single path in the third research phase. Each path and the patterns established for each path are briefly described below:-

Agile path

The Agile Path consisted of the two sub-steps, (1) Establish state of agile process practice and (2) Study management of software requirement information (see steps listed in the Agile path box in Figure 3.2). The goal was to establish patterns describing the state of agile process practice and the information needed for communicating information about software requirements and their implementation within the development cycle.

In the first step of the agile path, *Establish state of agile process practice,* we investigated the state of the agile process in three Canadian software organizations. We did this by comparing the industrial practice against a model that we synthesized from a set of current agile process models. Our goal was threefold: (1) to identify the state of industrial agile practice, (2) to compare it to the existing agile process models, and (3) to find out how the industry has approached both agile and heavyweight activities.

In this step, we first studied current agile process models. However, we selected XP and Scrum because they were the most widely accepted models and because they complement each other (Charette, 2001). Together they constitute a comprehensive framework covering both the engineering and management process levels.

We then elicited the development activities and put them into a synthesized agile process model. In doing this, however, we observed that some customary development activities were missing. To ensure the comprehensiveness of our synthesized model and to fulfill our third goal, we complemented it with some heavyweight software development activities taken from the standard software process

model (IEEE 12207, 1998). In this way, we could enquire about their applicability in an agile context.

Risk Management Path

The Risk Management Path consisted of two steps similar to those taken in the Agile Path, (1) Establish state of risk management process practice and (2) Study management of risk information. The goal was to establish the state of risk management process practice and to investigate the coverage and amount of risk information managed and documented in traditional and agile environments.

In the first step, we investigated the status of the risk management in 37 software organizations. We did this by comparing the industrial risk management models against a risk management process model that we synthesized from a set of current risk management process models, including the AS/NZS 4360 (Standards Australia, 2004), IEEE 1540 (IEEE 1540, 2001), Project Management Body of Knowledge (PMI, 2004) and the Software Risk Evaluation method (Williams et al, 1999). It was created by studying renowned risk management process models and standards and by synthesizing these models into one common model. Our goal was twofold: (1) to find out the status of risk management process in the industry today, and (2) to evaluate standard process models against the industrial practice.

3.3.3 Tentative Hypothesis Phase

The Tentative hypothesis phase involves further exploration of the identified patterns by studying and verifying them in new contexts. The aim is to specify a tentative hypothesis allowing one to examine the premises for formulating a theory or drawing general conclusions

based on the findings made so far. Hence, we explored the patterns established regarding the agile and risk management state of practice in the previous research phase, including the synthesized models and templates, in new industrial contexts.

The results supported all the previous findings indicating a lack of risk management in agile models. This allowed us to merge the *Agile* and *Risk management* paths into one single research path and to formulate a tentative hypothesis regarding the integration of the risk management agile processes as a possible solution to address the problems identified.

As depicted in Figure 3.2, the *Tentative Hypothesis* phase consisted of three steps: (1) *Establish state of process integration in practice,* (2) *Elicit process integration criteria,* and (3) *Outline integrated model.*

As a result of the work conducted in this phase, we were able to formulate a tentative hypothesis regarding integration as a solution to the problems addressed, which was then evaluated in the fourth and final research phase.

The tentative hypothesis was: Our proposal for integrating the risk management and agile models is a valid solution for addressing the lack of risk management in agile development.

3.3.4 Theory Phase

The fourth and final research phase, called *Theory*, aims at examining the premises for formulating a theory or drawing general conclusions based on the findings made so far. This is realized by collecting new empirical data and examining whether the conclusions as predicted by the tentative hypothesis can be supported in the new data.

As depicted in Figure 3.2, this phase consisted of three steps: (1) evaluate the proposed model in industry, (2) identify improvements to the model, and (3) establish a foundation for future work.

3.4 Questionnaire Design

It is known that the design of a survey instrument can affect the way the interviewees respond and the way interviewers get an understanding of the interviewees' domain (Biemer and Lyberg, 2004). To optimize the interview results, we made an effort to ensure good questionnaire design. It included for instance considering the following aspects:

Ordered Questions*:* We made the effort to assure that each question transitioned smoothly from previous questions. Questionnaires that jump from one unrelated topic to another feel disjointed and are not likely to result in effective interviews. We also ordered questions with respect to their ease of understanding. We first asked easy and general questions, to be then followed by more detailed and complex ones. This allowed the interviewers to follow up on interesting leads and understand the domain from the interviewee's perspective.

Used Cross-Checking Control Questions*:* To establish correctness of the answers, we created several cross-checking questions in the questionnaires.

Provided Expected Answers. Semi-structured and open-ended interviews can often lead to long discussions where it may difficult to distinguish the actual answer to a question. They may also lead to entirely irrelevant answers if the interviewer does not react adequately. To prevent these situations from occurring, we inserted expected answers in the majority of the questionnaires. This helped the interviewer keep the interview aligned with its objectives and to conduct to more effective and efficient interviews.

3.5 Research Sampling

An important aspect of scientific research is sampling. To maintain the validity and the reliability one needs to consider the population and the selection of study objects (Robson, 2002).

In order to find a representative sample in this research, we used convenience sampling. Convenience sampling means that the selection of respondents from a population is based on easy availability and accessibility (Robson, 2002).

The studied population is represented by totally 47 software organizations selected based on criteria that categorized them into certain types and sizes of businesses, certain types of software projects. The interviewees representing the organizations studied, in turn, were selected based on several criteria including minimum experience of the agile and/or risk management processes and role. The minimum requirements were that they were or had been involved in at least one project using the agile and risk management

processes in their organizations. They also represented different roles, which resulted in valuable data representing complementary perspectives of the development process (management, product owners, and developers). Note that the criteria were used to target the organizations and to restrict the population to be studied. It is in this population that the sample was found. Eventually, the organizations and individual interviewees w were randomly selected based on convenience sampling from this population.

3.6 Validity & Reliability

Validity measures the accuracy of a study and its results (Robson, 2002). However, there exists no universal definition of validity. Its meaning differs depending on the discipline, school of thought, research tradition, values and beliefs (Kuhn, 1970) (Little, 1995) (Longino, 1990).

Software engineering is multidisciplinary and hence influenced by several disciplines including logics, mathematics, engineering, social sciences, psychology and business administration. Consequently, this has some particular consequences on the concept of validity.

However, Land (2006) argues that research in software engineering typically focuses on the observation of aspects involving social and psychological aspects in the real world rather than natural aspects. In this respect, the degree of validity should be measured according to the criteria that are found in the tradition of social sciences.

There are four types of validity often referred to be relevant in software engineering research: construct validity, internal validity, external validity (generalizability) and reliability (Wohlin et al., 2003). They are briefly defined below:

Construct Validity*:* ensures objectivity and that the data collected, analyzed and used reflects the phenomenon under study. It can be achieved by triangulating data. Data triangulation involves collecting data from several independent sources, for instance by interviewing more than one individual and by combining different types of data, for instance interviews and statistics (Yin, 2003).

Internal Validity*:* ensures that the conclusions of the study are unbiased and based on accurate descriptions of the data. There are several ways to increase internal validity. For instance, by letting a respondent review the transcript of the interview ensures that the responses have been understood and interpreted correctly. Using a combination of triangulation techniques also reduces the risk of biased results. It could be achieved by combining data, method, theory and/or observer triangulation (Yin, 2003).

External Validity*:* measures the degree to which results of a study with a sample can be generalized to make statements about a much larger population outside the study. To claim external validity also requires that both construct and internal validity are achieved (Maxwell, 1992).

Reliability*:* refers to the repeatability of the study (Robson, 2002). Anybody should be able to follow the same procedure and arrive at the same conclusions. This requires that the research method is comprehensive and documented in detail (Yin, 2003).

Below, follows an evaluation of the degree of validity that has been achieved so far in the work of this thesis. Validity is evaluated according to construct validity, internal validity external validity, and reliability.

Construct Validity*:* To ensure construct validity, data triangulation has been applied during data collection. First, we have strived for achieving construct validity by investigating data from several independent sources. In the interviews, the data is represented by the responses from 55 respondents. Second, by triangulating the data resulting from both literature studies and interviews at large contributes to overall construct validity.

We have also collected different types of data, including qualitative and quantitative data. For instance, we looked for statistical figures supporting the qualitatively observed data.

Internal Validity*:* Internal validity can be claimed for several reasons. We have applied several types of triangulation, including data, method, observer and theory triangulation (Yin, 2003).

First, the accuracy of data has been ensured by data triangulation as described under construct validity. The interviewees have also been provided the opportunity to review and comment on the transcripts of the data collected in interviews.

Second, method triangulation is applied by the use of various data analysis methods. The qualitative approach is dominating but simple forms of measurements have also been carried out, e.g. by analyzing the number of organizations using certain risk management or development activities and by comparing these numbers to the use of these activities as prescribed by existing the risk management standards and agile models.

External validity*:* The research is still ongoing. However, the findings found so far should provide some useful guidelines for various organizations aiming at introducing risk management and

agile processes. At this stage, however, there is no formal foundation for claiming general application of the integration of the risk management and agile models.

Reliability*:* The research and the methods used in each step of the research were documented thoroughly in published research papers and technical reports. These include descriptions of the problem(s) addressed, purpose and goals, research method, and results.

Summing up, this thesis has strived for achieving construct validity, internal validity, external validity, and reliability as described above. The state of the validity could also be discussed by evaluating the results according to Kuhn's (1970) five criteria for distinguishing good scientific theory from bad scientific theory. The thesis at this stage of research could be argued to address three out of five of Kuhn's criteria including accuracy, consistency and scope. However, it still remains to evaluate the criteria of simplicity and fruitfulness, which only ca b answered by further validation.

International Journal of Advanced Multidisciplinary Scientific Research (IJAMSR) ISSN: 2581-4281
www.ijamsr.com CrossRef: https://doi.org/10.31426/ijamsr

MANAGING RISKS: CHALLENGES IN ADOPTION OF AGILE SOFTWARE DEVELOPMENT APPROACH

CHAPTER - 4

EXISTING AGILE METHODS: AN ANALYSIS

Agile - devoting "the quality of being agile; readiness for motion; nimbleness, activity, dexterity in motion" as mentioned in the Oxford Dictionary - software development methods are attempting to offer once again an answer to the eager business community asking for lighter weight along with faster and nimbler software development processes. To name a few of those developed: Adaptive Software Development (ASD), Agile Modeling, Crystal Methods, Dynamic System Development, Lean Development and Scrum. All these methodologies acknowledged that high quality software and more importantly customer satisfaction could only be achieved by bringing "lightness" to their processes. Some of the most used agile methodologies are listed below.

4.1 Extreme Programming (XP)

Extreme programming (XP) has evolved from the problems caused by the long development cycles of traditional development models. The XP process can be characterized by short development cycles, incremental planning, continuous feedback, reliance on communication, and evolutionary design. With all the above qualities, XP programmers respond to a changing environment with much more courage. Further, according to Williams, XP team members spend a few minutes on programming, few minutes on

project management, few minutes on design, few minutes on feedback, and a few minutes on team building many times each day. The term 'extreme' comes from taking these common sense principles and practices to extreme levels. A summary of XP terms and practices is shown below:

- Planning - The programmer estimates the effort needed for the implementation of customer stories and the customer decides the scope and timing of releases based on estimates.
- Small/short releases - An application is developed in a series of small, frequently updated versions. New versions are released anywhere from daily to monthly.
- Metaphor -, The system is defined by a set of metaphors between the easterner and the programmers which describes how the system works.
- Simple Design — The emphasis is on designing the simplest possible solution that is implemented and unnecessary complexity and extra code are removed immediately.
- Refactoring - It involves restructuring the system by removing duplication, improving communication, simplifying and adding flexibility, but without changing the functionality of the program
- Pair programming - All production code is written by two programmers on one computer.
- Collective ownership — No single person owns or is responsible for individual code segments rather anyone can change any part of the code at any time.
- Continuous Integration - A new piece of code is integrated with the current system as soon as it is ready. When integrating, the system is built again, and all tests must pass for

the changes to be accepted.

- 40-hour week - No one can work two overtime weeks in a row. A maximum of 40-hour working week otherwise it is treated as a problem.
- On-site customer -- Customer must be available at all times with the development team.
- Coding Standards - Coding rules exist and are followed by the programmers so as to bring consistence and improve communication among the development team.

The lifecycle of an XP project, shown in Figure 4.1, is divided into six phases: Exploration, Planning, Iterations to release, Production, Maintenance and Death.

In the *Exploration phase,* the customer writes out the story cards they wish to be included in their program. This leads to *Planning phase* where a priority order is set to each user story and a schedule of the first release is developed. Next in the *Iterations to Release phase,* the development team first iteration is to create a system with the architecture of the whole system then continuously integrating and testing their code. Extra testing and checking of the performance of the system before the system can be released to the customer is done in the *Production phase.* Postponed ideas and suggestions found in this phase are documented for later implementation in the updated releases made in the *Maintenance phase.* Finally the *Death Phase* is near when the customer have no more stories to be implemented and all the necessary documentation of the system is written as no more changes to the architecture, design or code is made.

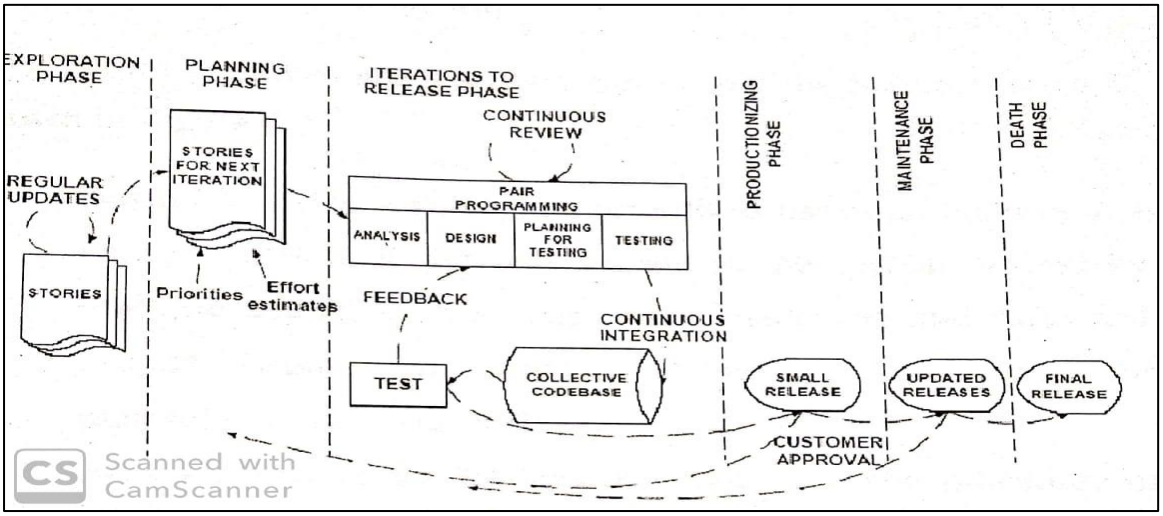


FIG.4.1: Lifecycle Of The XP Process

4.2 Scrum

Scrum is an iterative, incremental process for developing any product or managing any work. Scrum concentrates on how the team members should function in order to produce the system flexibility in a constantly changing environment. At the end of every iteration it produces a potential set of functionality. The term 'scrum' originated from a strategy in the game of rugby where it denotes "getting an out-of-play ball back into the game" with teamwork.

