Software Quality Journal manuscript No.
(will be inserted by the editor)

Prioritizing Agile Benefits and Limitations in Relation to Practice Usage

Adam Solinski · Kai Petersen

Received: date / Accepted: date

Abstract Context. In recent years there has been significant shift from rigid development (RD) towards agile. However, it has also been spotted that agile methodologies are hardly ever followed in their pure form. Hybrid processes as combinations of RD and agile practices emerge. In addition, agile adoption has been reported to result in both: benefits and limitations.

Objectives. This exploratory study (a) identifies development models based on RD and agile practice usage by practitioners; (b) identifies agile practice adoption scenarios based on eliciting practice usage over time; (c) prioritizes agile benefits and limitations in relation to (a) and (b).

Methods. Practitioners provided answers through a questionnaire. The development models are determined using hierarchical cluster analysis. The use of practices over time is captured through an interactive board with practices and time indication sliders. This study uses the extended hierarchical voting analysis framework (EHV-F) to investigate benefit and limitation prioritization.

Results. Four types of development models and six adoption scenarios have been identified. Overall, 45 practitioners participated in the prioritization study. A common benefit among all models and adoption patterns is knowledge and learning, while high requirements on professional skills were perceived as the main limitation. Furthermore, significant variances in terms of benefits and limitations have been observed between models and adoption patterns.

Adam Solinski
Misys,
Gdynia, Poland
E-mail: adam.solinski@misys.com

Kai Petersen (corresponding author)
Blekinge Institute of Technology
Karlskrona, Sweden
E-mail: kai.petersen@bth.se

Conclusions. The most significant internal benefit categories from adopting agile are knowledge and learning, employee satisfaction, social skill development, and feedback and confidence. Professional skills specific demands, scalability and lack of suitability for specific product domains are the main limitations of agile practice usage. Having a balanced agile process allows to achieve a high number of benefits. With respect to adoption, a big-bang transition from RD to agile leads to poor quality in comparison to the alternatives.

Keywords Agile · Benefits · Limitations · Practice Adoption · Prioritization

1 Introduction

Software development is a complex process, which includes aspects such as planning, requirements engineering, architecture, implementation, testing and reviews as well as inspections [28]. In order to handle the complexity and increase control over the success factors (scope of a project, budget, deadlines [42] and quality [10]), it is suggested to use a structured approach that would organize the development process [4,9,27,42]. Two development models are commonly distinguished: RD and agile [5,22,27,33].

In recent years, there has been observed a significant shift from traditional, RD development towards agile approaches, which is mostly caused by the market dynamics and constantly changing customer needs [22,26,33,44]. Researchers claim that combining the two approaches might result in recognizable advantages [5]. On the other hand, agile adoption is considered as a challenging process [35,25,22]. Also, hybrid approaches are emerging [11,19, 21,33,47]. Agile adoption results in both: Benefits and limitations [28]. However, evidence is limited of how and to what extent agile is adopted in industry and what the actual effects of adoption are [15,28], especially when time and evolution of agile practices use is taken into account. Dybå and Dingsøyr [15] explicitly emphasizes the lack of longitudinal studies when investigating agile, considering the time dimension.

This article presents an empirical investigation of software development organizations that claim to have adopted agile. The development processes are captured in terms of the applied RD and agile practices. The effects of agile adoption are measured through the extent to which practitioners perceive benefits and limitations. Contributions to knowledge about agile software development are:

1. Identification of development models by clustering and identifying patterns of current practice use. A development model in the context of this study is referred to as a similar set of practices used by a group of companies. These are established by cluster analysis to find groups of companies applying similar practices.
2. Identification of adoption scenarios by capturing practice usage over time. An adoption scenario is describing how practices are added and removed over time. For instance, a scenario could be a big-bang change in practices in comparison to a gradual addition of practices over time.

3. An investigation of agile benefits and limitations is followed by creating a ranking considering the relative significance of the benefits and limitations as perceived by practitioners in relation to 1. and 2. To capture the information an extended hierarchical voting analysis framework [24] is used. Hierarchical voting is a common approach in research studies to capture relative priorities [39,24,37].

The remainder of the paper is organized as follows: Section 2 presents related work. Section 3 explains the research method used, followed by the results in Section 4. Section 5 discusses the results, and provides the main implications for research and practice. Section 6 concludes the paper.

2 Related Work

2.1 Agile and Rigid Development

The literature shows a distinction between two approaches (methodologies types) towards software development: traditional RD and agile [5,22,27,33]. Both approaches are compared in [28].

Agile approaches focus on minimization of waste (e.g. reduction of the number of requirements that are elicited, documented and verified but are not eventually implemented [33]) and increased communication with the customer. Responding to changes is more important than strict compliance with a project plan [18]. Agile teams are self-organizing, mindful of business goals and involved in the life of a project on a daily basis. The most popular and recognized lightweight methodologies are Scrum and eXtreme Programming (XP) [12,33,41,44]. For each of the methodologies, there has been reported a number of both benefits and limitations [15,31,33].

There is no universal methodology nor particular approach that would be versatile enough to fit in any project type [27]. There is no "silver bullet" [6, 8]. No methodology can offer a set of principles and practices that could be adapted to be suitable for any purpose. Each methodology has a few strengths and weaknesses. There are some grounds (e.g. size of development teams, software criticality, the dynamism of requirements change) where approaches and corresponding methodologies dominate one another [6,7,28]. Back in 1992, a certain concern was stated: *"How can a system which is based on freezing requirements work in times of uncertainty?"* [8]. Over the past twenty years, the statement still is and will remain current. What has been observed recently is that companies are inclined to switch to more flexible, adjustable approaches.

There are a number of companies that have decided to alter their development approach by shifting from RD to agile. Agile is considered as more flexible and adaptive when customer needs change [22,26,33,44]. However, the adoption or transformation process is challenging. The changes affect not only the development process itself. Communication paths, user involvement, requirements and change management as well as current management style, people

and processes must be taken into consideration [22]. Still, RD approaches provide better predictability, repeatability and optimization opportunities [5]. A lot of attention is given to architecture design, which is very important in case of large-scale software and systems [34]. RD approaches are also more preferable in producing secure software [45] and providing high assurance [5].

Have regard to the above, it is undeniable that software development companies tend to combine RD and agile approaches. However, instead of full, expensive methodology replacement, existing processes are modified. Companies adopt new and replace currently used practices. Researchers claim that combining both approaches might result in recognizable benefits [5]. Research has shown that combined approaches are possible and, moreover, beneficial [5,19,33]. Companies follow hybrid approaches such as waterfall and XP [19], Scrum and XP [12].

The extent to which agile and RD practices are used is not clear. Little information can be found that would reveal patterns for combining actual practices of both approaches. Moreover, there is no clear evidence how adoption of agile practices actually affects the development processes and meets the adopters' expectations over time. There are likewise organizations that have a lack of structured approaches to software development [14,36] or, on the contrary, follow a completely RD methodology.

2.2 Literature on practices

For this prioritization study the practices to be surveyed needed to be identified. A number of studies have served as sources for identifying agile practices. Five studies were having an explicit focus on identifying and surveying agile practices [23,13,1,33]. The number of practices identified range from 14 [1] to 59 [13] .

A recent study on agile [23] elicited which agile practices were used by software development practitioners. The study consolidated a list of agile practices from a number of additional studies identifying and listing agile practices (cf. [15] and [48]). Given that Kurapati et al. [23] presents a manageable set of agile practices the study served as input for the survey used in this study. The studies described in the related work of Kurapati et al. [23] were reviewed again in order to verify the completeness of the practices set, and were complemented in case an important practice was missing. It is also important to mention that one practice used in the survey subsumes several other practices (e.g. *face-to-face communication* subsumes the practices *team sits together*, *open office space*, and *video conferencing needed in case of distributed teams*). An overview of the practices and which other practices they subsume is given in Appendix A.