Scrum does not require or provide any specific software development methods/practices to be used. Instead, it requires certain management practices and tools in different phases of Scrum to avoid the chaos by unpredictability and complexity.

Key Scrum practices are discussed below and the Scrum process is shown in Figure 4/2.

Product Backlog: This is the prioritized list of all features and changes that have yet to be made to the desired system by multiple actors, such as customers, marketing and sales and project team. The Product Owner is responsible for maintaining the Product Backlog.

Sprints - Sprints are 30-days in length, it is the procedure of adapting to the changing environmental variable (requirements, time, resources, knowledge, technology etc.) and must result in a potentially shippable increment of software. The working tools of the team are Sprint Planning Meetings, Sprint Backlog and Daily Scrum meetings.

Sprint Planning Meeting: Sprint planning meeting is first attended by the customers, users, management, Product owner and Scrum Team where a set of goals and functionality are decided on. Next the Scrum Master and the Scrum Team focus on how the product is implemented during the Sprint.

Sprint Backlog - It is the list of features that is currently assigned to a particular Sprint. When all the features are completed a new iteration of the system is delivered.

Daily Scrum - It is a daily meeting for approximately 15 minutes, which are organized to keep track of the progress of the Scrum Team and address any obstacles faced by the team.

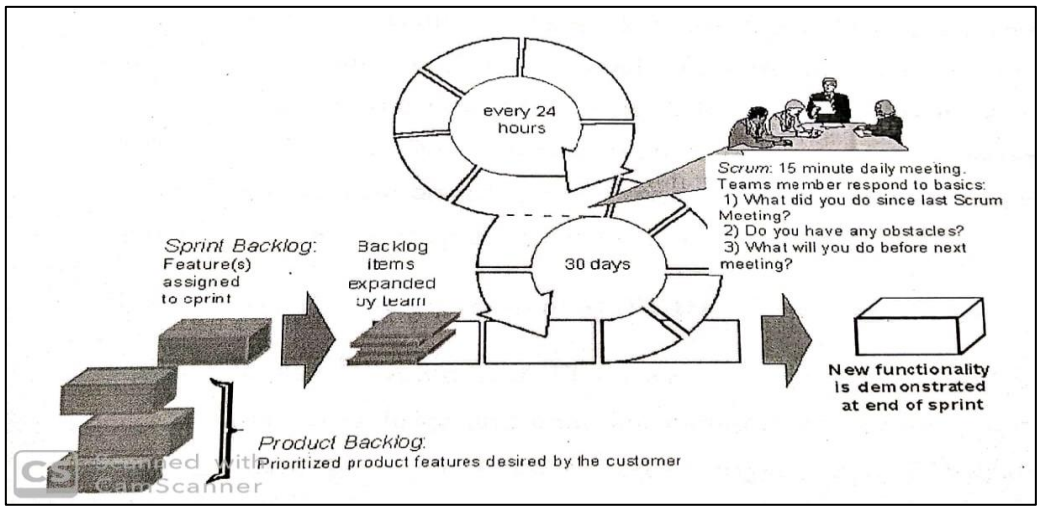


FIG. 4.2 Scrum Process

The Scrum process may change the job description and customs of the Scrum project team considerably. For example, the project manager, i.e. the Scrum Master, does no longer need to organize the team, but the team organizes itself and makes decisions on what to do. Ken Schwaber illustrates, "Most management is used to directing the project, telling the team what to do and then ensuring they do it. Scrum relies on self-organization, with the team deciding what to do while management runs interference and removes roadblocks". Scrum has been successfully used over thousands of projects in 50 organizations producing significant productivity improvement.

Rising and Janof suggest that "Clearly, Scrum is not an approach for large, complex team structures, but we found that even small, isolated teams on'a large project could make use of some elements of Scrum. This is true process diversity". Recently, efforts have been made to combine XP practices with Scrum project management framework to form an integrated package for software development team. However, more study is needed to support this package.

4.3 Feature Driven Development (FDD)

Feature Driven Development (FDD) was used for the first time in the development of a large and complex banking application project in the late 90's. Unlike the other methodologies, the FDD approach does not cover the entire software development process, but rather focuses on the design and building phases.

The first three phases are done at the beginning of the project. The last two phases are the iterative part of the process which supports the agile development with quick adaptations to late changes in requirements and business needs. The FDD approach includes frequent and tangible deliverables, along with accurate monitoring of the progress of the report.

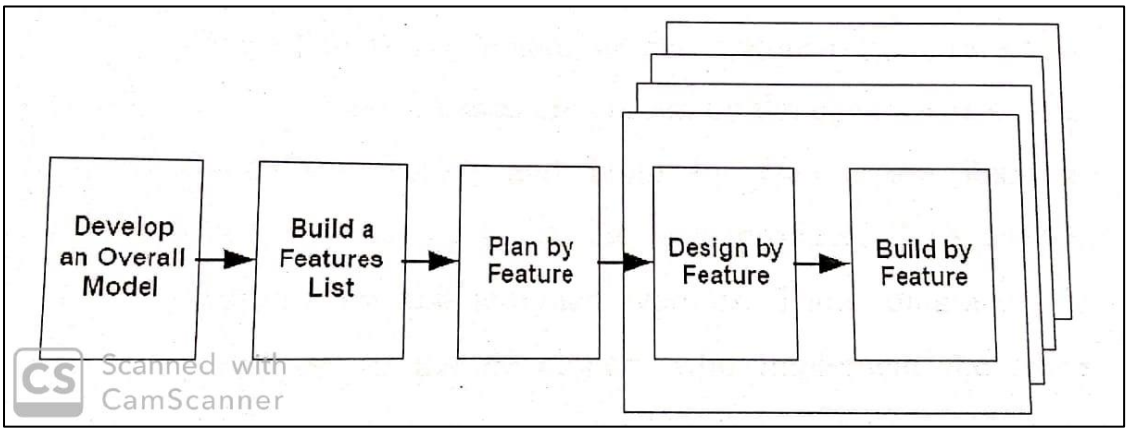


FIG. 4.3: Feature Driven Development Process

Develop an Overall Model: A high level walkthrough of the system scope and its context is performed by the *domain expert* to the team members and *chief architect.* Documented requirements such as use cases or functional specifications are developed.

Build a Features List: A categorized list of features to support the requirements is produced.

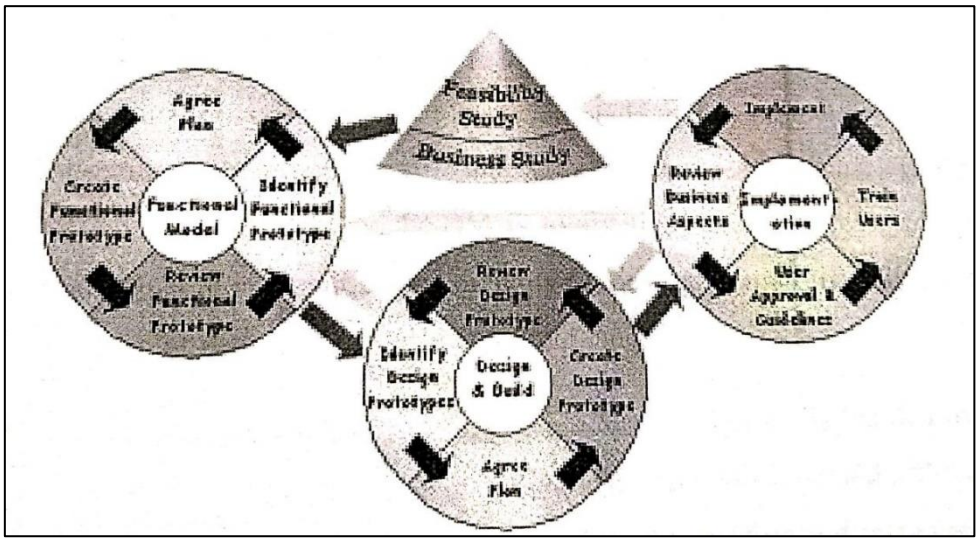
Plan by Feature: The development team orders the feature sets according to their priority and dependencies and assigned to chief programmers. Furthermore, the classes identified in the first phase are assigned to class owners (individual'developers). Also schedule and milestones are set for the feature sets.

Design by Feature & Build by Feature: Features are selected from the feature set and feature teams needed to develop these features are chosen by the class owners. The design, by feature and build by feature are iterative procedures during which the team produces the sequence diagrams for the assigned features. These diagrams are passed on to the developers who implement the items necessary to support the design for a particular feature. There can be multiple feature teams concurrently designing and building their own set of features. The code developed is then unit tested and inspected. After a successful iteration, the completed features are promoted to the main build.

3.4 Dynamic System Development Method (DSDM)

The DSDM, Dynamic System Development Method, was developed in the United Kingdom in the mid- 1990. It is a blend of, and extension to, rapid application development and Iterative development practices. Martin Fowler, one of the writers of the Agile Manifesto, believes, "DSDM is notable for matures much of the infrastructure of more traditional methodologies, while following the principles of the agile methods

approach". The fundamental idea behind DSDM is to fix time and resources, and then adjust the amount of functionality accordingly rather than fixing the amount of functionality in a product, and then adjusting time and resources to reach that functionality. DSDM consists of five phases (Figure 4.4):



96

FIG.4.4: DSDM Process

Feasibility Study: In this phase a decision is made whether to use DSDM or not. This is determined by judging him type of project and, organizational and people issues. In addition, two work products are produced; a feasibility report and an outline plan for development.

Business Study: The recommended approach of this phase is to organize a workshop to help understand the business domain of the project. The key outputs of this section are System architecture definition and an Outline prototype plan.

Functional Model Iteration: First iterative phase. This phase involves analysis, coding and prototypes. The results gained from these prototypes are used in improving the analysis models. The key output is a functional model which consists of the prototype code analysis models.

Design and Build Iteration: the system is mainly built in this phase. The design and function prototypes are reviewed by the users and further development is based on the users' comments.

Implementation: In this final phase the system is handed over to the users. Training is provided. User Manuals and a Project Review Report. However, the DSDM iterative and incremental nature means that maintenance can be viewed as continuing development. Instead of finishing the project in one cycle, the project can return to any of the phases, Design and Build phase, Functional Model Iteration, or even Feasibility phase so that previous step can be refined.

There are nine practices that define the ideology and the basis for all activity in DSDM. Some of the underlying principles include active user interaction, frequent deliveries, empowered teams, and testing throughout the cycle. There is an emphasis on high quality and Adaptivity towards changing requirements. Like other agile methods, DSDM approaches iterations as short time-boxed cycles of between two and six weeks.

4.5 Adaptive Software Development (ASD)

Adaptive Software Development (ASD), developed by James A. Highsmith, offers an agile and adaptive approach to high-speed and high-change software projects. It is not possible to plan successfully in a fast moving and unpredictable business environment. In ASD, the static plan-design life cycle is replaced by a dynamic speculate-collaborate-learn life cycle.

ASD focal point is on three non-linear and overlapping phases (Figure 4.5);

Speculate: To define the project mission, make clear the realization about what is unclear.

Collaborate: Highlights the importance of teamwork for developing high-change systems

Learn: This phase stresses the need to admit and react to mistakes, and that requirement may well change during development.

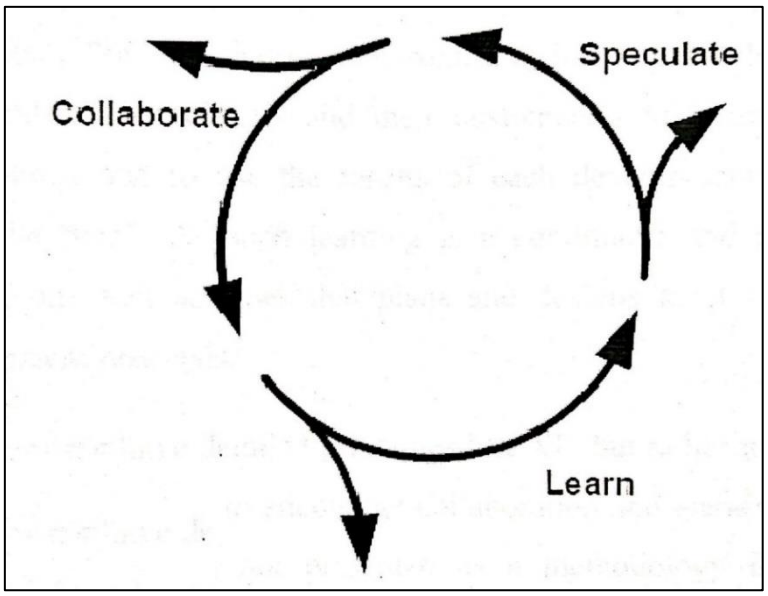


FIG. 4.5: ASD Lifecycle

Since outcomes are naturally unpredictable, Highsmith views planning as paradox in an adaptive environment. Normally in traditional planning when things do not go to plan it is seen as a mistake that should be corrected. However in an adaptive environment devialtions guide us towards the correct solutions.

ASD focuses more on the results and their quality than the tasks or the process used for producing the results. In an unpredictable environment you need people to collaborate in a certain manner to deal with the uncertainty. Management is more about encouraging communication rather than telling people what to do, so that more creative answers are delivered.

In traditional predictive environment, designs have followed the same way they were laid out, therefore learning is discouraged. Highsmith points out, "In an adaptive environment, learning challenges all stakeholders – developers and their customers – to examine their assumptions and to use the results of each development cycle to

adapt the next". As such, learning is a continuous and important feature, one that assumes that plans and designs must change as development proceeds.

ASD does not have detailed principles like XP, but rather it provides a framework for how to encourage collaboration and learning within the project. ASD is not presented as a methodology for doing software project, but rather it is an approach or an attitude that must be adopted by an organization when applying agile processes.

4.6 Agile Manifesto

On February 2001, seventeen representatives from the different agile methods decided to form an Agile Alliance to better promote their views and what emerged was the Agile 'Software Development' Manifesto. Most of the agile techniques have been used by developers before the alliance, but it is not till after the alliance that these techniques were grouped together into workable framework.

The focal values honored by the agilists are presented in the following:

Individuals and interactions over processes and tools **Working software** over comprehensive documentation **Customer collaboration** over contract negotiation **Regarding to change** over following a plan. That is, while there is value in the items on the right, we value the items on the left more.

The 12 principles of the Agile Software development made by the Agile Manifesto:

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity—the art of maximizing the amount of work not done--is essential.
- The architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

4.7 General Features and Comparison of Agile Methodologies

Comparison often implies valuing one method over the other. In this section Table 4.1 below discusses each method using three selected aspects: key points, special features and identified weakness. Key points detail the methods, principles, aspects or solution. Special features describe one or several aspects of the methods that differentiate them from others. Finally, identified weakness relates to some aspects of a method that have been documented in literature.

Method Name	Key Points	Special features	Identified weakness
ASD	Adaptive culture, collaboration, mission-driven component based iterative development	Organizations are seen as adaptive systems. Creating an emergent order out of a web of interconnected individuals	ASD is more about concepts and culture than the software practice
DSDM	Application of controls to	First truly agile software	While the method is

	RAD, use of timeboxing and empowered DSDM teams.	development method, use of prototyping, several user roles : "ambassador", "visionary" and "advisor"	available, only consortium members have access to white papers dealing with the actual use of the method
XP	Customer driven development, small teams, daily builds	Refactoring - the ongoing redesign of the system to improve its performance and responsiveness too change	While individual practices are suitable for many situations, overall view & management practices are given less attention
SCRUM	Independent, small, self-organizing	Enforce a paradigm shift from the	While Scrum details in specific how to

	development teams, 30-day release cycles.	"defined and repeatable" to the "new product development view of Scrum"	manage the 30-day release cycle, the integration and acceptance tests are not detailed
FDD	Five-step process, object-oriented component (i.e. feature) based development.	Method simplicity, design and implement the system by features, object modeling	FDD focuses only on design and implementation. Needs other supporting approaches.

Table 4.1: General Features of Agile Methods

In the software development viewpoint, ASD is the most abstract method. Its key goal "creating an emergent order out of a web" may be appealing, but practitioners may experience difficulties in translating the methods new concept to their use. XP represents practice-oriented viewpoints. It contains a number of empirically validated practices found useful by developers. DSDM is

differentiated from the other methods because of its use of prototyping. Also DSDM makes use of user roles such as ambassadors, visionary and advisor that other methods do not use.

The drawback of using DSDM is that the need to belong to the DSDM consortium in order to gain an access to the white papers discussing different aspects of the method. FDD assumes that some work has already been done with the project. Finally Scrum is a project management approach that relies on self-organizing independent teams implementing a software project in 30-day cycles called sprints.

One of the main decisive issues in the different agile methods is the size of the development team. XP and Scrum focus on small teams, preferably less than 10 developers. FDD, ASD and DSDM claim to be capable of up to 100 developers. However, when the development team size gets larger, the amount of documentation is likely to increase, thus making the project “less agile”. When the development group exceeds 20 developers, agilists’ put a greater deal into solving communication problems. As Alistair Cockburn states, “Good people are key to success with big teams”.

4.8 Characteristics of Agile Methodologies

According to Highsmith and Cockburn, "what is new about agile methods is not the practices they use, but their recognition of people as the primary drivers of project success, coupled with an intense focus on effectiveness and maneuverability. This yields a new combination of values and principles that define an agile world view."

Highsmith further transcribes from the book *Agile Competitors and Virtual Organizations* the definition of agility: "Agility... is a comprehensive response to the business challenges of profiting from rapidly changing, continually fragmenting, global markets for high-quality, high-performance, customer-configured goods and services."

The following principles of agile methodologies are seen as the main differences between agile and heavyweight:

People Oriented: Agile methodologies consider people - customers, developers, stakeholders, and end users — as the most important factor of software methodologies. As Jim Highsmith and Alistair Cockburn state, "The most important implication for managers working in the agile manner is that it places more emphasis on people factors in the project: amicability, talent, skill, and communication". If the people on the project are good enough, they can use almost any process and accomplish their assignment. If they are not good enough, no process will repair their inadequacy. As Highsmith highlights, "... people trump process... ".

Adaptive: The participants in an agile process are not afraid of change. Agilists welcome changes at all stages of the project. They view changes to the requirements as good things, because they mean that the team has learned more about what it will take to satisfy the market. Today the challenge has not stopped changing, but rather determining how to better handle changes that occur throughout a project. "External Environment changes cause critical variations. Because we cannot eliminate these changes, driving down the cost of responding to them is the only viable strategy".

Conformance to Actual: Agile methodologies value conforming to the actual results as opposed to conform to the detailed plan. Highsmith states, "Agile projects are not controlled by conformance to plan, but by conformance to the business value". Each iteration or development cycle adds business value to the ongoing product. For agilists, the decision on whether business value has been added or not given by the developers but instead by end users and customers.

Balancing Flexibility and Planning: Plans are important, but the problem is that software projects cannot be accurately predicted far into the future, because there are so many variables to take into account. A better planning strategy is to make detailed plans for the next few weeks, very rough plans for the next few months, and extremely crude plans beyond that. In this view one of the main sources of complexity is the irreversibility of decisions. If you can easily change your decisions, this means it's less important to get them right - which makes your life much simpler. The consequence of agile design is that designers need to think about how they can avoid irreversibility in their decisions. Rather than trying to get the right decision now, look for a way to either put off the decision until later or make the decision in such a way that you will be able to reverse it later on without too much difficulty.

Empirical Process: Agile methods develop software as an empirical (or nonlinear) process. In engineering, processes are either defined or empirically. In other words, defined process is one that can be started and allowed to run to completion producing the same results every time. In software

development it cannot be considered a defined process because too much change occurs during the time that the team is developing the product. Laurie Williams states, "It is highly unlikely that any set of predefined steps will lead to a desirable, predictable outcome because requirements change as technology changes, people are added and taken off the team, and so on".

Decentralized Approach: Integrating a decentralized management style can severely impact a software project because it could save a lot of time than an autocratic management "process. Agile software development spreads out the decision making to the developers. This does not mean that the developers take on the role of management. Management is still needed to remove roadblocks standing in the way of progress. However, management recognizes the expertise of the technical team to make technical decisions without their permission.

Simplicity: Agile teams always take the simplest path that is consistent with their goals. Fowler states, "They (agile teams) don't anticipate tomorrow's problems and try to defend against them today". The reason for simplicity is so that it will be easy to change the design if needed at a later date. Never produce more than what is necessary and never produce documents attempting to predict the future as documents will become outdated. "The larger the amount of documentation becomes, the more effort is needed to find the required information, and the more effort is needed to keep the information up to date".

Collaboration - Agile methods involve customer feedback on a regular and frequent basis. The customer of the software works closely with the development team, providing frequent feedback on their efforts. As well, constant collaboration between agile team members is essential. Due to the decentralized approach of the agile methods, collaboration encourages discussion. As Martin Fowler describes, "Agile teams cannot exist with occasional communication. They need continuous access to business expertise".

Small Self-organizing teams - An agile team is a self organizing team. Responsibilities are communicated to the team as a whole, and the team determines the best way to fulfill them. Agile teams discuss and communicate together on all aspects of the project. That is why agility works well in small teams. As Alistair Cockburn and Jim Highsmith highlight, "Agile development is more difficult with larger teams. The average project has only nine people, within the reach of more basic agile processes. Nevertheless, it is interesting to occasionally find successful agile projects with 120 or even 250 people".

International Journal of Advanced Multidisciplinary Scientific Research (IJAMSR) ISSN: 2581-4281
www.ijamsr.com CrossRef: https://doi.org/10.31426/ijamsr

MANAGING RISKS: CHALLENGES IN ADOPTION OF AGILE SOFTWARE DEVELOPMENT APPROACH

CHAPTER - 5

INTEGRATING RISK MANAGEMENT & AGILE PROCESSES

In this chapter, we outline a model integrating risk management and agile development. To begin with, however, we examine the agile processes from a risk management perspective to identify the risk management aspects that are absent in the agile model. We do this by using a set of fundamental risk management aspects as our comparison criteria. We continue by proposing a model for integrating the risk management and agile process models. It is created based on the identification of integration points between the two processes as excited from studies of the integration of agile and risk management processes in the industry. Using this integration model, we then outline the integrated model.

5.1 Comparison Of Risk Management And Agile Models

Effective risk management prevents the likelihood that undesirable problems occur or it decreases the severity of their consequences, should they occur (IEEE 1540, 2001). By identifying and attending to risks, it aids in making informed decisions and taking appropriate measures before risks become problems. Risk management helps avoid problems, rework, disasters and it stimulates successful project outcomes. For this reason, it should be an inherent component of software development (Boehm, 1988

Agile software development models claim to be risk-driven (Beck, 2004). They state that their iterative approach enables continuous attention to risks throughout the whole development project. When studying the agile and risk management processes, however, we have found that the agile development models' implement few risk management practices.

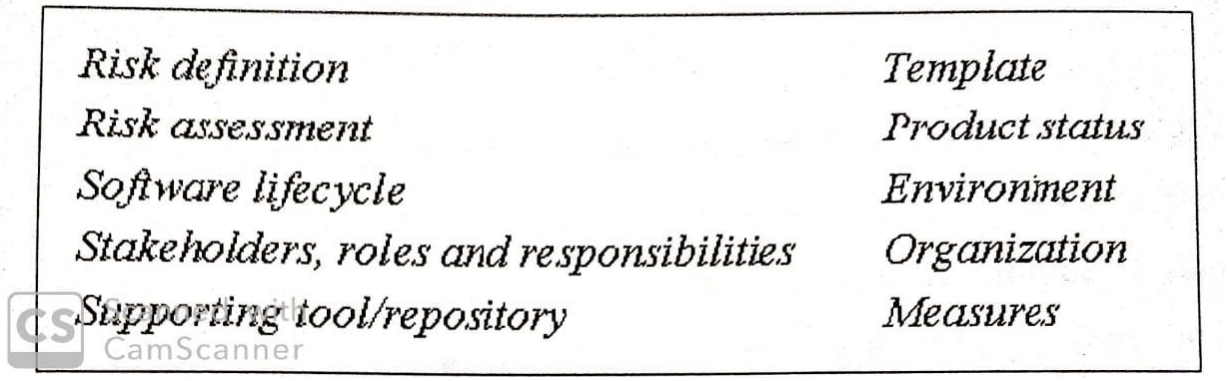


FIG. 5.1 Fundamental Risk Management Aspects

Extreme Programming claims to be model that addresses risk at all levels of the development process (Beck, 2004). However, risk management is not explicitly described anywhere in the XP model. There are no specific guidelines for managing risks or activities describing how to identify, analyze and control them.

Scrum claims to be risk-driven because it is based on an iterative and incremental approach enabling early and frequent feedback making it possible to identify and reduce risk early (Schwaber and Beedle, 2001). In contrast to XP, however, Scrum makes some explicit statements about risk management during the development process. In Scrum, risk management is described as a part of the planning phase, where it is stated that risk should be identified, assessed and actions for controlling the identified risk defined (Scrum, 2003). The risks should be listed and planned for when defining the project.

However, it is not explained how these risk management tasks can be carried out, and there are no precise guidelines for how to identify, classify, assess or manage risks.

These are examples of some general observations of the state of risk management in Scrum and XP. However, in order to more exactly identify the aspects of risk management that are omitted requires a more systematic examination of the agile models from a risk management perspective.

To accomplish this, we conducted an in-depth study of risk management standards and identified some fundamental aspects of risk management that we use as our comparison criteria. They are the following: (1) *'Risk definition,* (2) *Risk assessment,* (3) *Software lifecycle,* (4) *Templates,* (5) *Tools/Repositories,* (6) *Stakeholders, roles and responsibilities* (7) *Product status,* (8) *Environment,* (9) *Organization,* and (10) *Measures.* Each aspect is listed in Figure 5.1 and briefly described below:-

Risk definition*:* Risk is defined as an event or a condition that may affect the outcome of a project (IEEE 1540, 2001). It is characterized by two distinctive elements: probability and impact (Boehm, 1991). The probability defines the likelihood that a risk event may occur. The impact defines the outcome of a risk. Risk can be either a loss or a gain (an opportunity) (Standards Australia, 2004). A loss is an unwanted or negative effect, whereas a gain is a positive or progressive effect. In this comparison, we investigate whether and how Scrum and XP define risk. A definition is a prerequisite for defining the risk management process. It helps in understanding the process and facilitates the comparison process.

Risk Assessment: To make right and informed decisions, it is of importance to correctly identify and analyze risks. Hence, one needs to classify and assess them (Carr et al., 1993). Risk taxonomies (classifications) help identify risks systematically and thereby facilitate the analysis process (Williams et al., 1999). However, they do not suffice for an exhaustive risk analysis. They need to be complemented with various assessment properties (attributes) such as *Risk Probability, Risk, Impact, Risk Priority, Risk, Exposure,* and the like (Boehm, 1991). Values should be assigned to these attributes. Due to the fact that risk assessment is subjective, it may be difficult to assign relevant values (Williams et al., 1999).

Hence, guidelines for assessing risks in the form of various techniques should be provided. Taxonomies, assessment attributes and techniques greatly help organizations in planning various measures such as designation of risk management, estimation of the mitigation effort size, and identification of policies to guide them. We examine if XP and Scrum specify risk taxonomies. We also examine if they suggest attributes and techniques for assessing risks.

Software Lifecycle: Effort spent on mitigating risks within development may be wasted if one does not consider risks within the whole lifecycle process (IEEE 1540, 2001) (PMI, 2001). Risk management activities may differ in various lifecycle processes. We investigate whether the agile models studied cover risk management within the total lifecycle process, what lifecycle processes they approach and whether they provide guidelines for these processes.

Stakeholders, Roles and Responsibilities: Stakeholder roles are individual roles or groups of roles who have a stake in or may be impacted by a giv«n activity (IEEE 1540, 2001). Stakeholders can

either be internal or external (Standards Australia, 2004). Internal stakeholders include any management or technical roles participating in a project, including project managers, developers, testers, maintainers, product owners, business analysts and managers, quality managers and support personnel. External stakeholders are other roles, such as customers, contractors, suppliers and sponsors.

The coverage of stakeholder roles within risk management is very important. It is only then one may be sure that all the risk sources and targets have been identified and scrutinized from all possible perspectives. Designation of roles is a prerequisite for defining risk management process and responsibilities within the process (IRM, 2002) (IEEE1540, 2001).

S*upporting Tools/Repository:* To enable effective risk information management, analysis and tracking, organizations need repositories (preferably electronic) for documenting risks and the risk management process (IEEE 1540, 2001). They should also be able to extract important experience and lessons learned which they may in turn use in various contexts, such as process assessment, improvement, root cause analysis, resource assignment, and the like. For this, they need experience base recording historical risk information. We investigate whether the agile models studied suggest the use of tools and repositories for documenting risk management information.