2.3 Agile benefits and limitations

Agile benefits and limitations have been extensively investigated in previous years. There are a number of studies in which researchers attempted to identify particular factors or simply reported positive and negative experiences from agile adoption processes. Studies focusing on capturing and reflecting on agile benefits and limitations are used to derive the lists of benefits and limitations to be surveyed in this study. Ten studies in total are used, each study being shortly summarized.

Dybå and Dingsøyr [15] conducted a comprehensive literature review on empirical studies of agile software development. The study investigated the known benefits and limitations. However, most of the studies under investigation were related to XP.

Petersen and Wohlin [31,33] attempted to refer benefits and limitations from literature to an industrial case. The research was based on 33 interviews carried out in a large-scale company. The researchers tried to detect potential problems that need to be addressed in large scale development when adopting agile.

Petersen [29] compared two software development paradigms - lean and agile development. The author, apart from comparing goals, principles, practices and processes of both paradigms, mentions a list of advantages and disadvantages.

Begel and Nagappan [1] conducted an exploratory study on agile development. The research was confined to development teams at Microsoft. The researchers tried to ask the participants what the top 3 benefits and limitations of agile were. This part of the survey was free-response. The answers were then consolidated and a list of common themes related to benefits and limitations was created. A ranking of commonly reported benefits and limitations was formed afterwards.

Ramesh et al. [46] conducted a survey of early adopters on agile development. They reported a list of agile benefits and challenges as perceived by practitioners. They collected the data through structured questions about factors. Moreover, respondents had opportunity to leave complement comments. The majority of the respondents, 82%, were taken from the United States of America. The study revealed a ranking of benefits and limitations, however, the method used was not explained.

Pikkarainen et al. [35] attempted to identify strengths and barriers of agile adoption in three software companies. The companies were initially RD organizations adopting agile methods. The strengths and barriers were collected from development related people during interviews and workshops.

Laanti et al. [25] conducted a survey based research with a purpose of agile benefits recognition during large-scale transformation within Nokia. Quantitative as well as qualitative data was collected from the opinions of practitioners. In the quantitative part, the respondents were requested to indicate on a sliding scale whether they agree or disagree with a benefit-related statement. Afterwards, the participants were asked to express in a free-response section

opinions on benefits and challenges of agile methods. A set of benefits and limitations was identified. Many of the findings are comparable with [15] and [31].

Previous studies have focused on investigating which practices do practitioners use, and which benefits and limitations have been noted. However, previous research has not focused on identifying the relative degree to which benefits and limitations are experienced. Furthermore, the benefits and limitations have not been related to combinations of practice usage and taking the history of practice use into account. This paper focuses on both, relative priorities as well as their relation to practice usage.

3 Research Method

3.1 Research Questions

Four research questions are answered in this research study:

- RQ1: Which practice combinations are currently used by practitioners?
- RQ2: Which strategies were followed in adopting agile practices over time?
- RQ3: Which are the most significant/insignificant benefits of agile practice usage?
- RQ4: Which are the most significant/insignificant limitations of agile practice usage?

3.2 Subjects

Research studies using a similar research methodology as this study, but with different research questions, have obtained 18 answers [39] and 24 answers [24], using convenience sampling. The same sampling strategy (relying mainly on personal contacts in Poland and Sweden) in combination with the use of agile forums and communities (Twitter, Meetup, LinkedIn, agile communities, Yahoo and Google groups) has been applied in this study. This makes it challenging to gather answers only through internet forums and mailing lists. In total 63 respondents started the survey and 39 respondents completed the benefits and limitations part. Table 1 gives a complete overview of the number of responses obtained. The majority of answers have been obtained through personal contacts (see Table 2), given that the prioritization study is time intensive as many comparisons have to be made, and alternatives have to be weighted against each other.

3.3 Questionnaire Design and Construction

The data collection instrument is a self-developed questionnaire provided online, including four core parts.

Part 1 Warm-up and context information: It is important to establish context elements to understand to which degree results are transferable between similar contexts [32], the following context information has been obtained:

- Roles and experience in years.
- Application type the organization is developing, using the empirically created taxonomy proposed by [17].
- Size of the organization.
- Size of the development team.
- Organization name (optional), allowing to identify the number of unique organizations captured.
- The experience with the development process for which the practitioner selects the practices and conducts the prioritization.

Part 2 Practices (RQ1 and RQ2): To capture practice usage (agile as well as RD) over time the practitioners could select which practices are used in the process, and when they started/stopped using the practice using interactive sliders. They also had the option to select *"do not know"* or *"did not use"*). The design provided an insight on the details of practices adoption and revealed the exact order of applying and abandoning particular practices. The practices asked for are provided in Appendix A.

Part 3 Benefits and limitations priorities (RQ3 and RQ4): A suitable technique for showing the importance of particular elements is prioritization [2]. Two approaches allow to prioritize showing the relative importance, namely Analytical Hierarchy Process (AHP) and Cumulative Voting (CV). AHP is only suitable for a very small number of items to be prioritized, hence hierarchical cumulative voting was chosen for this study [3]. Cumulative voting means that a number of points (e.g. 100 points) are distributed between a fixed number of items according to relative importance. Practitioners prioritized external benefits (outcome variables relevant to the customer), internal benefits (positive attributes of the agile process used), and limitations (negative attributes of the agile process used). For benefits and limitations, given the high number of items, the items were classified into categories. Practitioners prioritize the categories, and the items within the categories (i.e. prioritization in hierarchies [3]). Appendices B and C list the items that have been prioritized. The list of benefits and limitations (Part 3 of the survey) has been obtained based on the identified literature that focused on identifying benefits and limitations. We concentrated on different studies, such as case studies, secondary studies on agile aggregating a larger number of empirical agile studies, as well as surveys made in industry. Given that systematic reviews were already conducted no systematic review approach has been used in this study. From the identified relevant papers we utilized open and axial coding for the identification of chunks of statements that are related to agile benefits and limitations. For instance, all statements related to high testing effort for continuous testing were grouped. That way we achieved formulations for each individual benefit and limitation that was listed. Thereafter, relationships between them were determined. For instance, several types of effort were mentioned related to

testing, hence these were classified in one category effort. In a similar way all groups and their related benefits were created. Bear in mind that, even with an exhaustive search, it is considered not possible to identify all relevant studies (cf. [49]), there is a risk that benefits and limitations were missed. To reduce this threat, the survey was piloted and reviewed by experts.

Part 4 Contact: At the end the participants were asked (optionally) to leave a contact email address and had the opportunity to comment on the questionnaire.

The initial version of the questionnaire was tested before the final version was accessible to practitioners. The pilot study involved two experts of agile methods from Software Engineering Research Laboratory at Blekinge Institute of Technology, who separately reviewed the questionnaire. The changes were minor, including rearrangement of questions, adding more alternatives and free-response fields, language review, questions style unification, making some questions optional and changing the taxonomy that classified developed applications by type [17]. No additional factors were added. When the questionnaire was made public, three practitioners suggested a few minor changes which were introduced.

The questionnaire was available in two language versions - in Polish and English. In each step, there was a validation mechanism checking whether the participant answered all mandatory questions completely.

3.4 Data collection

The data was collected using the questionnaire described in Section 3.3. As mentioned earlier, the sampling strategy was convenience sampling relying on personnel contacts as well as on-line communities (Twitter, Meetup, LinkedIn, agile communities, Yahoo and Google groups). The respondents answered the on-line questionnaire with the guidance and explanations given in the questionnaire. The questionnaire used during the data collection can be found on the web [1].

3.5 Study Execution

During 8 weeks (start date June 2012) 63 responses were collected, out of which 45 were complete. Each participant had to spend about one hour to complete the questionnaire. Hence, each complete response added a high value to the research. However, some of the respondents decided to quit after completing the 2) practices part or skipped the practices part and proceeded to the following 3) benefits and limitations part. Therefore, the number of complete answers for both parts differs. For the 2) practices part it is 40 and for the 3) benefits and limitations part it is 39. Not all of the respondents who wished

[1] `http://www.bth.se/tek/aps/kps.nsf/attachments/agilebenlim_pdf/$file/agilebenlim.pdf`

to take part in the study decided to complete the entire questionnaire. On the other hand, even partially complete responses were included in the analysis to enhance the value of the research where possible. Table 1 shows the number of complete answers for each part of the questionnaire. The numbers in Table 1 related to Part 2 and Part 3 indicate the number of valid answers. Valid means that the respondent did not skip any question nor provided answers not making sense, e.g. by marking all the practices with the *"do not know"* answer.

Table 1 Number of responses

Condition	Responses
Participants that started answering	63
Part 1: Warm-up and context	45
Part 2: Practices part	40
Part 3: Benefits and Limitations	39
Participants that completed Part 2 and Part 3	34

Table 2 illustrates how many responses come from particular countries for Parts 2 and 3. Furthermore, the number of responses from personal contacts and communities is shown.

Table 2 Spread of answers with respect to countries for practices and benefits/limitations

Country	Responses practices	Responses benefits/limitations
Poland	21	21
Sweden	4	3
USA	4	3
India	3	2
UK	2	4
Bolivia	1	1
France	1	1
Holland	1	1
Iran	1	1
Italy	1	0
Ukraine	1	1
Finland	0	1
Total	**40**	**39**
Personal contacts	30	27
Communities	10	12

3.6 Analysis

In order to answer our research questions, it was necessary to illustrate how practices are combined to development models and what the practices adoption strategies are followed.

Identifying development models: The diversity of practice use was high. Hence, similar models need to be identified to conduct an analysis. For this purpose, a suitable solution was hierarchical cluster analysis. However, the distance measure algorithm was a matter of concern as the data was not continuous, but represented with 0 and 1 values. In this situation, Euclidean measures are not applicable and other measures, such as Russell/Rao Index, Jaccard coefficient, matching coefficient or Dice's coefficient had to be considered [16]. Clustering of binary data is very similar to the application of many of these techniques [16]. Hence, it was sufficient to choose any of them. Moreover, a posteriori examination of different clustering outcomes and graphical analysis of the processes defined by practitioners also revealed that clusters are more reasonable with this measure. The method chosen for cluster extraction was Ward's clustering algorithm as it was claimed to perform well at recovering clusters [16].

Identifying adoption scenarios: To identify adoption scenarios it was necessary to examine the scenarios in a graphical way. Six adoption scenarios have been identified by looking at: (a) What was the start situation (e.g. more agile or RD, pure or hybrid)?; (b) What was the end-situation?; (c) How did the organizations move between the two states?.

Analyzing the priorities: The data collected on benefits and limitations were structured as follows: 1) the level of categories (groups of benefits and limitations) and 2) level of factors (single benefits and limitations within the categories). There was a need to structure the data analysis approach and apply a method that would make it possible to show the relation among factors within groups as well as from a global perspective (all factors). Kuzniarz and Angelis developed a framework, EHV-F [24] to analyze hierarchical cumulative voting data. The validity of the framework has been checked in [24].

4 Results

4.1 Research Context

In the development process stakeholders of diverse interests and needs are involved [40]. Most often, responses come from programmers (44.44%), process experts (17.22%) and project managers (13.33%). The most experienced group of respondents are process experts (16 years). In general, the respondents are experienced practitioners. Only few answers have been provided by persons in the roles quality assurance and business analyst.

With regard to software types, most of the answers (80%) fall into the category of data-dominant software (e.g. web browsers, implementation tools, applications for viewing information, on-line booking etc.). The group that had no responses is computation-dominant software (e.g. information processing). Only few answers fall into the categories of control-dominant software (13.33%), such as embedded or real-time software), or software systems (6.67%), including operating systems, support utilities and middleware.

Organizational culture differs (e.g. in terms of flexibility, absence of bureaucracy, rigidity in decision-making and many more) with respect to its size [20]. In the survey companies in the range of less than 50 employees up to 4500 and more employees participated. Responses were evenly distributed among organizational sizes. The most frequently reported development team size was "less than eleven" (around 76% of responses). Still, there have been answers that revealed teams of more than 10, 20 or 40 members.

Answers from 22 unique companies have been received. Five organizations contribute to around 30% of the answers, while 17 unique organizations contribute to around 34% of the answers. Overall, this indicates that the results are not biased to an individual company.

The respondents were requested to answer how often had they followed the current development process that they attempted to define. About 40% of the respondents declared a thorough knowledge of the process, 44% answered that they had followed the process in many projects and 15% had followed the process at least in a single project.

4.2 RQ1: Which practice combinations are currently used by practitioners?

Cluster analysis showed the distance between individual answers. Groups of similar develop models could be identified, and each group contained a number of models reported by the respondents (see Figures 1 and 2). The remaining answers were very distant to each other and could not be grouped due to a lack of similarity, as identified by the algorithm. In the analysis the remaining answers are treated as an own category "Others" in Sections 4.4 and 4.5.

Members of Cluster16 follow an agile process with few RD practices, claiming to use all of the agile practices with a few exceptions. There is some disagreement in applying the following practices: 1) On-site customer (applied in 40% cases), 2) Pair-programming (50% cases), 3) Test-driven development (70% cases). However, the remaining practices are agreed on at least in 90% of cases. Members of Cluster16 strongly agree on applying the repeatable development process practice (80%). There are less noticeable signs of applying other RD practices: 1) Up-front documentation of requirements, 2) Extensive time planning, 3) Detailed management plans and documentation.

Members of Cluster13 apply many agile practices (from 64% to 71% of the entire set), and overall represent a balanced hybrid process. Pair-programming and Test-driven development are not applied at all. The group is characterized with frequent application of Up-front documentation of requirements (100%) and detailed up-front architecture design (75%). The remaining RD practices are also applied by the members. The development process is a balanced hybrid and mixture of agile and RD practices, where the average ratio of RD practices and agile is 2/3 respectively.

The process described by members of Cluster10 is mainly agile. All the practices applied by these organizations are considered belonging to agile development approaches based on literature.

Practice	Answer ID									
	23	33	26	28	8	50	47	14	16	19
# Up-front documentation of requirements	x				x	x	x			
# Detailed up-front architecture design										
# Big bang integration and infrequent releases										
# Extensive time planning	x	x								x
# The project follows a sequential flow of phases										
# Detailed mngm.plans and process documentation				x						
# Repeatable development process	x	x	x	x	x	x	x		x	
Time boxing	x	x	x	x	x	x	x	x	x	x
Continuous integration with testing	x	x	x	x	x	x	x	x	x	x
Short iterations and releases	x	x	x	x	x	x	x	x	x	x
Iteration planning meeting	x	x	x	x	x	x	x	x	x	x
Iteration reviews, retrospectives	x	x	x	x	x	x	x	x	x	
Test-driven development	x	x	x	x	x	x		x		
Collective code ownership	x	x	x	x	x		x	x	x	x
Face-to-face communication	x	x	x	x	x	x	x	x	x	x
Technical excellence	x	x	x	x	x	x	x	x	x	x
Small self-organizing cross-functional teams	x	x	x	x	x	x	x	x	x	x
On-site customer	x	x	x	x						
Frequent planning/reporting	x	x	x	x	x	x	x	x	x	x
Prioritized list of requirements	x	x	x	x	x	x	x	x		x
Pair-programming			x	x	x	x		x		
	Cluster1		Cluster2		Cluster3		Cluster5		Cluster12	
	Cluster4				Cluster6					
	Cluster7									
	Cluster16									

Fig. 1 Clusters of Development Models

The process described by members of Cluster11 is dominated by RD practices with only few agile elements. The members claim to apply 6 out of 7 RD practices (detailed up-front architecture design is not applied).

4.3 RQ2: Which strategies were followed in adopting agile practices over time?

The respondents were asked to mark on a time-line the points in time when each agile or RD practice was adopted by their organization. This allows to investigate how the practices combinations are developed. The order of practices and strategies of adoption can be observed (e.g. big-bang adoption, incremental). Also organizations being different and not comparable to others in the set were assigned to the category "Others".