Template*:* A clfar, complete and correct risk description is an important prerequisite for its effective management (Carr et al. 1993) (Hulett, 2001). To aid in maximizing the quality of the risk information, one should provide guidelines for what information should be managed during the risk management process (Williams et

al., 1999).

Product Status: Instances of the risk management process vary strongly with respect to the quality of the product, its life expectancy, and life cycle stage (IEEE 1540, 2001). For this reason, risk management models should consider the product aspects when managing risks. Risk management may vary greatly in aged systems whose quality is undermined, life expectancy is low, and the system is close to retirement.

Environment: To implement risk management effectively, the team should consider the project in its cultural, social, international, political and physical contexts (PMI, 2004). Systems may be developed in distributed and non-distributed environments. In a non-distributed environment, the team(s) is (are) co-located and work(s) together. The environment is 100% non-distributed if the customer is internal.

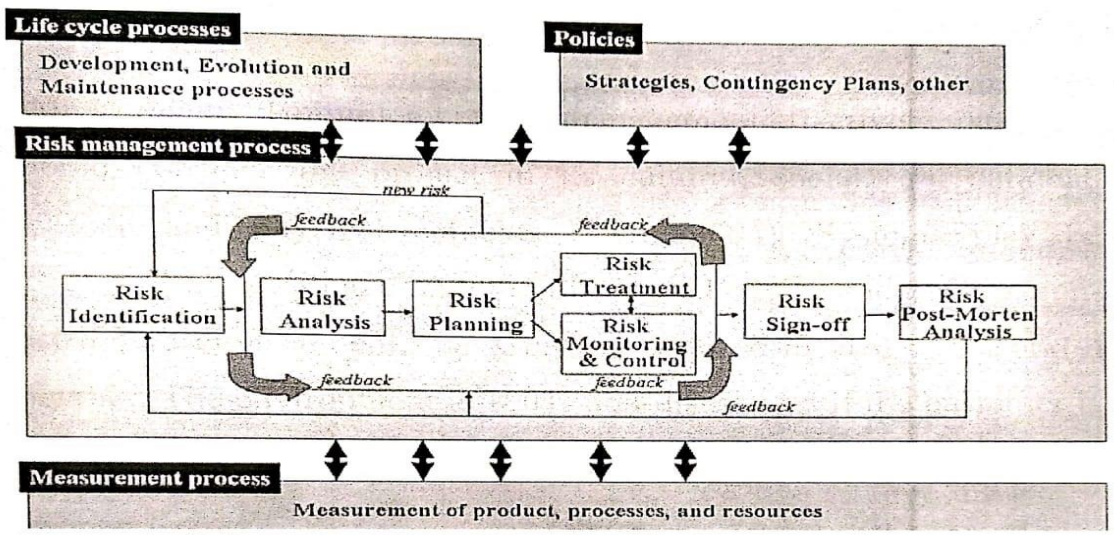


FIG. 5.2 Risk Management Processes

Regarding the distributed environment, it can span from having to deal with only external customers to having to deal with distributed teams in different organizations and different countries, where aspects such as culture can play a crucial role for the effectiveness (Hofstede, 2003). Hence, the degree of distribution and its associated risks should be considered in risk management.

Organization*:* To adopt a risk management program successfully, factors such as attitude towards risks, organizational maturity, competency and training should be considered (PMI, 2004). It is harder to implement risk management effectively in immature and incompetent organizations or with risk averse attitude (Hulett, 2001). Hence, all the stakeholders involved should acquire proper training in the product they produce/acquire and the processes they use.

Measures*:* One of the main purposes of risk management is to identify uncertainties and take action to either remove or transform them into acceptable risks (Xu et al., 2005). For this, one needs a portfolio of appropriate measures in place. Such a portfolio covers processes, suggesting activities for attending to risks, resources required for performing the processes and policies for ensuring that procedures and strategies for conducting risk management are defined and agreed upon (PMI, 2004).

The middle box in Figure 5.2 outlines a risk management process as covered by most of the risk management examples and standards today, e.g. by the AS/NZS 4360 (2004), IEEE1540 (2001) and PMBoK (2004). To provide useful feedback to the organization, the process needs to be integrated with other organizational processes such as various lifestyle and measurement processes and policy management. These processes should continuously provide feedback

on risks, their progress, resources used, effectiveness of the resources and policies chosen, and plans for contingency readiness (Westfall, 2001). We investigate whether the models studied suggest risk management processes and whether they, provide guidelines for resource management, process measurement, policy management, and integration with other processes.

The results of the comparison show that the agile models studied make some assertions about risk management. However, they do not provide any detailed suggestions for managing risks, thus leaving many areas unattended. The comparison results for each risk management aspect are listed and briefly discussed below. They are summarized In Table 5.1.

Risk Definition: The definition of risk is the same risk management and agile models. However, the recognition of risk as an opportunity is not always as explicit in the risk management standards as in the agile models.

Risk Assessment*:* Risk management is aimed at mitigating risks and thus provides well established techniques for identifying and assessing risk. Scrum and XP do not provide any guidance at all. In this respect, the agile models can learn from risk management models to ensure effective risk management.

Software Lifecycle*:* None of the risk management models studied encompasses management of risks within other software lifecycle processes than development. The iterative, incremental and evolutionary approach in agile development means that in each iteration, enhancements, corrections and minor improvements are being made in parallel. Hence, the agile models cover to some extent development, enhancements and problem resolution in one and the

same iteration. In this respect, one could argue that they encompass other lifecycle processes than development. However, the management of risk is not made an explicit part of the product lifecycle. We believe that the agile approaches should consider risk over the entire lifecycle.

Stakeholders, Roles And Responsibilities: Both XP and Scrum suggest that the, team shares the responsibility for project success. They do not, however suggest particular guidance for the roles that might be relevant in the risk management process. In our opinion, they should explicitly identify and recognize all the stakeholders, roles and their responsibilities to ensure that all the risk sources and targets have been identified and scrutinized from all possible perspectives.

Template*:* The agile models do not suggest templates for communicating information about risks and their management, despite the fact that high risk information quality is one of the most important prerequisites for effective risk mitigation (Carr et al. 1993). In our opinion, agile process models should be complemented with guidance for what information to collect and how to structure it.

Supporting Repository/Tool*:* Neither Scrum nor XP prescribe any recording of risk information. Generally, risk management models advocate permanent storage, whereas agile models advocate temporary one. Risk management models suggest a risk management repository and experience base, supported by electronic tools. The agile models, on the other hand, mainly prescribe the informative workspace. From a risk management perspective, permanent recording of risk information is the only way of assuring that all important information is being remembered, paid heed to and that

lessons learned can be easily disseminated to enable process improvement (IEEE 1540, 2001). In this respect, we suggest that agile methods should consider the idea of permanent storage as a compliment

Product Status*:* None of the agile models studied relates product status to specific risk management measures. This is an important factor to consider when tailoring the risk management process to specific product status and allocating resources to it.

Environment*:* The agile models studied provide some practical guidelines on the environmental aspects and physical context of development that could be used also for risk management, for instance the XP practices of *Sit together* and *Informative Workspace* (Beck, 2004). However, they do not consider distributed environments. Agile development models have spread to distributed environments, but more evidence is needed from a risk management perspective.

Organization*:* Regarding guidance for managing organizational issues such as attitudes, training, and maturity, XP provides relatively detailed guidance on how to deal with teambuilding, training and competence development, for instance by practices such as *Whole team* and *Pair Programming* and by principles such as *Diversity* and *Failure* (Beck, 2004).

We believe these could be useful also for risk management. However, they do not address risk management in particular. Hence, the agile models should be more active in recognizing the importance of these organization issues also for risk management purposes. It is only in this way, one may make sure that risk management is implemented and run in an effective way.

Measures*:* The risk management models studied suggest integration of risk management with measurement processes. However, they do not suggest any metrics or measurement models for actually measuring risk management, nor do they provide guidance on how to integrate the processes.

The agile models integrate measurement processes in their own specific way. Scrum uses empirical process control, implying that one continuously inspects the process. Hence, our conclusion is that agile models seem to integrate measurement models. However, concerning the integration with other organizational processes, such as risk management, there are no guidelines. We suggest that the agile models complement their models with suggestions for how to integrate with other organizational processes to ensure useful and valuable communication and feedback on the information needed in the entire organization.

In addition, neither XP nor Scrum explicitly identifies various risk management phases. We believe that this is a serious omission. A phase such as the *Risk Sign-Off phase* is pivotal in making sure that serious risks have been attended to. Lack of this aspect could also lead to serious legal consequences.

Table 5.1 Results of comparison of agile and risk management models

Criteria	Present	Comment
Risk definition	Yes	Risk is also recognized as an opportunity.
Risk assessment	No	
Software lifecycle	No	
Stakeholders, roles and responsibilities	Partly	The agile team shares the responsibility for the project success, but there is no recognition of its implications for the management of risk.
Templates	No	
Repository/tool	No	
Product status	No	
Environment	Partly	Although some guidelines are provided for distributed agile projects, risk management aspects are not included.
Organization	Partly	The XP principles covers organizational aspects on the individual and team level, but not so much on the organizational level (e.g. maturity) and risk management is not included.
Measures	No	

As summarized in Table 5.1, the results of this comparison of the agile and risk management processes clearly indicate that agile development models implement few risk management practices. One can conclude that a lot remains to be done to make the agile models more risk-driven.

In the following, we will use these results and the fundamental risk management aspects listed here as a starting point for reasoning about the integration of the agile and risk management models. We start by studying the possible ways of integration and then continue by proposing a model for integrating them.

5.2 Creation of Integration Model

By definition, an integrated process is "a series of interrelated activities that share and exchange data to achieve a common purpose" (Byrnes, 2006).

Hence, in this context, integration is the activity of connecting the agile and risk management processes and making them communicate with each other to support the goal of organization-wide management of risks in the agile process.

Integrating risk management with agile development requires the integration of two different types of processes, management (risk management) and engineering (agile). The process integration endeavor has been undertaken before, for instance, within the enterprise architecture and business process modeling fields (Zachman, 1987). However, very little has been done regarding the integration of processes from the two disciplines studied here, except for a few attempts (Armenta and Gaono, 2008). More importantly, there is no general integration method or process satisfying our purposes. It has also been stated that the integration of management and engineering processes needs further clarification (Chroust and Hardt, 1996). For these reasons, we need to create our own integration model.

By studying various agile process scenarios in which risk management was executed in the industry, we were able to identify several integration points between the two processes. By integration point, we mean anything that explicitly connects the two processes and realizes an exchange between them supporting the risk management goals of the organization.

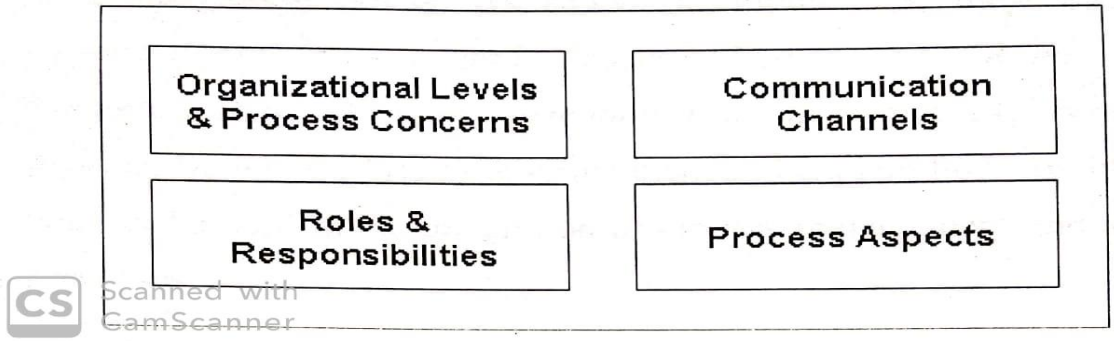


FIG. 5.3: Basic Process Integration Points Of Integration Model

An integration point may involve (a) means of exchange, such as people, tools and documents, (b) activities, such as communication, and (c) cross organizational flows, such as process flows, work flows and data flows.

Because there were no general models for integrating the processes of interest here, we started with the most rudimentary set of integration points as identified in our studies of integration in the industry. Using them, we created a simple integration model describing four basic integration points (see Figure 5.3). We callthem (1) Organizational Levels and Process Phases, (2) Roles and Responsibilities, (3) Communication Channels, and (4) Process Aspects.

5.3 **Integration Model**

In this section, we describe the integration model. It consists of four basic integration points. As shown in Figure 5.3, they are (1) *Organizational Levels and Process Phases,* (2) *Roles and Responsibilities,* (3) *Communication Channels,* and (4) *Process Aspects.* Below, we briefly describe them and explain how we intend them to be used in the integration of the risk management and agile processes.

5.3.1 Process Phases And Organizational Levels

A critical integration point involves the identification of *when* and *where* risk management takes place in the development process. The agile development process consists of several phases and it spans over several organizational levels. By finding out where risk management occurs clarifies the various points of integration between the two processes. Hence, one needs to map out where the risk management takes place in the agile process. It is a prerequisite for further integration.

5.3.2 Roles and Responsibilities

In the context of process integration, roles comprise a primary crossing point for the exchange of information between the two processes. According to standard risk management practice, risks should also be owned by various roles to ensure that they are effectively managed (PMI, 2004). Hence, to establish productive risk management, one needs to identify appropriate roles and their responsibilities in the integrated process. A role is not always equivalent to one specific person. One person may have several roles and one role may be assigned to several people.

5.3.3 Communication Channels

Agile development and risk management are communication intensive processes. The communication gets intensified if risks are managed across the whole organization and several processes and process phases. To achieve effective communication, one needs, designate appropriate communication channels. Such channels, integrate the processes and their phases by specifying the flow of communication on an organization-wide level.

5.3.4 Process Aspects

Processes are varied and dynamic. Their instances vary due to many different factors affecting the process design. Hence, such factors need be considered. In this integration model, they are represented by the fundamental aspects of risk management listed in Figure 5.1. Each of these determines the magnitude of risk management required within the integrated process, thus aiding in adapting an instance of the integrated model to the specific situation at hand. Hence, they constitute guidelines for making decisions on how to adapt the process according to the needs of risk management. The following aspects should be considered when tailoring instances of the integrated process:

Risk Definition*:* constitutes a control that there exists a comprehensible risk definition to facilitate the communication of risk within the organization.

Risk Assessment: defines a control that there are guidelines for assessing risks effectively.

Software Lifecycle*:* defines a control for designating adequate risk management activities according to the varying needs of different software lifecycle phases.

Stakeholders, Roles And Responsibilities*:* defines a control for streamlining the degree of involvement of various stakeholders, roles and their responsibilities as determined by factors such as project size and project risk profile.

Supporting Tools/Repositories*:* constitutes as a control for considering the need of tools to support effective communication of risk and risk management within an organization.