The data was analyzed in a graphical way. For each of the responses, a graph with practices and their layout with respect to time was drawn. Out of all responses, 18 cases were found as relevant to investigate agile adoption strategies. Four groups and six scenarios have been distinguished. The criteria for creating the groups were as follows: the initial process (e.g. RD or agile or

Practice	Answer ID							
	18	21	9	39	5	42	1	40
# Up-front documentation of requirements	x	x	x	x			x	x
# Detailed up-front architecture design		x	x	x				
# Big bang integration and infrequent releases			x				x	x
# Extensive time planning			x				x	x
# The project follows a sequential flow of phases		x		x			x	x
# Detailed mngm plans and process documentation							x	x
# Repeatable development process	x			x			x	x
Time boxing	x	x	x					
Continuous integration with testing			x	x	x	x	x	x
Short iterations and releases			x	x	x	x		
Iteration planning meeting	x	x	x	x	x	x		
Iteration reviews, retrospectives	x	x	x	x	x	x	x	
Test-driven development					x	x		
Collective code ownership	x	x	x	x	x		x	x
Face-to-face communication	x	x		x	x	x		x
Technical excellence	x	x	x		x			x
Small self-organizing cross-functional teams	x	x	x	x	x	x	x	x
On-site customer	x	x			x	x		
Frequent planning/reporting	x	x	x	x		x	x	
Prioritized list of requirements	x	x	x	x	x	x		
Pair-programming						x		
	Cluster8		Cluster9		Cluster10		Cluster11	
	Cluster13							

Fig. 2 Clusters of Development Models (cont.)

both), the change of process over time (e.g. if some practices are adopted, are any other abandoned - transition). A visual representation of the identified scenarios is shown in Figure 3.

Group 1 contains organizations that significantly transform from RD to agile (represented by 4 organizations). Two scenarios were observed, incremental and big-bang. In the case of the incremental and big-bang scenario, it was important to have a repeatable development process before and after the transition. Detailed management plans and documentation centric processes were not abandoned, and were emphasized also after the transition.

Group 2 represents organizations that enrich their RD process with agile practices (represented by 3 organizations). The initial process only has a few practices implemented that are emphasized in agile, such as collective code ownership and face-to-face communication.

Group 3 represents organizations that have initially a very complex development process and shape the process over time (top-down style), represented by 2 organizations. Here, the initial process is very complex in terms of practices used. That is, approximately 85% of all the practices surveyed were implemented in the processes of these organizations.

Group 4 recently created organizations building their processes over time (bottom-up style) using two different scenarios, represented by 9 organiza-

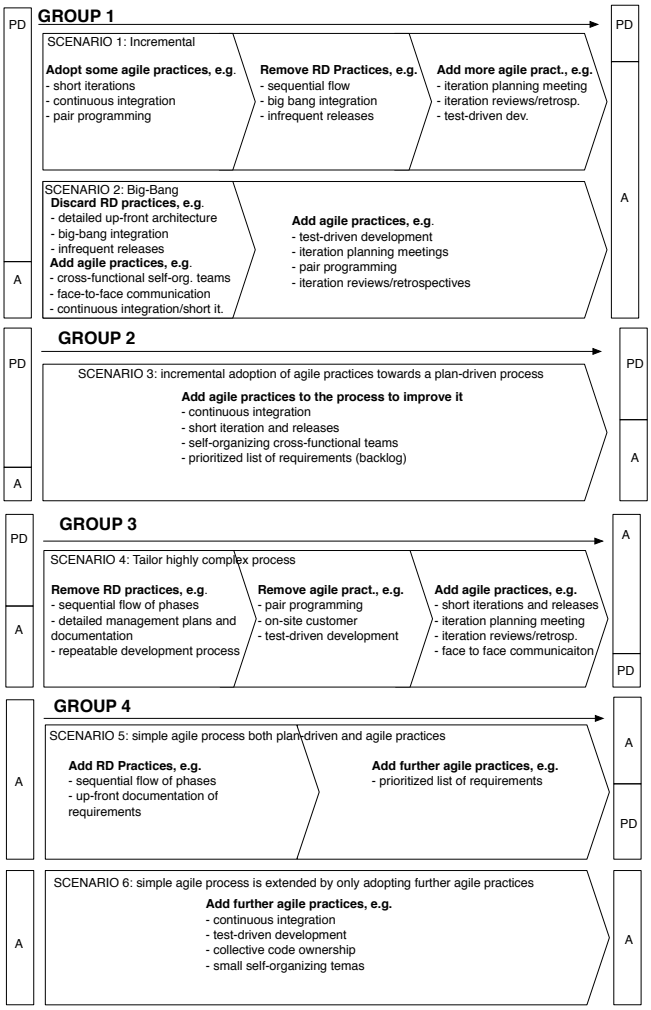

Fig. 3 Transformation Scenarios

tions. In the first scenario, the organizations already have continuous integration, iteration planning meeting, iteration reviews, and on-site customer implemented. In the second scenario, the process starts with having time-boxing, short iterations and releases, and iteration planning meeting.

4.4 RQ3: Which are the most significant/insignificant benefits of agile practice usage?

Two types of benefits have been prioritized, namely external and internal benefits. External benefits, which reflect the outcome of the development ac-

tivity in terms of higher value delivered (the customer receives software that he desires, with no understatements, the goals are met etc. [1,46]), quality (fewer defects, and increased reliability, usability etc. [25,1]), time (faster time to delivery [15]), cost (development and maintenance costs of software [15, 1,31]), and the relationship built with the customer (improved relationships, increased confidence and satisfaction level due to frequent demos, releases etc. [15,1,33]). Internal benefits are related to the development organization, such as knowledge and learning, employee satisfaction, etc. Each benefit category is broken down into additional benefits, e.g. knowledge and learning contains communication, cooperation, and adaptability skills (see Appendix B). The analysis is conducted for all answers, for different combinations of practices currently used (based on RQ1) and scenarios of practice adoption over time (based on RQ2).

4.4.1 External Benefits

Figure 4 shows the ratings of external agile benefits in the form of a heat-map. The answers are split into all respondents, respondents grouped by their use of practices, and their adoption strategy. Answers that could not be grouped due to a lack of similarity are considered in the category "Others". The darker the color (in this case green) the higher the rating by the respondents. Five items have been prioritized relative to each other, i.e. the average importance is 20 (100 divided by the number of items). For one item to be above average (i.e. is considered important in relation to other items), another value needs to be below (i.e. is thought unimportant).

	All	Groups of practices used					Adoption strategy						
		Cluster 16 (Agile/RD)	**Cluster 13** (Balanced)	**Cluster 10** (Agile)	**Cluster 11** (RD/agile)	**Others**	**Group 1 scenario 1:** Great inc. RD to Agile	**Group 1 scen. 2:** Big Bang RD to Agile	**Group 2:** Inc. adop. agile prac. to a RD process	**Group 3:** Tailoring a complex hybrid process	**Group 4 scenario 5:** Initial simple agile process by adopting both agile and RD prac.	**Group 4 scenario 6:** Initial simple agile process by adopting only agile	**Others**
Value	25,51	25,56	23,33	40,00	20,00	25,56	25,00	22,50	15,00	32,50	31,43	25,00	24,76
Quality	20,38	21,11	25,00	15,00	10,00	24,44	15,00	25,00	26,67	17,50	19,29	40,00	25,24
Time	13,58	10,00	13,33	20,00	27,50	19,44	12,50	22,50	18,33	2,50	12,86	2,50	18,57
Cost	16,28	17,22	11,67	5,00	30,00	15,83	17,50	17,50	20,00	15,00	14,29	2,50	16,90
Rel. Cust.	24,23	26,11	26,67	20,00	12,50	14,72	30,00	12,50	20,00	32,50	22,14	30,00	14,52

Fig. 4 Prioritization of External Benefits

In the following, we highlight interesting observations for all respondents, groups of practices used, and adoption strategies. For all answers, it can be observed that no benefit clearly stands out, while insignificant benefits are *Cost* and *Time*.