Template*:* constitutes a control for identifying the degree of formality of templates needed for recording risk and its management effectively.

Product Status: defines a control for determining the amount of risk management needed with respect to the product's quality, business value and life expectancy.

Environment*:* constitutes a control for considering how to adopt risk management with regard to the project's cultural, social, political and physical context.

Organization*:* defines a control for considering organizational aspects such as people's attitudes towards risk, organizational maturity, competency and training and their impact on a risk management program.

Measures*:* constitutes a control for determining the need for integrating the risk management process with other organizational processes, such as measurement processes, to provide useful information and feedback to the organization.

This summarizes our integration model and its four components. It is used as a basis for outlining an integrated model described in the next section.

5.4 Outline of Integrated Model

The integrated model is illustrated in Figure 5.4. Note that we use the synthesized agile and risk management, models as the basic constituents of the integrated model.

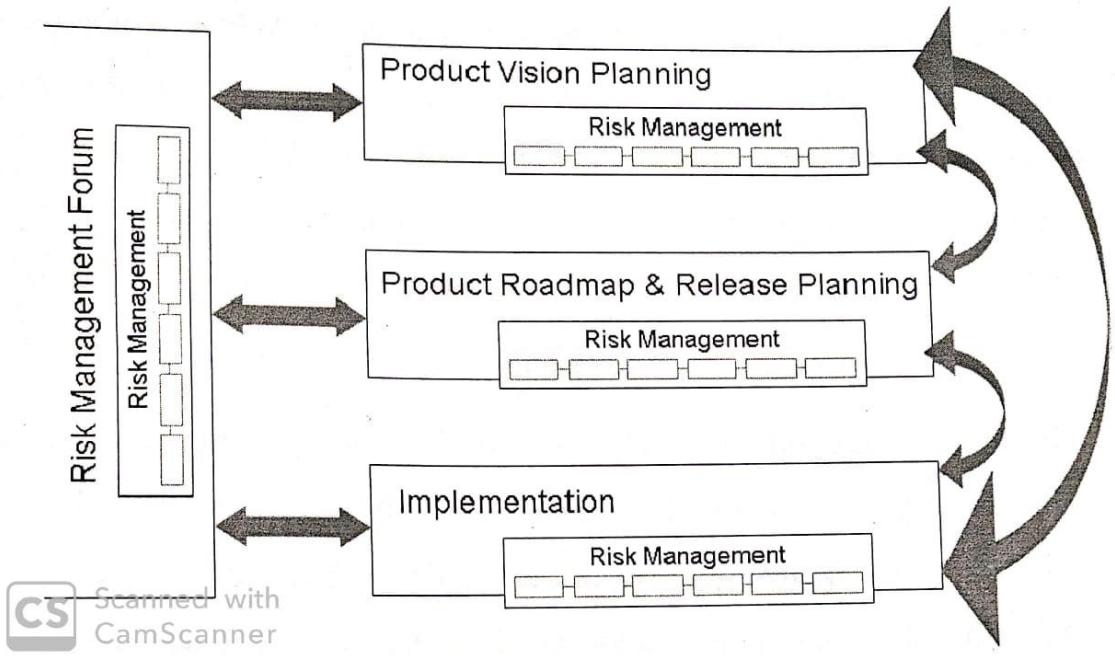


Fig. 5.4: Intergrated Model

The integrated model manages all risks that are encountered within the agile development process on an organization-wide basis. As depicted in Figure 5.4, the agile process covers three main development phases, *Product Vision Planning*, *Product Roadmap and Release Planning*, and *Implementation*.

Most of the risks undergo a complete risk management process within these development phases as mapped out in Figure 5.4. The risk management process consists of the six phases of *Risk Identification, Risk Analysis, Risk Management Planning, Risk Monitoring and Control, Risk Sign-Off* and *Risk Post-Mortem Analysis.* Some of the risks, however, may not be fully managed or mitigated within a single phase. In this case, they may have to be transferred to the next phase, and/or get reported for an organizational function called *Risk Management Forum (RMF).* This transfer is represented by the double-edged arrows between the three agile development phases and the box representing the *RMF* on the left-hand side of Figure 5.4.

The Risk Management Forum is a function for coordinating risk management across the organization. It manages any risks that cannot be managed within a certain development phase and/or may have an impact on other parts of the organization, and which promptly have to be disseminated to all the concerned parties, for instance other teams or organizational units. It consists of a cross-functional group represented by the roles responsible for or concerned with or otherwise capable of managing organizations-wide risks.

In the following, we provide a brief overview of the outline of the integrated model. It is described according to the integration model and its four integration points: (1) *Process Phases and Organizational Levels,* (2) *Roles and Responsibilities,* (3) *Communication Channels,* and (4) *Process Aspects.*

5.4.2 Process Phases And Organizational Levels

The integrated model covers the entire agile development process. This involves integration of risk management on two organizational levels: the *Business* and *Engineering*. The *Business Level* consists of the *Product Vision Planning* phase. The *Engineering Level* consists of the *Product Roadmap and Release Planning* and *Implementation* phases. The agile phases and the management of risks in each phase are briefly outlined below:-

Product Vision Planning'. The *Product Vision Planning* phase is the first phase in the synthesized agile model. It involves creating a product vision plan describing the product goals, overall business and product structure and return on investment. The resulting product vision plan guides the work carried out in subsequent planning, decision making, and development (Scrum, 2003). Risk management within this phase mainly concerns the identification and analysis of business related risks, such as budget and resource risks.

Product Roadmap and Release Planning*:* The *Product Roadmap and Release Planning* phase is the second development phase in the synthesized agile model. Here, one first creates a high-level roadmap plan for the identified product releases, which one then regularly revisits for more detailed planning before each release starts (Scrum, 2003). The risk management on the *Product Roadmap and Release Planning* phase comprises risk identification, risk analysis and risk action planning. It involves both business and technical risks.

Implementation*:* The third agile development phase is the *Implementation* phase. In this phase, the team, product management and other relevant stakeholders meet to plan the work to be conducted in the coming iteration. The plan is then executed in the

iteration to deliver an increment of working product functionality (Beck, 2004). Risk management in the *Implementation* phase primarily covers the monitoring and controlling of the risks that has been transferred in this phase from previous development phases. New risks are also continuously identified, analyzed and planned for during this phase. Risks are mainly of technical character.

5.4.3 Roles and Responsibilities

In the integrated model, we have identified several roles having various responsibilities with respect to risk management and its communication within and between the various agile process phases. Generally, however, the risks are owned by the roles in the phase where the risk is originally identified. Below, we describe the roles and their responsibilities:

RMF Members own all the organization-wide risks. Their main task is to supervise and coordinate all the organization-wide risks and make decisions on them. However, they may delegate their management to other roles either within the *Business* or *Engineering* levels or both.

Business Manager is responsible for managing risks at the *Business Level*. This role owns all the risks relevant to this level. However, he may delegate their management to the roles in the *Product Roadmap and Release Planning* phase or *Implementation* phase. The choice depends on the character of the risk and where in the organization, it is most adequately managed. However, he still keeps the risk ownership till the delegated risks get mitigated.

Product Manager is responsible for all the risks managed in the *Product Roadmap and Release Planning* phase. This role owns all the risks relevant to this level. However, similar to the business manager, he may delegate their management to other roles in the organization, if needed. He still keeps the risk ownership till the delegated risks get mitigated.

Team Leader and Team Members are responsible for managing risks within the *Implementation* phase. The team leader supervises the risk management within this phase. Usually, team members own the risks that concern the development tasks assigned to them. However, same as with the risk owners in the other phases, the team may also delegate risks to others in the organization depending on the risk and risk management needs.

5.4.5 Process Aspects

Our industrial study has helped us to identify the impact of Process Aspects of the integrated model. This impact is materialized in the following guidance for adapting risk management in the integrated process:

- Risk definition is a main prerequisite for identifying risks and deciding when to use the risk management process to control risk: It is of high relevance when making decisions on when to perform risk management and when to incorporate risk management into software development.
- Risk assessment results identify pertinent actions for performing risk driven development: The risk classification and assessment techniques are important for knowing when and how to manage risks effectively in subsequent process phases. Hence, guidelines for assessing risks are needed.

• The product's software lifecycle stage aids in designating appropriate risk management actions: The portfolio of risk types and risk management activities differs considerably depending on whether a project concerns new development or maintenance of a legacy system. In early lifecycle phases, the business risks often get a higher priority value, for instance due to time-to-market concerns, whereas technical risk may be of greater importance in the maintenance phase. This means that the process is adapted with regard to the prevailing risk types and their prioritization.

- Designation of and coverage of stakeholders, roles and responsibilities are determined by factors such as project risk profile, project type and size: The coverage of stakeholder roles and the degree of their involvement depends on the type of project, size, the risk complexity, risk scope and the risk criticality. In small, agile projects, too many roles managing minor risks can lead to too much coordination and overhead. In more complex cases, one needs to introduce more formality to make sure that that the risks get proper attention

- The use of supporting tools and repositories is determined by factors such as project risk profile, project type, project size and team distribution: The relevance of tools and repositories when adopting risk management depends on the size of the organization, the project needs and the project risk profile.

- The degree of recording formality varies with respect to project risk profile, project type, size, and team distribution: Templates provide relevant support for describing and communicating risks. However, the level of recording information in them varies with respect to project type, project size, team distribution and risk severity. In agile

projects tackling low severity risks, templates should always be kept very simple. The degree of recording formality increases as soon as the risk criticality and/or project or development organization grows.

- Product status, life expectancy and business value help determining the amount of the risk management process needed: The amount of attention put into the risk management process varies greatly depending on the product status, its life expectancy and business value. For instance, systems close to retirement do not get as much attention from the risk management perspective as systems that are developed from scratch. The reason is that the risk and their criticality generally vary with the status and the business value of the product.

- The environment and the project's physical context determine the formality of the risk management process: Systems that are developed in a non distributed environment where the team members and the customer representative (product owner) are co-located and work closely together do not generally need to coordinate risk management in a formal way. However, if the organization is large or on the way to expand, generally require a more formal and conventional risk management process as distributed environments will imply more coordination.

- Organizational maturity and training aids in adopting a risk management program successfully: Organizational maturity, such as people's attitude towards risks, competency, and capability to perform risk management are of relevance for successful adoption of risk management. For instance, training should be part of the risk management program in organizations where the risk management awareness or knowledge is low.

- Software development needs are integrated with other organizational processes: The development and risk management processes need to be integrated to provide useful feedback to the organization, especially in larger organizations. Hence, one needs guidelines for making such integration explicit.

International Journal of Advanced Multidisciplinary Scientific Research (IJAMSR) ISSN: 2581-4281
www.ijamsr.com CrossRef: https://doi.org/10.31426/ijamsr

MANAGING RISKS: CHALLENGES IN ADOPTION OF AGILE SOFTWARE DEVELOPMENT APPROACH

CHAPTER - 6

CONCLUSIONS

6.1 Conclusion

This research was based on the concept of managing risk where the researcher sought to discover the extent of risk management practices in agile SD projects. This analysis involved a decomposition of the primary elements of risk management, namely risk identification, estimation and evaluation.

An analysis of the literature revealed little evidence on the extent of the use of risk management practices in agile methods. However, it is widely recognized that agile methods themselves were introduced to combat well-known risks associated with SD project failures such as scope creep, cost overruns and schedule pressures. Their use of incremental development and active user involvement is an attempt to combat such risks.

Controlling risks, improves essential software development features such as product quality, precision planning and cost-efficiency (Englund, 1997) (Ropponen and Lyytinen, 2000). For this reason, the inclusion of risk management in software development is an important factor to consider if one wishes to achieve project success (Kontio, 1999). Unfortunately, our research shows that many software development models, both the traditional and agile ones, are not well aligned with the risk management process practices.

In this thesis, we have addressed this by outlining a model specifically aimed at aligning the agile model with risk management. This is accomplished by integrating the agile and risk management processes. The proposed solution consists of an integration model, providing guidelines for integrating the two processes and an integrated model, that is, a reference model against which organizations can compare their risk management practice.

In its current version, it is primarily targeted towards process engineers and business developers or other roles involved in process engineering and process improvement.

The results show that the model places risk management on specific agile development phases. The integrated model also suggests preliminary patterns for *where* and *when* risk management takes place in the agile process.

Also, in contrast to other existing models for including risk management in software development, such as the risk-driven spiral model (Boehm, 1988) and the approaches suggested by for instance Sliger (2006) and Li et al. (2006), our model recognizes risk management in development as an organization-wide activity. It places risk management over the entire software development lifecycle and it involves both the *Business* and *Engineering* organizational levels. The *RMF* and the *Communication Channels* add structure to the process and help coordinate and control the organization-wide management of risks.

Furthermore, it makes explicit that different roles are responsible for different types of risk as managed on different organizational levels and in different process phases. Finally, the *Process Aspects* aid in designing the instances of an integrated process with regard to the

need of risk management, which is also relevant for reasoning about agility.

The model is however still in its infancy. It needs to be further elaborated and complemented with more details. Hence, the premises presented herein must be further validated. Despite this, we believe that it already provides (1) a platform for communicating about risk management within agile development, (2) a reference model for software organizations to examine their practices and see how they compare to the integrated model, (3) a starting point for researchers and industrial organizations to start defining their own integrated development models, and (4) a foundation for future work.

Finally, regarding the aspect of agility the solution is intended to maintain a level of agility adequate in the context at hand. Essentially, agile development and risk management both aim at the same thing that is, helping development teams to do the right things in critical development situations.

Hence, they should not be in conflict with each other, With our solution, we also argue that agility can be maintained and at the same time support the goals and practices of risk management.

6.2 Scope for Further Research

While the research produced many interesting findings, there is much scope for further research as follows:

- Given the diversity with which methods are adopted across organizations, a large-scale quantitative study may identify more generalisable themes regarding the adoption and deployment of risk management practices across a large number of projects.

- A final potential area for further research may be a comparison between the existing state of risk management in agile project environments and traditional project environments. This form of research may produce interesting findings in relation to risk management maturity over time.

MANAGING RISKS: CHALLENGES IN ADOPTION OF AGILE SOFTWARE DEVELOPMENT APPROACH

REFERENCES

- A.C Edmondson, S.E. Mcmanus, Meethodological fit in management field research, Academy of Management Review 32 (4) (2007) 1155- 1179.
- Abrahamsson P. et al. (2002) Agile Software Development Methods: Review and Analysis. VTT Electronics/Oulu University, Oulu.
- Abrahamsson, P., J. Canboy, K. and Wang, X., "Lots Done, More To Do: The Current State of Agile Systems Development Research". Euporean Journal of Information Systems, 18, 281-284, 2009.
- Abrahamsson, P., J. Warsta, et al. (2003). New Directions on Agile Methods: A Comparitive Analysis. Proceedings of the 25th International Conference on Software Engineering. IEEE Computer Society, 244-254.
- Abrahamson, P., O. Salo, et al. (2002). Agile Software Development Methods: Review and Analysis, VTT Pulications.
- Abrahamsson, Pekka. Agile Software Development Methods: Review and Analysis. Julkaisua-Utgivare-Publisher. 2002.
- Addison T. and Vallabh S. (2002), Controlling Software Project Risk: An Emperical Study of Methods used by Experienced Project Managers. In proceedings of the Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on

Enablement Through Technology (SAICSIT), South Africa, pp. 128-140.