Concerning groups of practices used the prioritization of agile with few RD practices and the balanced process are similar. The data shows that the companies achieving time and cost benefits combine their practices as in Cluster 10, while companies emphasizing value and quality would choose one of the remaining clusters. The first three adoption strategies in Figure 4 (Gruops 1

and 2) are relatively balanced, while Groups 3 and 4 (scenario 6) have very low values for time or cost improvements.

4.4.2 Internal Benefits

Figure 5 shows the prioritization of the internal benefit categories. The ten categories described in Figure 5 were prioritized. Furthermore, the prioritization within the categories is shown in Figure 6. Looking at both figures combined, we highlight the interesting patterns in the data.

	All	Groups of practices used					Adoption strategy						
		Cluster 16 (Agile/RD)	**Cluster 13** (Balanced)	**Cluster 10** (Agile)	**Cluster 11** (RD/agile)	**Others**	**Group 1 scenario 1:** Great inc. RD to Agile	**Group 1 scen. 2:** Big Bang RD to Agile	**Group 2:** Inc. adop. agile prac. to a RD process	**Group 3:** Tailoring a complex hybrid process	**Group 4 scenario 5:** Initial simple agile process by adopting both agile and RD prac.	**Group 4 scenario 6:** Initial simple agile process by adopting only agile prac.	**Others**
1. Knowledge and learning	12,62	10,89	13,00	15,00	16,00	11,28	12,50	10,00	21,33	15,00	8,43	12,50	12,90
2. Social skill development	10,69	9,22	13,25	10,00	7,50	8,39	7,50	10,00	5,67	7,50	7,57	7,50	13,90
3. Employee satisfaction	11,41	11,00	11,75	20,00	5,00	11,17	7,50	10,00	6,00	11,50	10,29	7,50	10,71
4. Feedback and confidence	10,74	10,44	12,50	15,00	5,00	11,17	15,00	2,50	9,00	9,50	11,86	12,50	13,24
5. Quality	9,18	10,89	8,00	10,00	2,50	9,67	15,00	5,00	7,33	14,00	11,43	10,00	7,81
6. Verification and validation	10,49	11,44	9,00	7,50	30,00	10,83	10,00	20,00	21,67	9,00	8,86	17,50	10,10
7. Planning and estimations	8,82	8,11	7,50	5,00	7,50	8,44	2,50	12,50	5,67	12,50	11,14	7,50	8,05
8. Monitoring and contorlling	9,95	10,78	12,00	5,00	5,00	11,06	7,50	15,00	6,00	7,50	11,00	10,00	8,05
9. Adaptability	8,69	9,44	10,00	5,00	16,00	8,89	17,50	5,00	13,00	3,50	8,86	7,50	8,71
10. Decreased effort	7,41	7,78	3,00	7,50	5,50	9,11	5,00	10,00	4,33	10,00	10,57	7,50	6,52

Fig. 5 Prioritization of Internal Benefit Categories

In Figures 5 and 6 it is visible that with all answers combined the results are balanced, with only few items below average importance.

Comparing the groups of practices used, it is evident that Clusters 16 and 13 are hybrid processes having similar patterns, while clusters 10 and 11 emphasize specific benefits much stronger (see Figures 5 and 6). The agile process stands out in improving *confidence and satisfaction*, as well as *people feeling purposeful*. The rigid development process with few agile practices is particularly strong in *improving turn-around for bug fixing*, and *utilizing testers more efficiently* (see Figure 6), the same being true for utilizing an incremental adoption strategy. The least *cost efficient* combination of practices is to balance agile and rigid development practice, while balancing led to the highest values for *monitoring and controlling*.

Benefits in *verification and validation* are highlighted in three out of the six identified adoption strategies (Figure 5). *Knowledge and learning* is most strongly facilitated in the incremental adoption of agile practices (see Figures 5 and 6).

		All	Groups of practices used					Adoption strategy						
			Cluster 16 (Agile/RD)	**Cluster 13** (Balanced)	**Cluster 10** (Agile)	**Cluster 11** (RD/agile)	**Others**	**Group 1 scenario 1:** Great inc. RD to Agile	**Group 1 scen. 2:** Big Bang RD to Agile	**Group 2:** Inc. adop. agile prac. to a RD process	**Group 3:** Tailoring a complex hybrid process	**Group 4 scenario 5:** Initial simple agile process by adopting both agile and RD prac.	**Group 4 scenario 6:** Initial simple agile process by adopting only agile prac.	**Others**
1.1	Improved communication	4,44	3,42	4,70	5,57	4,18	4,07	3,89	3,51	9,06	3,55	2,23	4,62	4,73
1.2	Common understanding	3,98	3,04	4,66	5,57	7,32	3,68	5,05	2,41	6,45	6,10	3,15	3,59	3,79
2.1	Develop communication skills	3,38	2,73	5,49	3,02	3,53	3,44	2,11	2,19	2,54	3,65	1,80	0,96	4,46
2.2	Develop cooperation skills	3,81	3,10	4,28	4,24	5,29	4,08	1,87	4,37	3,90	2,16	3,24	3,42	4,31
2.3	Adaptability skills	3,39	2,74	4,33	3,34	0,00	3,98	4,18	2,32	0,18	1,30	2,81	2,92	4,32
3.1	Increased motivation and empowerment	3,62	3,99	4,96	4,24	0,61	3,18	3,31	3,55	1,98	4,93	2,54	3,74	4,10
3.2	People are comfortable and relaxed	2,78	2,95	4,03	5,21	2,44	1,70	1,40	2,86	2,41	0,50	2,71	1,79	3,29
3.3	Inrease in confidence and satisfaction	3,68	3,67	3,66	7,82	1,22	2,85	2,42	1,90	1,60	4,68	3,89	2,77	4,19
3.4	People feel purposeful	4,52	3,88	3,48	8,80	1,83	4,18	3,06	3,95	2,00	4,68	4,26	1,44	5,44
4.1	Early verification of design decisions	3,35	2,86	3,48	9,33	1,83	3,20	5,17	0,79	3,28	3,18	2,12	3,92	3,80
4.2	Increase awarneess of what has to be developed	3,01	2,70	4,88	2,12	0,91	2,59	5,89	0,81	2,67	1,54	4,88	3,58	2,46
4.3	Early requirements validation	3,99	3,56	4,86	3,63	1,83	4,18	4,74	0,79	3,34	4,38	4,64	4,42	4,02
5.1	Code reuse and higher quality	3,26	2,94	2,88	1,04	0,78	3,49	5,43	1,59	2,61	6,48	4,04	3,54	2,71
5.2	Improved quality of design, architecture, and performance	2,36	3,05	1,76	0,45	1,18	2,38	5,43	1,59	2,87	2,55	3,34	2,87	1,67
6.1	Improved turn-around time for fixing bugs	3,97	4,02	2,25	3,02	17,14	3,07	3,16	4,79	12,07	2,41	4,05	7,77	2,57
6.2	Quicker and more thorough response to defects from dev.	3,00	3,60	2,53	3,60	2,74	3,02	3,16	7,35	2,29	2,49	2,80	5,08	2,58
6.3	More efficient use of time of testers	3,40	3,50	3,75	0,92	12,80	3,20	4,21	5,61	9,27	3,59	2,42	4,08	2,52
7.1	Small and precise scope estimations	2,36	1,80	1,64	1,59	2,74	2,65	0,48	0,48	1,83	4,20	4,35	1,76	2,02
7.2	More efficient way of project planning	3,23	2,95	2,16	2,34	2,74	3,69	1,19	9,65	2,34	1,29	3,93	2,76	2,93
7.3	Agile increases the ability to forecast	2,86	3,83	1,91	1,37	1,37	3,32	0,71	0,48	1,14	6,39	3,40	2,78	3,03
8.1	Agile methods complement state gate management	3,19	3,01	5,28	1,02	0,00	4,16	2,38	5,34	0,00	2,93	2,78	1,60	3,82
8.2	Improved resource management, producti functionality, on-time delivery, and quality control	2,98	3,87	5,04	2,04	2,29	3,00	1,19	3,62	1,52	3,16	3,41	4,87	2,95
8.3	Increased process control, transparency, and quality	3,32	4,04	3,52	1,02	1,52	3,52	4,76	8,04	2,49	1,98	3,13	2,44	3,13
8.4	Fosters continuous improvement and refinement of of the dev. Process	2,97	3,64	3,64	1,02	2,29	2,71	1,19	2,90	4,47	2,37	3,45	3,97	2,74
8.5	Allows to fail cheaply (each fail is detected on every small step)	2,84	3,94	3,06	5,10	1,52	2,56	2,38	1,19	1,51	1,39	2,65	3,14	3,41
9.1	Agile thrives in radically different environments	2,67	2,55	3,01	0,00	2,67	3,84	6,23	1,59	2,12	0,98	1,93	2,77	2,91
9.2	Agile methods are adaptable and compatible with tranditional practices	2,68	2,54	3,75	0,00	8,84	2,76	6,23	1,59	7,26	1,19	2,76	1,85	1,98
10.1	Decreases effort related to changing requirements	2,43	2,78	0,21	0,88	1,26	3,12	2,00	3,75	1,45	5,65	2,35	2,31	2,21
10.2	Decrease effort for redesign	2,24	3,55	0,21	3,30	1,26	1,96	2,00	3,25	1,21	3,75	3,12	2,31	1,87
10.3	Decreases effort related to waste (not used/delivered but completed work)	2,36	2,30	0,26	4,01	1,68	2,27	1,76	4,33	1,12	2,32	3,27	1,15	2,22
10.4	Decreases effort due to rework related to faults	1,81	1,61	0,21	0,88	2,52	2,07	1,51	1,19	1,68	1,93	1,58	2,88	1,87
10.5	Decreases effort of extensive documentation	2,15	1,80	0,15	3,50	1,68	2,10	1,51	2,27	1,36	2,28	2,97	2,88	1,95

Fig. 6 Prioritization of Internal Benefit within Categories

4.5 RQ4: Which are the most significant/insignificant limitations of agile practice usage?

An overview of the limitation categories and the limitations within the categories is shown in Figures 7 and 8. The most significant limitations of utilizing agile practices in the development process are *professional skill specific demands* and *scalability*. Limitations associated with cost drivers (*increased testing* as well as *process effort*) received low prioritization values in comparison. The most favorable alternatives to avoid *increased effort* and *lead-time*

limitations are agile with few rigid development practices and rigid with few agile practices.

	All	Groups of practices used					Adoption strategy						
		Cluster 16 (Agile/RD)	**Cluster 13** (Balanced)	**Cluster 10** (Agile)	**Cluster 11** (RD/agile)	**Others**	**Group 1 scenario 1:** Great inc. RD to Agile	**Group 1 scen. 2:** Big Bang RD to Agile	**Group 2:** Inc. adop. agile prac. to a RD process	**Group 3:** Tailoring a complex hybrid process	**Group 4 scenario 5:** Initial simple agile process by adopting both agile and RD prac.	**Group 4 scenario 6:** Initial simple agile process by adopting only agile prac.	**Others**
11. Employee dissatisfaction	11,49	6,89	12,50	6,50	0,00	17,72	0,00	0,00	0,00	17,50	15,71	15,00	17,71
12. Professional skills specific demands	25,51	36,67	11,25	48,00	35,00	15,89	55,00	70,00	43,33	20,00	22,57	22,50	16,71
13. Limitation to specific product domains	13,54	9,22	15,00	16,00	10,00	14,61	10,00	0,00	13,33	4,00	9,00	12,50	17,71
14. Scalability	16,38	16,78	16,25	6,00	20,00	13,50	15,00	0,00	20,00	13,50	23,00	20,00	12,00
15. Increased testing effort	13,54	15,11	12,25	5,00	25,00	16,28	10,00	10,00	16,67	12,50	16,14	22,50	13,00
16. Increased process effort	10,00	7,11	16,25	13,50	5,00	10,78	10,00	10,00	3,33	15,00	8,71	2,50	10,95
17. Lead time and efficiency	9,54	8,22	16,50	5,00	5,00	11,22	0,00	10,00	3,33	17,50	4,86	5,00	11,90

Fig. 7 Prioritization of Limitation Categories

For the groups of practices used, the priorities are evenly distributed in the case of the balanced process. The practitioners also did not prioritize *professional and skill specific demands* highly for the balanced process, while the heatmap in Figure 7 shows that the priority values are at least three times higher for the alternatives. On the more detailed level it is visible that, depending on the groups of practices or adoption strategies used, the specific skills required for the different alternatives differ (see Figure 8).

For the adoption strategies, *professional skill specific demands* in Groups 1 and 2 in Figure 7 have been rated approximately twice as high as in the remaining adoption strategies. It is also evident that *employee dissatisfaction* is considered as irrelevant by Groups 1 and 2, while it has a relatively high priority in Groups 3 and 4. Furthermore, *testing effort* reeceived relatively high values in Group 4.

5 Discussion

5.1 Reflections on practice usage and adoption strategies

Tables 3 and 4 highlight the top ranked benefits in relation to combinations of practices used and adoption scenarios.

As Table 3 shows knowledge and learning perceived as a significant benefit across all identified approaches, which were attributed to improved communication and achievement of a common understanding. At the same time, professional skill demands were clearly prioritized as the main inhibitors and limitations in adopting agile practices. This applies independently of combinations of practices utlized or adoption strategies (see Tables 3 and 4) Also concerns of scalability and increased testing effort were widely regarded as major limitations in general, except for using those practices of Cluster 10. Comparing with the literature, scalability is often mentioned as a concern