- Agile Alliance (2001), [Online] URL: http://www.agilealliance.com. Accessed April 2008.
- Agile Manifesto, Manifesto for Agile Software Development. [internet], http://agilemanifesto.org [Accessed last 01-03-2006].
- Agile Manifesto, Manifesto for Agile Software Development, January 2012;http://www.agilemanifesto.org/. Beck, K., and Andres, C., Extreme Programming Explained: Embrace Change, 2nd Edition. Boston: Addison-Wesley, 2004.
- Ahren D., Clouse, A. Turner R. (2003) CMMI Distilled. 2nd Ed., Addison-Weslry, Boston, MA.
- Ahren, D. M., Clouse, A, and Turner, R. (2003) CMMI Distilled: A Practical Introduction to integrated Process Improvement 2nd ed. OK Addison Wesley.
- Ambler, S. W. (2002). Agile Modeling: Best Practices for the Unified Process and Extreme Programming. New York, John Wiley & Sons.
- Andrew Begel and Nachiappan Nagappan, "Usage and perceptions of agile software development in an industrial context: An exploratory study," in Proceeding of the First International Symposium on Empirical Software Engineering and Measurement, Washindton, DC, USA, 2007, ESEM '07, pp. 225-264, IEEE Computer Society.
- Armenta A.L and Gaono A. (2008), A Proposal of a Process Integration Model for Agile Software Development and Risk Management. Master thesis, Department of Computer and System Sciences, Stockholm University/ Royal Institute of Technology, Stockholm, Sweden.

- Atlas, "Accidental adoption: The story of scrum at amazon.com," in Agile 2009 conference, Chicago, IL, 2009, pp. 135-140.
- Aveling, B. (2004). XP Lite Considered Harmful? Extreme Programming and Agile Processes in Software Engineering. J. Eckstein and H. Baumeister, Berlin, Germany, Springer.
- Avison, F. Lau, M. Myers, P.A. Nielsen, Action research,
- B. Boehm, Get ready for agile methods, with care, IEEE Computer 35 (1) (2002) 64-69.
- B. Boehm, R. turner, Balancing Agility and Discipline: A Guide for the Perplexed, Addison- Wesley, Boston, 2003, ISBN 9780321186126.
- B.A, Kitnchenham, Guidelines for performing Systematic Literature Reviews in Software Engineering Version 2.3.
- Keele Iniversity and University of Durham, EBSE Technical Report, 2007.
- B.A. Kitchenham, S.L. pfleeger, L.M. Pickard, P.W. Jones, D.C.
- Beck K. (2000), Extreme Programming Explained: Embrace Change. Addison-Wsley, Upper Sadle River, NJ.
- Beck K. (2000), Extreme Programming Explained: Embrace Change. 2ny Ed., Addison-Wesely, Upper Sadle River, NJ.
- Beck K. et al. (2001), Manifesto for Agile Development. [Online] URL: http:agilemanifesto.org. Accessed October 2009.
- Beck, K. (1999). Extreme Programming Explained. Reading, MA, Addison Wesley.
- Beck, K., (2000) Extreme Programming Explained: Embrace Change, Addison- Wesley Press.

- Beecham, S., Hall, T. and Rainer, A. (2003), Defining a Requirements Process Improvement Model. Software Quality Journal Volume 13 Number 13 September 2005, Springer, pp.247-279.
- Berteig M. (2005), Agile Work Uses lean Thinking: Empirical Process Control. [Online] URL: http://www.agileadvice.com/2005/11/09/ agilemanagement/ agile-work-uses-lean-thinking-empirical-process-control/.
- Accessed May 2008.
- Beynon-Davies, P. and M.D. Williams (2003). The Diffusion of Information Systems Development Methods, Journal of Strategic Information Systems, 12(1), 29-46.
- Biemer P. and Lyberg L. (2003), Introduction to Survey Quality. John Wiley & Sons, Hoboken, NJ.
- Boehm B. (1988), A Spiral Model of Software Development and Enhancement. IEEE Computer, Vol. 21 (5), pp. 61-72.
- Boehm B. (1991), Software Risk Management: Principles and Practices. IEEE Software, Vol. 8(1), pp. 32-41.
- Boehm B. and Turner R. (2004), Balancing Agility and Discipline: A guide for the Perplexed. Addison-Wesley, Boston, MA.
- Boehm B. and Turner R. (2005), Management Challenges to Implementing Agile Processes in Traditional Software Organizations. IEEE Software, Vol. 22 (5), pp. 30-39.
- Boehm B., Brown W., Turner R. (2005), Spiral Development of Software-Intensive Systems of Systems. In proceedings of the 27th International Conference on Software Engineering, St Louis, USA.

- Boehm, B. (2003), Value-Based Software Engineering ACM SIGSOFT Software Engineering Notes, 28(2) March 2003, pp 3-15.
- Boehm, B. W. (1988). A Spiral Model of Software Development and Enhancement, IEEE Computer, 21(5), 61-75.
- Boehm, B. W. (1991). Software Risk Management: Principles and Practices, IEEE Software 8(1), 32-41.
- Boehm, B., "Get Ready For Agile Methods, With Care", *IEEE Computer,* 35(1), 64-69, 2002.
- Boehm, B., and Turner, R., "Balancing Agility and Discipline: Evaluating and Integrating Agile and Plan- Driven Methods", *Proceedings of the 26th International Conference on Software Engineering*, 718-719, 2004.
- Boehm, B., and Turner, R., "Using Risk to Balance Agile and Plan-Driven Methods", *IEEE Computer, 36(6),* 57-66, 2003.
- Boehm, B., Port, D., Jain, A., and Basili, V. (2002), Achieving CMMI Level 5 Improvements with MBASE and the CeBASE Method. [Internet] Cross Talk Journal.
- Bohner S. and Coram M. (2005), The Impact of Agile Methods -6n Software Project Management. In proceedings of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems.
- Bonoma, T., (1985). Case Research in Marketing: Opportunities, Problems and a Process, Journal of Marketing Research, 22(2), 199-208.
- Bostrom B., et al. (2006), Extending XP Practices to Support Security Requirements Engineering. In proceedings of the Workshop on Software Engineering for Secure Systems,

Shanghai, China.

- Brodman, J. and Johnson, D (1997), A Software Process Improvement Approach for small organisation and small projects. Proceedings of the 19th International Conference in Software Engineering, 19th may 1997, Boston- MA, ACM Press, pp661-662.
- Brown N. (1996), Industrial-Strength Management Strategies. IEEE Software, Vol. 13 (4), pp. 94-103'.
- Byrnes D. (2006), The Case for Integrated Processes. SAP Insight. [Online] URL: http://www.alignjournal.com /whitepapers-pdf/The%20Case%20for%20Integrated%20 Processes. pdf?CFID=78553&CFTKEN=93223018. Accessed May 2008.
- C.K. Riemenschneider, B.C. Hardgrave, F.D. Davis, Explaining ,software developer acceptance of methodologies: a comparison of five theoretical models, IEEE Transactions on Software Engineering 28 (12) (2002) 1135-1145.
- Carayannis, E.G. *Agile Project Management for IT Project.* Greenwood Press / Quorum Books. 2002.
- Carr M.J. et al. (1993), Taxonomy-Based Risk Identification. SEI Technical Report CMU/SEI-93-TR-006 ESC-TR-93-183, SEI/CMU, Pittsburg, PA. [Online] URL: http://www. sei.cmu.edu/pub/documents/93.reports/pdf/tr06.93.pdf. Accessed March 2007.
- Carver J., Seaman C., Jeffery R. (2004), Using Qualitative Methods in Software Engineering. International Advanced School of Software Engineering, Los Angeles, CA.

- Casey, V. and Richardson, I. (2002), A Practical Application of Ideal Model. Product Focused Software Process Improvement, 4th International Conference (PROFES), December 9-11, Rovaniemi — Finland,' Springer, pp. 172-184.
- Chapman, C. and S. Ward (1997). Project Risk Management: Processes, Techniques and Insights, John Wiley & Sons Ltd.
- Chapman, R. J. (1998). The Effectiveness of Working Group Risk Identification and Assesment Techniques, International Journal of Project Management, 16(6), 333-343.
- Charette R. (1989), Software Engineering: Risk Analysis and Management. McGraw-Hill, New York, NY.
- Charette Rr (2001), The Decision is in: Agile vs Heavy Methodologies. Cutter Consortium, Vol. 2(19). [Online] URL: http://www.cutter.com/content/project/fulltext/updates/2001/ep muOl 19.html. Accessed October 2009.
- Charette R. and O'Brien P. (1999), Draft IEEE PI540 Risk Management Standard - Process and Implications. [Online] URL:
- http://www.psmsc.com/UG1999/Presentations/Software%20Ri k%20Management%20Standard%20Update.pdf. Accessed May 2008.

- Charette, R. N. (1989). Software Engineering Risk Analysis and Management, Multiscience Press Charette, R. N. (1990). Applications Strategies for Risk Analysis, Multiscience Press Inc.
- Cho, Juyun. *Issues and Challenges of agile software development with SCRUM.* Issues in Information System. VOL IX, No. 2. 2008.

- Chrissis, M. B., Konrad, M. and Shrun, S. (2003) CMMI: Guidelines for Process Integration and Product Improvement, UK, Addison Wesley.
- Chrissis, M. B., Wemyss, G., Goldenson, D., Konrad, M., Smith, K. and Svolou, A. (2003) CMMI Interpretive Guidance Project : Preliminary Report [Internet], Software Engineering Institute.
- Christie, A. Jyl. (1999), Simulation in support of CMM-based process improvement. Journal of Systems and Software, (46): 107-112.
- Chroust, G. and Hardt, S. (1996), Executing Process Models Activity and Project Management. In proceedings of IEEE Symposium and Workshop on Engineering of Computer Based Systems, Friedrichshafen, Germany.
- Coad, P., J. de Luca, et al. (1999). Java Modelling in Color, Englewood Cliffs, NJ, Prentice Hall.
- Cockburn A. (2002), Agile Software Development. Pearson Education, Boston, MA.
- Cockburn A. (2005), Crystal Clear: A Human-Powered Methodology for Small Teams. Pearson Education, Upper Saddle River, NJ.
- Cockburn, A. (2001). Crystal Clear: A human-powered software development methodology for small teams, Reading, MA, Addison-Wesley.
- Cockburn, A., (2004) Crystal Clear A Human-Powered Methodology for Small Teams, and it Addison- Wesley Press.
- Cockburn, Crystal Clear: A Human-Powered Methodology for Small Teams, Addison-Wesley, 2004, ISBN 0-201-69947-8.
- Cockburn, Selecting a project's methodology, IEEE Software 17 (4) (2000) 64-71.

- Cohen D. et al. (2003), Agile Software Development: a DACS State-of-the- Art Report. Fraunhofer Center for Experimental Software Engineering/ University of Maryland.
- Cohen, M. Lindvall, P. Costa, An introduction to agile methods, in: M.V. Zelkowitz (Ed.), Advances in Computers, Advances in Software Engineering, vol. 62, Elsevier, Amsterdam, 2004.
- Cohen, S. *A Software System Development Life Cycle Model for Improved Stakeholders' Communication and Collaboration.* Int. J. of Computers, Communications & Control. Vol. V, No. 1, pp. 20-4. 2010.
- Cohn M. (2006), Agile Estimating and Planning. Pearson Education, Upper Saddle River, NJ.
- Cohn, M., and Ford, D., "Introducing an Agile Process to an Organization", *IEEE Computer, 36(6),* 74-78, 2003.
- Collis, J. and R. Hussey (2003). Business Research, Palgrave Macmillian.
- Communications of the ACM 42 (1) (1999) 94-97.
- Coppendale, J. (1995). Manage Risk in Product and Process Development and Avoid Unpleasant Surprises, Engineering Management Journal, 5(1), 35-38.
- Cule P. et al. (2000), Strategies for Leading off IS Project Failure. Information Systems Management, Spring, pp. 65-73.
- Dangle, K., Larsen, P. and Shaw, M. (2005) software process improvement in small organisation: A case study, IEEE Software November/ December 22(6) pp 68-75.
- DeMarco T. (2004), Risk Management for Software Projects. The Atlantic Systems Guild, Paper 130-29, Camden, ME.
- DeMarco, T? and T. Lister (2003). Risk Management during Requirements, IEEE Software, 20(5), 99-101.

- DeMarco, T. and T. Lister (2003). Waltzing with Bears: Managing Risk on Software Projects, Dorset House Publishing Co.
- DSDM Consortium Manual Version 4.2 (2002-2006). Risk Management.
- DSDM Consortium White Paper (1999-2003). Risk Management.
- Dutta, S., Lee, M. and Wassenhove, L. K. (1999) Software Engineering in Europe : A study of best practices, IEEE Software Volume (16), Issue(3).
- Dyba, T. (2003), Factoes of Software Process Improvement Success in Small and Large Organizations: An Empirical Study in the Scandinavian Context. Proceedings of the 9th European Software Engineering Conference held jointly with 10th ACM ACM SIGSOFT International symposium on foundations of software Engineering, September 2003, Helsinki-Finland, ACM Press, pp. 148-157.
- Dyba, T., and Dings0yr, T., "Empirical Studies of Agile Software Development: a Systematic Review", *Information and Software Technology,* 50 (9-10), 833-859, 2008.
- Dyba, Tore. *Empirical studies of agile software development: A systematic review.* 24 January 2008.
- Eclipse **Process** Framework (EPF) (2007), OpenUP Process. [Online]
  URL:
  http://www.eclipse.org/epf/general/OpenUP.pdf. Accessed April 2008.
- Eisenhardt, K., "Building Theories from Case Study Research", *The Academy of Management Review, 14(4),* 532-550, 1989.