		All	Groups of practices used					Adoption strategy						
			Cluster 16 (Agile/RD)	**Cluster 13** (Balanced)	**Cluster 10** (Agile)	**Cluster 11** (RD/agile)	**Others**	**Group 1 scenario 1:** Great inc. RD to Agile	**Group 1 scen. 2:** Big Bang RD to Agile	**Group 2:** Inc. adop. agile prac. to a RD process	**Group 3:** Tailoring a complex hybrid process	**Group 4 scenario 5:** Initial simple agile process by adopting both ag. and RD prac.	**Group 4 scenario 6:** Initial simple agile process by adopting only agile prac.	**Others**
11.1	Decreases developer motivation	3,65	2,74	5,45	0,00	0,00	4,93	0,00	0,00	0,00	3,68	7,43	4,25	3,04
11.2	Increase of stress level	3,46	1,95	3,53	0,00	0,00	4,82	0,00	0,00	0,00	6,37	4,43	6,38	3,03
12.1	Requires a comparabel level of qualifications of the developers	7,50	5,58	5,63	30,09	7,31	6,38	1,33	5,71	6,02	7,24	8,49	9,04	8,21
12.2	Requires high qualifications from all team members	9,25	15,81	5,29	34,64	1,25	4,80	2,67	45,71	2,55	2,60	10,98	9,04	9,48
12.3	Requires sufficiently trained managers	11,60	19,17	5,50	5,93	29,96	5,86	41,33	17,62	45,69	11,81	5,66	10,88	7,81
12.4	Makes team members less interchangeable	3,83	3,35	3,75	0,00	4,28	2,58	11,33	0,00	2,85	1,56	7,75	0,56	3,31
13.1	Limited support for developing safety-critical software	7,66	3,18	9,28	15,28	5,00	10,21	3,33	0,00	5,24	1,22	2,73	6,48	6,50
13.2	Limited support for developing legacy systems	3,16	1,98	4,14	7,65	1,25	3,57	3,33	0,00	3,69	1,00	1,86	2,78	3,03
14.1	Poorly scalable with respect to distributed environments	3,73	4,72	5,37	0,00	4,38	3,29	5,00	0,00	2,92	1,66	4,48	6,83	3,68
14.2	Not applicable for large teams	6,71	5,10	9,01	0,00	7,94	9,10	5,00	0,00	5,29	6,17	9,23	7,03	5,01
15.1	High effort for compensating for low test coverage	3,62	4,45	3,35	0,00	3,75	3,54	4,00	2,14	2,50	4,24	4,70	7,47	3,32
15.2	Requires high effort for producing testing documentation	2,63	1,93	2,92	0,00	6,90	2,68	2,00	0,00	4,60	3,88	2,90	2,28	2,71
15.3	Requires high effort for continuous testing	6,06	7,48	3,18	0,00	12,36	7,84	4,00	5,00	8,24	2,42	8,92	13,09	5,63
16.1	Increases the effort of configuration management	4,11	2,25	1,32	4,49	1,56	3,17	3,33	2,38	1,04	6,47	3,05	1,39	6,03
16.2	Increases the effort for commercial packaging of prod.	2,02	1,16	1,98	0,00	1,56	2,51	8,33	0,00	1,04	3,51	3,34	0,93	2,31
16.3	Increasese development effort and delays due to dependencies	1,96	1,44	1,98	0,00	1,56	2,31	1,67	0,00	1,04	3,55	1,73	0,00	2,70
16.4	Increases management overhead	3,41	1,81	4,61	1,92	1,56	4,29	1,67	2,38	1,04	6,45	2,78	2,31	4,06
16.5	Increases maintenance effort	2,73	1,43	3,30	0,00	1,56	3,85	1,67	7,14	1,04	1,25	5,35	0,00	4,03
17.1	Inefficient in hand-overs from requirements to design	2,39	2,65	5,88	0,00	1,56	2,77	0,00	0,00	1,04	2,49	0,85	1,85	2,69
17.2	Results in delays in getting functional tests running	3,77	2,01	4,84	0,00	1,56	5,43	0,00	5,95	1,04	6,37	2,34	6,48	5,13
17.3	Inefficient way of reporting progress on frequent meetings	2,59	1,77	1,99	0,00	1,56	3,36	0,00	3,57	1,04	6,78	0,82	0,93	2,99
17.4	Reduced efficiency of development as close cooperation leads to exhaustion	1,58	2,93	1,91	0,00	1,56	1,41	0,00	0,00	1,04	3,19	0,13	0,00	2,01
17.5	Makes inter-team communication inefficient	2,57	5,12	5,81	0,00	1,56	1,84	0,00	2,38	1,04	6,09	0,05	0,00	3,30

Fig. 8 Prioritization of Limitations within Categories

for using agile methods [21,33,46]. However, research also acknowledges that scalability is considered as a rather manageable risk of using agile [6], and solutions are mentioned in Petersen and Wohlin [33]. Vijayasarathy and Turk [46] also identified a steep learning curve as one of the major hinders in adopting agile. This also became visible when we investigated the adoption scenarios over time. With the big-bang transition and major changes made to the initial process, professional skills specific demands and items within that category were visibly highlighted (see Figures 7 and 8).

Besides having many prioritizations in common across clusters and adoption strategies, there were also major differences. As one example, time as an external benefit has been rated very highly for Cluster 10 (Agile) and Cluster 11 (RD/agile), while this was not the case for other practice combinations (see Figure 4). As another example, if one would like to excel in terms of time and cost, the survey indicates that practices of Cluster 11 are best suited, while to perform very well on value Cluster 10 is emphasized (see Figure 4). On the other hand, if all benefits should be achieved good degree, Clusters 16 and

Table 3 Top Benefits in relation to Combinations of Practices Used

	Cluster 16 (Agile with few RD practices)	Cluster 13 (Balanced hybrid process)	Cluster 10 (Mainly agile practices)	Cluster 11 (Mainly RD with few agile practices)
Benefits (groups) Top 3	1. Verification and validation 2. Employee satisfaction 3 Knowledge and learning	1. Social skill development 2. Knowledge and learning 3. Feedback and confidence	1. Employee satisfaction 2. Knowledge and learning 3. Feedback and confidence	1. Verification and validation 2. Knowledge and learning 3. Social skill development
Benefits (items) Top 5	1. Increased process control, transparency, and quality 2. Improved turn-around time for fixing bugs 3. Increased motivation and empowerment 4. Allows to fail cheaply 5. People feel purposeful	1. Develop communication skills 2. Complements state gate management 3. Improved resource management 4. Increased motivation and empowerment 5. Increased awareness of what has to be developed	1. Early verification of design decisions 2. People feel purposeful 3. Increase in confidence and satisfaction 4. Improved communication 5. Common understanding	1. Improved turn-around time for fixing bugs 2. More efficient use of time of testers 3. Agile methods are adaptable and compatible with trad. pract. 4. Common understanding 5. Develop cooperation skills
Limitations (groups) Top 3	1. Professional skills specific demands 2. Scalability 3 Increased testing effort	1. Lead time and efficiency 2. Scalability 3. Increased process effort	1. Professional skills specific demands 2. Limitation to specific product domains 3. Increased process effort	1. Professional skills specific demands 2. Increased testing effort 3. Scalability
Limitations (items) Top 5	1. Requires sufficiently trained managers 2. Req. high qualification from all team members 3. Requires high effort for continuous testing 4. Req. comparable level of qual. of the developers 5. Makes inter-team communication inefficient	1. Limited support for developing safety-critical SW 2. Not applicable for large teams 3. Inefficient hand-overs from requirements to design 4. Makes inter-team communication inefficient 5. Req. comparable level of qual. of the developers	1. Req. high qualification from all team members 2. Req. comparable level of qual. of the developers 3. Limited support for developing safety-critical SW 4. Limited support for developing legacy systems 5. Requires sufficiently trained managers	1. Requires sufficiently trained managers 2. Requires high effort for continuous testing 3. Not applicable for large teams 4. Req. comparable level of qual. of the developers 5. Requires high effort for producing testing documentation

13 would be the preferred ones (see Figures 5 and 6). Given that numerous differences were observed between the strategies, this research indicates that adoption strategies as well as practice combination have an important impact on the benefits achieved and the limitations manifesting themselves. Details on which benefits and limitations are reported for combinations of practices and adoption strategies are provided in Sections 4.4 and 4.5.

5.2 Practical implications

Independent of the set of practices used or the adoption strategy followed, when comparing for adoption of agile practices practitioners should assure sufficient training of the software developers, e.g. by utilizing agile experts and coaches. This is of particular relevance given that the category professional skills specific demands was highly prioritized. In the training it should be observed that all members require high qualifications, and also should have a comparable level of qualifications. Furthermore, managers also need to be sufficiently trained. Scalability issues need to be addressed, which were mainly related to distributed environments and applicability to large teams.