- Englund H. (1997), A Case Study to Explore Risk Management Models. Master Thesis, Royal Institute of Technology, Stockholm, Sweden.
- Erickson, J., Lyytinen, K., and Siau, K., "Agile Modeling, Agile Software Development, and Extreme Programming: the State of Research", *Journal of Database Management,* 16(4), 88-100,2005.
- Fitzgerald, B. (2000). Systems Development Methodologies: The problem of tenses. Information Technology and People, 13(3), 174-185.
- Fitzgerald, B., "An Empirical Investigation into the Adoption of Systems Development Methodologies", *Information and Management,* 34, 317-328, 1998.
- Fitzgerald, B., "Formalized Systems Development Methodologies: a Critical Perspective", *Information Systems Journal,* 6,3-23, 1996.
- Fitzgerald, B., "Systems Development Methodologies: the Problem of Tenses", *Information Technology and People,* 13, 13-22,2000.
- Fitzgerald, B., Hartnett, G. and Conboy, K., "Customising Agile Methods to Software Practices at Intel Shannon", *European Journal of Information Systems,* 15, 200-213, 2006.
- Fitzgerald, B., Russo, N. L., and O'Kane, T., "Software Development Method Tailoring at Motorola", *Communications of the ACM,* 46(4), 64-70, 2003.
- Frame, J. D. (2002). The New Project Management: Tools for an Age of Rapid Change, Complexity, and Other Business Realities, Jossey-Bass; A Wiley Company.

- Fuerer, R. and K. Chaharbaghi (1996). Competitive Environments, Dynamic Strategy Development Capabilities and Business Performance. Benchmarking for Quality Management and Technology, 3(3), 32-49.
- G. Benefield, "Rolling out agile in a large enterprise," in 41st Hawaii International Conference on System Science. 2008, IEEE Computer Society.
- G. Cloke, "Get your agile freak on! agile adoption at yahoo! music," in Proceedings of the AGILE 2007, Washington, DC, USA, 2007, pp. 240-248, IEEE Computer Society.
- G. Keefer, Extreme Programming Considered Harmful for Reliable Software Development 2.0, AVOCA GmbH, Online Report, 2003.
- G.W. Noblit, R.D. Hare, Meta-Ethnography: Synthesizing Qualitative Studies, Sage Publications, London, 1988.
- GatherSpaceTeam, "Agile software development," Retrieved January 15,2011, January 2011.
- Glasser E.G. and Strauss, A.L. (1967), The Discovery of Grounded Theory: Strategies for Qualitative Research. Aldine Publishing, Chicago, IL.
- Glib, T. (2003), Software Project Management Adding Stakeholder Metrics to Agile Projects. The European Journal for the Informatics Professional, IV(4) August 2003, pp.5-9.
- Greene, "Agile methods applied to embedded firmware development," in Proceedings of the Agile Development
- Conference, Washington, DC, USA, 2004, pp. 71-77, IEEE Computer Society.
- Grey, S. (1995). Practical Risk Assessment for Project Management, John Wiley & Sons Ltd.

- Gunasekaran, Agile manufacturing: A framework for research and development, International Journal of Production Economics 62 (1- 2) (1999) 87-105.
- H. Erdogmus, M. Morisio, M. Torchiano, On the effectiveness of the test-first approach to programming, IEEE Transactions on Software Engineering 31 (3) (2005) 226-237.
- H. Takeuchi, I. Nonaka, The new product development game, Harvard Business Review (January) (1986) 137-146.
- Hall, E. M. (1998). Managing Risk: Methods for Software Systems Development, Addison Wesley.
- Henssen, Xjeir K. *Maintenance and Agile Development Challenges, Opportunities and Future Directions.* 2009.
- Herbsleb, J. D. and Goldenson, D. R. (1996), A Systematic Survey of CMM experience and results. Proceedings of the 18th international conference on Software Engineering, May 1996, Berlin, Germany, IEEE Computer Society, pp.323-330.
- Hevner S., March, Park J. and Ram S. (2004), Design Science Research in Information Systems. *Management Information Systems Quarterly*, Vol. 28 (1), pp. 75-105.
- Highsmith J. (2002), Agile Software Development Ecosystems. Addison- Wesley, Boston, MA.
- Highsmith, J., (2004) Agile Project Management, Creating innovative products, Addison- Wesley.
- Highsmith, J., and Cockburn, A., "Agile Software Development: The business of Innovation", *IEEE Computer, 34(9)*, 120-127,2001.
- Hillson, D. (2001). Extending the Risk Process to Manage Opportunities, Fourth European Project -Management Conference, PMI Europe 2001, London, UK.

- Hoaglin, K. El Emam, J. Rosenberg, Preliminary guidelines for empirical research in software engineering, IEEE Transactions on Software Engineering 28 (8) (2002) 721-734.
- Hofstede, G. (2003), Culture's Consequences, Comparing Values, Behaviors, Institutions and Organizations Across Nations. 2nd Ed., Sage Publications, Newbury Park, CA.
- Hulett D.T. (2001), Key Characteristics of a Mature Risk Management Process. In proceedings of the European Project Management Conference/PMI Europe, UK.
- IBM: *Rational Unified Process: Best practices for software development teams* (2003), http://www.ibm.com/developerworks/rational/library/253.html
- IEEE 12207 (1998), IEEE Std 12207.0-1996 Software Life Cycle Processes. The Institute of Electrical and Electronics Engineers Inc., New York, NY.
- IEEE 1540 (2001), IEEE 1540 Standard for Lifecycle Processes-Risk Management. The Institute of Electrical and Electronics Engineers Inc., New York, NY.
- IEEE830 (1993), IEEE Std 830-1993 Guide to Software Requirements Specifications. The Institute of Electrical and Electronics Engineers Inc, New York, NY.
- Ihme, T. and Abrahamsson, P., '(2005) The Use of Architectural Patterns in the Agile Software Development of Mobile Applications, International Journal of Agile Manufacturing, Vol. 8, issue 2, 97-112.
- Institute of Risk Management, Association of Insurance and Risk Managers and National Forum for Risk Management in the Public Sector (IRM) (2002), A Risk Management Standard. ;UK. [Online] URL:

http://www.tlteirm.org/publications/PUstandard.html.
Accessed February 2007.

- ISO/IEC 73 (2002), ISO/IEC 73:2002 Risk Management Vocabulary -Guidelines for Use in Standards. International Organization for Standardization, Geneva, Switzerland.
- J. Cohen, A coefficient of agreement for nominal scales, Educational and Psychological Measurement 20 (1960) 37-46.
- J. Erickson, K. Lyytinen, K. Siau, Agile Modeling, Agile software development, and extreme programming: the state of research, Journal of Database Management 16 (4) (2005) 88-100.
- J. Mcavoy, T. Butler, The impact of the Abilene Paradox on doubleloop learning in an agile team, Information and Software Technology 49 (6) (2007) 552-563.
- J. Stapleton, DSDM: Business Focused Development, second ed., Pearson Education, 2003, ISBN 978-0321112248.
- J. Tabaka, "11 ways agile adoption fails," StickyMinds.com Column, stickyMinds.com, retrieved March 2011.
- J.P.T. Higgins, S. Green (Eds.), Cochrane Handbook for Systematic Reviews of Interventions, Version 5.0.0 (updated February 2008), The Cochrane Collaboration, 2008. Available from: <www.cochrane-handbook.org>.
- J.R. Landis, G.G. Koch, The measurement of observer agreement for categorical data, Biometrics 33 (1) (1977) 159-174.
- Jayakanth Srinivasan and Kristina Lundqvist, "Using agile methods in software product development: A case study," Information Technology: New Generations, Third International Conference on, vol. 0, pp. 1415-1420, 2009.

- John Hunt, Agile Software Construction, Springer, September 2005.
- Johnson, J., Boucher, K. D., Connors, K. and Robinson, J. (2001), Collaborating on Project Success. [Internet], Software Magazine.
- K. Beck, Extreme Programming Explained: Embrace Chage, second ed., Addison-Wesley, 2004, ISBN 978-0321278654.
- K. Beck, Extreme Programming Explained: Embrace Change, Addison- Wesley, 2000, ISBN 0-201-61641-6.
- K. Conboy, B. Fitzgerald, Toward a conceptual framework of agile methods: a study of agility in different disciplines, in: Proceedings of XP/Agile Universe, Springer Verlag, 2004.
- K. Schwaber, M. Beedle, Agile Software Development with Scrum, Prentice Hall, Upper Saddle River, 2001.
- K. Scotland and A. Boutin, "Integrating scrum with the process framework at yahoo leurope," in Agile 2008 conference, Toronto, ON, 2008, pp. 191 - 195.
- K.S. Khan, G. Ter Riet, J. Glanville, AJ. Sowden, J. Kleijnen, Undertaking Systematic Review of Research on Effectiveness, CRD's Guidance for those Carrying Out or Commissioning Reviews, CRD Report Number 4, second ed., NHS Centre for Reviews ahdJDissemination, University of York, 2001.
- Karlstrom D. (2003), Effects of Introducing Agile Methodologies in Software System Engineering. In proceedings of the 3rd National Conference on Software Engineering Research and Practice in Sweden, Sweden.
- Keil, M., P. E. Cule, et al. (1998). A Framework for Identifying Software Project Risks, Communications of the ACM, 41(11), 76-83.

- • Kendall, K. E., and Kendall, J. E., "Observing Organizational Environments: A Systematic Approach for Information Analysts", *MIS Quarterly*, 5(1), 43-55, 1981.
- Kendall, K. E., and Kendall, J. E., "STROBE: A Structured Approach to Observation of the Decision- Making Environment", *Information and Management*, 7(1), 1-11, 1984a.
- Kendall, K. E., and Kendall, J. E., "Structured Observation of the Decision-Making Environment: A Validity and Reliability Assessment", *Decision Sciences*, 75(1), 107-118, 1984b.
- Kendall, K. E., and Kendall, J. E., Systems Analysis and Design, eighth edition, Prentice Hall, 2011.
- Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick.
- Kieran Conboy, Sharon Coyle, Xiaofeng Wang, and Minna Pikkarainen^ "People over process: Key challenges in agile development," IEEE Software, vol. 28, pp. 48-57, 2011.
- Kontio J. (1999), Risk Management in Software Development: a Technology Overview and the Riskit Method. In proceedings of the 21st International Conference on Software Engineering, LA, USA.
- Kruchten P. (2000), The Rational Unified Process: An Introduction. 2nd Edition. Addison-Wesley, Reading, MA.
- Kuhn T. (1970), The Nature and Necessity of Scientific Revolutions. 2nd Edition. University of Chicago Press.

- L. Williams, A. Cockburn, Agile Software development: it's about feedback and change, IEEE Computer 36 (6) (2003) 39-43.
- L.M Sanchez, R Nagi, A review of agile manufacturing systems, International Journal of Production Research 39 (16) (2001) 3561-3600.
- Land R. (2006), Software Systems In- House Integration: Observations and Guidelines Concerning Architecture and Process. Ph D Thesis, Department of Computer Science and Electronics, Malardalen University, September.
- Larman, V.R. Basili, Iterative and incremental development: a brief histore, IEEE Computer 36 (6) (2003) 47-56.
- Lee, A. S., "A Scientific Mehtodology for MIS Case Studies", MIS Quarterly, 13(1), 33-50, 1989.
- Leffingwell D. (2007), Scaling Software Agility: Best practices for Large Enterprises. Pearson Education, Boston, MA.
- Li M. et al. (2006), A Risk, Driven Method for eXtreme Programming Release Planning. Lab for Internet Software Technologies, Institute of Software, Chinese Academy of Sciences, pp. 423-430.
- Li, E., Chen, H. and Lee., T. (2002), Software Process Improvement of Top Companies in Taiwan:a comparative study. Total Quality Management 13(5) March 2002, pp701-703.
- Little D. (1995), Objectivity, Truth, and Method. A Philosopher's Perspective on the Social Sciences - Commerntary. Anthropology Newsletter.
- Longino H. (1990), Science as Social Knowledge: Values and Objectivity in Scientific Inquiry. Princeton University Press.

- Lyard, A. and Orci, T. (2000), Dynamic CMM for Small organizations. Proceedings ASSE 2000, the first Argentine Symposium on Software Engineering, Seotember 2000, Tandil- Argentina, pp. 133-149.
- M. Aoyama, Web-based agile Software development, IEEE Software 15 (6) (1998) 56-65
- M. Cohn, Succeeding with Agile Software development using scrum, Addison-Wesley, 2009..
- M. Ho st, P. Runeson, Chechlists for software engineering case study research, in: Proceedings of the First International Symposium on Empirical Software Engineering and Measurement, IEEE, Madrid, Spain, 2007, pp.479-481.
- M. Poppendieck, T. Poppendieck, Lean Software Development - An Agile Toolkit for Software Development Managers, Addison-Wesley, Boston, 2003, ISBN 0-321-15078-3.
- M. Stephens, D. Rosenberg, Extreme Programming Refactored: The Case Against XP, Apress, Berkeley, CA, 2003, ISBN 1-59059-096-1.
- M.B. Miles, M. Huberman, Qualitative Data Analysis: An Expanded Sourcebook, second ed., Sage Publications, 1994, ISBN 0803955405.
- M.C. Paulk, C.V. Weber, B. Curtis, M.B. Chrissis, The Capability Maturity Model: Guidelines for Improving the Software Process, Addison-Wesley, Boston, 1995, ISBN 0-201-54664-7.
- Mantel, S. J., J. R. Meredith, et al. (2001). Project Management in Practice, John Wiley & Sons, Inc.
- Marchewka, J. T. (2003). Information Technology Project Management: Providing Measurable Organisational Value, John Wiley & Sons, Inc.

- Mark Kennaley, SDLC 3.0: Beyond a Tacit Understanding of Agile, Fourth Medium Press,2010.
- Martinf: R.C., *Agile Software Development, Principles, Patterns and Practice.* Prentice Hall. 2002.
- Maxwell J. (1992), Understanding and Validity in Qualitative research. Harvard Educational Review, Vol. 62 (3), pp. 279-300.
- McManus, J. (2004). Risk Management in Software Development Projects, Elsevier Butterworth- Heinemann.
- Merriam-Webster (2008), Merriam-Webster Online. [Online] URL: http://www.merriam-webster.com. Accessed April 2008.
- Mifflin Company, Boston, 2002.
- Mikael Lindvall, Dirk Muthig, Aldo Dagnino, Christina Wallin, Michael Stupperich, David Kiefer, John May, and Tuomo K?hk?nen, "Agile software development in large organizations," Computer, vol. 37, pp. 26-34, 2004.
- Mike Cohn and Doris Ford, "Introducing an agile process to an organization," Computer, vol. 36, pp. 74-78, 2003. International Journal of Managing Value and Supply Chains (IJMVSC) Vol. 2, No. 3, September 2011.
- Moynihan T. (1997), How Experienced Project Managers Assess Risk. IEEE Software, Vol. 14 (3), pp. 35-41.
- N. Britten, R. Campbell, C. Pope, J. Donovan, M. Morgan, R. Pill, Using meta ethnography to synthesise qualitative research: a worked example, Journal of Health Services Research and Policy 7 (4) (2002) 209-215.
- Nawrpcki, J., Walter, B. and Wojciechowski, A. (2001), Towards the maturity model for extreme programming, 27th Euromicro Proceedings 4-6 September 2001, pp.233-239.

- Nerur, S., and Balijepally, V., "Theoretical Reflections on Agile Development Methodologies", *Communications of the ACM, 50(3),* 79-83, 2007.
- Nerur, S., Mahapatra, R. K., and Mangalaraj, G., "Challenges of Migrating to Agile Methodologies", *Communications of the ACM, 48(5* 72-78, 2005.
- Nikiforova, O., Nikulsins, V., Sukovskis, U.: *Integration of MDA Framework into the Model of Traditional Softwar Development.* In: Frontiers in Artificial Intelligence and Applications, Databases and Information Systems V, vol. 187, pp. 229-239. IOS Press, Amsterdam (2009).
- North American and European **Enterprise** Software and Services Survey, Business Technographics Ed., 2005.
- P. Abrahamsson, J. Warsta, M.T. Siponen, J. Ronkainen, New directions on agile methods: a comparative analysis, in: Proceedings of the 25th International Conference on Software Engineering (ICSE'03), IEEE Press, 2003.
- P. Abrahamsson, O. Salo, J. Ronkainen, J. Warsta, Agile software development methods: review and analysis, VTT Technical report, 2002.
- P. Checkland, J. Scholes, Soft Systems Methodology in Action, Wiley, Chichester, 1990, ISBN 0-471-98605-4.
- P. Kjutchen, The Rational Unified Process: An Introduction, third efl., Addison-Wesley, Boston, 2003.
- P. Mcbreen, Questioning Extreme Programming, Pearson Education, Boston, MA, USA, 2003, ISBN 0-201-84457-5.
- P. Meso, R. Jain, Agile software development: adaptive systems principles and best practices, Information Systems Management 23 (3) (2006) 19-30.

- Padayachee K. (2002), An Interpretive Study of Software Risk Management Perspectives. In proceedings of the Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on Enablement Through Technology (SAICSIT), South Africa.
- Palmer S. and Felsing J. (2002), A Practical Guide to Feature-Driven Development. Prentice Hall, Upper Saddle River, NJ.
- Paulk, M. C. (2001) Extreme Programming from a CMM Perspective IEEE Software 18(6) November/December 2001, pp 19-26
- Paulk, M. C., Weber, C. V., Curtis, B. and Chrissis, M. B. (1995) The Capability Maturity Model for Software : Guidelines for Improving the Software Process (SEI). USA, Addison Wesley.
- Paulk, M., Curtis, M., and Weber, C.,(1993) Software Process Maturity Questionnaire : Capability Model version 1.1. [internet], Carnegie Mellon-Software Engineering Institute, Available from <http:/Avww.sei.cmu.edu/publications/documents/93 .reports/9 3.tr.02 4.html> [Accessed 11-01-2007]
- Paulk, M., Weber, C. and Curtis, M.(1999), The Capability Maturity Model for Software. In K. Emam & N. Madhavji (Eds.), Elements of software process Assessment and Improvement. IEEE Computer Society Press, PP.3-22.
- Paulk, M.C. (1998), Using the Software CMM in Small Organisations. The Joint 1998 Proceedings of the Pacific Northwest Software Quality Conference and the Eighth International Conference on Software Quality, 13-14 October 1998, Portland, USA, Software Engineering Institute, PP.350-

361.

- Persse, J. R. (2001) Implementing the Capability Maturity Model, USA, John Wiley & Sons Inc.
- Peterson, Kai. *A Comparison of Issues and Advantages in Agile and Incremental Development between State of the* Art *and an Industrial Case.* Journal of System and Software. 2009.
- Pikkarainen, M. and Mantyniemi, A., (2006) An Approach for Using CMMI in Agile Software Development Assessments: Experiences from Three Case Studies, SPICE 2006 conference, Luxemburg.
- Poppendieck M. and T. (2003), Lean Software Development: An Agile Toolkit. Pearson Education, Upper Saddle River, NJ Poppendieck, M. (2001). Lean Programming, Software Development Magazine 9(5), 71-75.
- Powell, "P. L. and J. H. Klein (1996). Risk Management for Information Systems Development, Journal of Information Technology, 11(4), 309-319.
- Project Management Institute (PMI) (2004), A Guide to the Project Management Body of Knowledge (PMBoK), 3rd Ed. ANSI/PMI 99-001-2004, Project Management Institute, Newton Square, PA.
- R.L. Ackoff, Alternative types of planning, in: Ackoffs Best: His Classic Writings on Management, Wiley, New York, 1999, pp. 104-114.
- Raynus, J. (1999) Software process improvement with CMM, London, Artech House.
- Raz, T., A. J. Shenhar, et al. (2002). Risk Management, Project Success and Technological Uncertainty, R&D Management, 32(2), 101-109.

- Reifer, D. (2002). How good are agile methods? IEEE Software, 19(4), 16-18
- Remenyi, D. (1999). Stop IT Project Failures through Risk Management, Butterworth-Heinemann.
- Rico, David F. *What is the ROI of agile vs. traditional Methods.* 2008.
- Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas, "Manifesto for agile software development," 2001.
- Robson : C. (2002), Real World Research. Blackwell Publishing, Alden.
- Ropponen J., and Lyytinen K. (2000), Components of Software Development Risk: How to Address Them? A Project Manager Survey. IEEE Transactions on Software Engineering, Vol. 26 (2), pp. 98-112.
- Rowe, W. D. (1977). An Anatomy of Risk, John Wiley & Sons Ltd.
- Russo, N. and Wynekoop, J., "The Use and Adaptation of System Development Methodologies", *Proceedings of the Information Resources Management Association International Conference*, 162, 1995.
- S. Nerur, R. Mahapatra, G. Mangalaraj, Challenges of migrating to agile methodologies, Communications of the ACM (May) (2005) 72-78.
- S. Nerur, V. Balijepally, Theoretical reflections on agile development methodologies, Communications of the ACM 50 (3) (2007) 79-83.
- S.R. Palmer, J.M. Felsing, A Practical Guide to Feature-driven Development, Prentice Hall, Upper Saddle River, NJ, 2002, ISBN 0-13-067615-2.

- Schwaber K. (2004), Agile Project Management with Scrum, Microsoft Press, Redmond, WA.
- Schwaber K. (2007), The Enterprise and Scrum. Microsoft Press", Redmond, WA.
- Schwaber K. and Beedle M. (2001), Agile Software Development with Scrum. Prentice Hall.
- Schwaber, K. and Beedle, M., Agile Software Development with SCRUM, Upper Saddle River, Prentice- Hall, 2002.
- Schwaber, K. and M. Beedle (2002). Agile Software Development with Scrum, Upper Saddle River, NJ, Prentice-Hall.
- Schwaber, K., Agile Project Management with SCRUM, Microsoft Press, 2004. Schwaber, S. and Beedle, M., (2002) Agile Software Development With Scrum, Prentice Hall.
- Schwalbe, K. (2000). Information Technology Project Management, Course Technology, Thomson Learning.
- Scrum (2003), Scrum Methodology: Incremental, Iterative Software Development from Agile Processes. Rev. 0.9. Advanced Development Methods Inc.
- SEI (2008), Software Engineering Institute/Carnegie Mellon University, "Risk Management Overview". [Online] URL: http://www.sei.cmu.edu/ risk/. Accessed April 2008.
- • Sj0berg, J.E. Hannay, O. Hansen, V.B. Kampenes, A. Karahasanovic, N.-K. Liborg, A.C. Rekdal, A survey of controlled experiments in software engineering, IEEE
- Transactions on Software Engineering 31 (9) (2005) 733-753.
- Sj0berg, T. Dyba°, M. J0rgensen, The Future of Empirical Methods in Software Engineering Research, in: Future of Software Engineering (FOSE'07), IEEE, 2007, pp. 358-378.

- Sliger M. (2006), Relating PMBoK Practices to Agile Practices (Part 3 of 4). [Online] URL:
- http://www.stickyminds.com/sitewide.asp?Function=: edetail&
- ObjectType= COL &ObjectId=11133&commex=l#4993. Accessed February 2008.
- Smith, P. and R. Pichler (2005). Agile Risks/Agile Rewards. Software Development, 13(4), 50-53.
- Smith, P. G. and G. M. Merritt (2002). Proactive Risk Management: Controlling Uncertainty in Product Development, Productivity Press.
- Sommerville I. (2006), Software Engineering, 7th Ed., Addison-Wesley, Reading, MA.
- Standards Australia and New Zealand (2004), "Australian/New Zealand Standard Risk Management AS/NZS 4360:2004". 3rd Ed. Standards Australia/ New Zealand, Sydney/Wellington.
- Standish Group (2001), Extreme Chaos. The Standish Group International Inc.
- Stapleton J. (1997), DSDM Dynamic Systems Development Method: the Method in Practice. Addison-Wesley.
- Stapleton, J. (1997). DSDM: Dynamic Systems Development Method, Harlow, England, Addison Wesley.
- Strauss, J. Corbin, Basics of Qualitative Research, second ed., Sage Publications, 1998, ISBN 0-8039-5939-7.
- Systems *Development Lifecycle: Objectives and Requirements.* Bender RPT Inc. 2003.
- Szalvay, Victor. *An Introduction to Agile Software Development.* Danube Technologies Inc. 2004.

- T. Dyba° , E. Arisholm, D. Sj0berg, J. Hannay, F. Shull, Are two heads better than one? On the effectiveness of pair-programming, IEEE Software 24 (6) (2007) 10-13.
- T. Dyba° , T. Dings0yr, G.K. Hanssen, Applying systematic reviews to diverse study types: an experience report, in: Proceedings of the 1st International Symposium on Empirical Software Engineering and Measurement (ESEM'07), IEEE Computer Society, Madrid, Spain, 2007, pp. 225-234.
- T. Dyba°, B. Kitchenham, M. J0rgensen, Evidence-based software engineering for practitioners, IEEE Software 22 (1) (2005)58-65.
- T. Dyba°, Improvisation in small software organizations, IEEE Software 17 (5) (2000) 82-87.
- T. Gilb, Competitive Engineering: A Handbook for Systems Engineering, Requirements Engineering, and Software, Elsevier Butterworth- Heinemann, Oxford, 2005, ISBN 0-7506-6507-6.
- T. Ohno, Toyota Production System: Beyond Large-scale Production, Productivity Press, New York, USA, 1988, ISBN 0-91^299-14-3.
- Thomsett, R. (1993). Third Wave Project Management: A Handbook for Managing the Complex Information Systems for the 1990s, Prentice-Hall Inc.
- Tim, K. (2004) Practical insight into the CMMI, USA Artech house.
- Tore Dyb°a and Torgeir Dings0yr, "Empirical studies of agile software development: A systematic review," Information and Software Technology, vol. 50, pp. 833-859, August 2008.

- Trochim W. (2006), Research Methods Knowledge Base. [Online] URL: http://www.socialresearchmethods.net/kb. Accessed April 2008.
- Turk, R. France, B. Rumpe, Assumptions underlying agile software development processes, Journal of Database Management 16 (4) (2005) 62-87.
- Turner, R. and Jain, A. (2002) Agile meets CMMI: Culture clash or common cause XP/Agile universe 2002, LNCS 2418 pp. 60-69.
- V. Rajlich, Changing the paradigm of software engineering, Communications of the ACM 49 (8) (2006) 67-70.
- Vijayasarathy, L. and Turk, D., "Drivers of Agile Software Development Use: Dialectic Interplay Between Benefits and Hindrances", *Information and Software Technology, 54* (2), **137-AS, 2012.**
- Vijayasarathy, Leo R. *Agile Software Development: A survey of early adopters.* Journal of Information Technology Management Volume XIX, Number 2. 2008.
- W.R. Shadish, T.D. Cook, D.T. Campbell, Experimental and Quasi- Experimental Designs for Generalized Causal Inference, Houghton.
- • W.S. Humphrey, PSP: A Self-Improvement Process for Software Engineers, Addison-Wesley, 2005, ISBN 978-0321305497.
- Walker R. (1985), Applied Qualitative Research, Gower Publishing Company Ltd.
- Walters, S.A., J.E. Broady et al. (1994). A Review of Information Systems Development Methodologies. Library Management, 15(6), 5-19.

- Wenger, Communities of Practice: Learning, Meaning and Identity, Cambridge University Press, Cambridge, UK, 1998, ISBN 0-521-43017-8.
- Wiegers K. (1998), Know Your Enemy: Software Risk Management. Software Development Magazine, October.
- Wiegers, K. E. (1998). Know Your Enemy: Software Risk
- Management, Software Development, 6(10), 38-42.
- Williams R. et al. (1999), Software Risk Evaluation (SRE) Method Description (Version 2.0). Technical Report CMU/SEI-99-TR-029, SEI/CMU, Pittsburg, PA.
- Williams, L. and A. Cockburn (2003). Agile Software Development: It's about Feedback and Change, IEEE Computer Society, 36(6), 39-43.
- Williams, L., and Cockburn, A., "Agile Software Development: It's about Feedback and Change", *IEEE Computer, 36(6*\ 39-43, 2003.
- Williams, R. C., J. A. Walker, et al. (1997). Putting Risk Management into Practice, IEEE Software, 14(3), 75-81.
- Willis, G. B., Cognitive interviewing: A tool for improving questionnaire design. Thousand Oaks, CA: Sage Publications, 2004.
- Wohlin C., Host M, Henningsson K. (2003), Empirical Research Methods in Software Engineering. Empirical Methods and Studies in Software Engineering. Lecture Notes in Computer Science, Vol. 2765, Springer-Verlag, Heidelberg.
- Wysocki R. K., McGary R., *Effective Project Management*, Third Edition, John Wiley & Sons © 2003.
- XP (2001), Extreme Programming: A Gentle Introduction. [Online] URL: http://www.extremeprogramming.org,

Accessed in April 2008.

- Xu R.Z. et al. (2005), Optimizing Software Process Based On Risk Assessment and Control. In proceedings of the 5th
- International Conference on Computer and Information
- Technology, China.

- Y. Dittrich, M. John, J. Singer, B. Tessem, For the special issue on qualitative software engineering research, Information and Software Technology 49 (6) (2007) 531-539.
- Yin R.K. (2003), Case Studsy Research: Design and
- Methods.3rd Edition, Sage Publications, Thousand Oaks, CA.

- Yin, R. K., Applications of case study research. Thousand Oaks, CA: Sage Publications, 2003a.
- Yin, R. K., Case study research: Design and methods. Thousand Oaks, CA: Sage Publications, 2003b.
- Zachman J. (1987), A Framework for Information System Architecture. IBM Systems Journal, Vol. 26 (3). IBM Publication G321-5298. URL: http://www.research.inm.com /journal/50th/applications/zacheman.html.
- Zaharan, S. (1998) Software Process Improvement: Practical Guidelines for Business Success, 1st .USA, Addision- Wesley.