Table 4 Top Benefits in relation to Combinations of Practices Used

	Group 1 Scenario 1	Group 1 Scenario 2	Group 2	Group 3	Group 4 Scenario 5	Group 5 Scenario 6
Benefits (groups) Top 3	1. Adaptability	1. Verification and validation	1. Verification and validation	1. Knowledge and learning	1. Feedback and confidence	1. Verification and validation
	2. Quality	2. Monitoring and controlling	2. Knowledge and learning	2. Quality	2. Quality	2. Feedback and confidence
	3. Feedback and confidence	3. Planning and estimations	3. Adaptability	3. Planning and estimations	3. Planning and estimations	3. Knowledge and learning
Benefits (items) Top 5	1. Adaptable and compat. with traditional methods	1. More efficient way of project planning	1. Improved turn-around time for fixing bugs	1. Code reuse and higher quality	1. Increase awareness of what has to be dev.	1. Improved turn-around time for bug fixing
	2. Thrives in radically different environments	2. Increased process control, trans. and qual.	2. More efficient use of time of testers	2. Agile increases the ability to forecast	2. Early requirements validation	2. Quicker/more thorough response to def. from dev.
	3. Increases awareness of what has been dev.	3. Quicker/more thorough response to def. from dev.	3. Improved communication	3. Common understanding	3. Small and precise scope estimations	3. Improved resource management
	4. Improves qual. of design, arch. and perf.	4. More efficient use of time of testers	4. Adaptable and compat. with trad. pract.	4. Decreases effort related to changing requirements	4. People feel purposeful	3. Improved communication
	5. Code reuse and higher quality	5. Complement state gate management	5. Common understanding	5. Increases motivation and empowerment	5. Improved turn-around time for bug fixing	2. Early requirements validation
Limitations (groups) Top 3	1. Professional skills specific demands	1. Professional skills specific demands	1. Professional skills specific demands	1. Professional skills specific demands	1. Scalability	1. Professional skills specific demands
	2. Scalability	2. Increased testing/ process effort	2. Scalability	2. Lead time and efficiency	2. Professional skills specific demands	2. Increased testing effort
	3 Increased testing/process effort; product domains	3. Lead time and efficiency	3. Increased testing effort	3. Employee dissatisfaction	3. Increased testing effort	3. Scalability
Limitations (items) Top 5	1. Requires sufficiently trained managers	1. Req. high qualification from all team members	1. Requires sufficiently trained managers	1. Requires sufficiently trained managers	1. Req. high qual. from all team members	1. Requires high effort for continuous testing
	2. Makes team members less interchangeable	2. Requires sufficiently trained managers	2. Requires high effort for continuous testing	2. Req. comparable level of qual. of the developers	2. Not applicable for large teams	2. Requires sufficiently trained managers
	3. Increases effort for commerce. pack. of prod.	3. Increases maintenance effort	3. Req. comparable level of qual. of the developers	3. Ineff. way of reporting progress on freq. meet.	3. Requires high effort for continuous testing	3. Req. high qual. from all team members
	4. Not applicable for large teams	4. Results in delays in getting fund. test running	4. Not applicable for large teams	4. Increases effort of configuration mgt.	4. Req. comparable level of qual. of the developers	4. Req. comparable level of qual. of the developers
	5. Poorly scalable with respect do distr. env.	5. Req. comparable level of qual. of the developers	5. Limited support for dev. safety-critical software	5. Increases management overhead	5. Makes team members less interchangeable	5. High effort for compens. for low test cov.

The results presented allow practitioners to choose the benefits (internal as well external) they would like to achieve most and the limitations they are willing to tolerate, address, or would like to avoid. As mentioned earlier, many benefits and limitations differ in their severity between combinations of practice usage and adoption strategies. With this information the results allow practitioners to choose the practice combination and adoption strategies that would fit their priorities best (e.g. balanced distribution of benefits versus highly emphasizing a specific benefit, such as cost). However, it is important to highlight that the information provided in this paper should be used to reflect and discuss, rather than to automatically choose an approach from the data. To achieve this level of confidence, further studies are needed, as is discussed in the research implications.

Based on the data multiple observations could be made. We present the observations, compare them to related work and highlight the main lessons.

With respect to benefits of agile adoption two main observations could be made:

Observation 1: The most significant external benefits visible outside the organization were value, relationship with the customer, and quality.

Comparison with related work: One of the main concepts of agile is to deliver products that the customers desire [18]. As our research has shown, this

is the most recognizable benefit by the surveyed practitioners. The customer is involved in the development process and can respond to changes or doubts very quickly. In our study this is confirmed by the high rating of value delivered and the importance of the relationship with the customer. It has been reported in literature that agile leads to improved quality of software and decreased lead times [35]. Whereas the first benefit maps to the results of the survey (Quality benefits), the second (Lead time) was identified as the least significant benefit of agile for the customer in this survey. The top agile benefits that appear in the research by Vijayasarathy and Turk [46] are: (1) Better meets customer needs; (2) improved software quality; (3) increased flexibility in development; (4) faster time to delivery; (5) lower development costs. They recognized (4), faster time do delivery, as a very significant benefit of agile usage. The agile adopters in our study report this benefit as the least singnificant. The other benefits map well to the results of this study.

Implications: In the long run, it turns out that increased value, relationship with the customer and quality remain the most essential benefits of using agile.

Observation 2: The most significant internal benefit categories from adopting agile are knowledge and learning, employee satisfaction, social skill development, and feedback and confidence. On the detailed level, agile makes people feel purposeful, improves knowledge transfer and learning between team members, leads to early requirements validation due to frequent feedback, and improves turn-around time for fixing bugs.

Comparison with related work: In general, it can be observed that the benefits of agile are strongly related to human factors. As it was mentioned in the Agile Manifesto [18], *"it's people who make the difference between success and failure"* and *"it's important to maximize that first-order people factor"*. Agile embraces the social aspects and supports quality of the working life. The prioritization of benefits partially overlaps with the one presented in Begel and Nachiappan [1]. The top reported agile benefits of the study [1] were: (1) Improved Communication and Coordination; (2) Quick Releases; (3) Flexibility of Design. The benefit number (2) is recognized as a practice in this research. Nevertheless, the benefit number (1) refers to Social Skills Development and Knowledge and Learning, whereas (3) is strongly linked to Feedback and confidence.

Implications: Based on the results above, it is recommended that practitioners should adopt agile methods in order to increase employee satisfaction, improve knowledge transfer and learning, help people develop social skills and increase confidence of the development teams that what they are doing is consistent with what customers expect.

Observation 3: Professional skills specific demands, scalability and lack of suitability for specific product domains are the main limitations of agile practice usage. On a detailed level, agile requires sufficiently trained managers and high qualifications from all team members. It provides limited support for developing safety-critical systems and is not applicable to large teams.

Comparison with related work: Skill demands and Scalability were rated as relatively important over other elements. Professional Skills Specific Demands is an agile matter that has already been raised in literature [35,6]. Building an effective agile team is challenging. Lack of experience may result in great delays when new practices are implemented [19]. Other issues were reported when a team consisting of very experienced quality assurance professionals was trying to adapt automated testing into their new agile process [43]. In this case, only agile consultants' coaching could finally guide the team to make the process work properly. Moreover, managerial skills are crucial. The team leader is responsible for constant reduction of risks, recording the progress of his team, and reacting to each difficulty as soon as possible [38].

Scalability is a recognized issue of agile methods. It is often assumed that agile is only suitable for small teams and projects [21]. However, scalability is considered as a rather manageable risk of using agile [6]. Although some possible solutions exist [33], the majority of the research participants have recognized scalability as a relatively significant limitation and it is still an open issue to be addressed by researchers.

Vijayasarathy and Turk [46] revealed limitations that appear when adopting agile methods. The most important ones were: Limited support for distributed development environment; limited support for development involving large teams; steep learning curve. These issues relate to scalability as well as learning. Furthermore, the lack of suitability for developing safety-critical software was identified by Vijayasarathy and Turk [46] as a major factor, aligning well with our results.

Implications: It is recommended that practitioners should invest in training on agile in order to make the adoption work effectively. As it has been observed in previous studies, practitioners underestimate the need of high professional skills. However, in this study it is apparent that lack of knowledge on agile methods is one most significant obstacles on the way to effective agile processes. With respect to scalability, it was observed that agile methods were successfully applied also in large-scale organizations (c.f. [33]). Given that scalability is still perceived as a major limitation by practitioners, technology transfer from research and practice is needed to help organizations in large-scale agile development. In particular, quality of design seems to be an issue that needs to be addressed so that practitioners could benefit from agile adoption in large-scale organizations.

Observation 4: Different benefits and limitations in relation to practice combinations are observed. In particular, agile with few rigid development practices and balanced processes allow to achieve a high number of benefits.

Implications: Based on the results reported, we found that agile adoption is beneficial. Our research indicates that practitioners should aim at agile-dominated processes with a few rigid development practices. In contrast to other development models, this approach allows to perceive a number of benefits of relatively similar significance in all development related aspects (e.g. quality, employee satisfaction etc.). Choosing a different model (e.g. rigid development with few agile practices) leads to excellent performance in verification and validation, though other benefits are compromised (see Figure 5).

Observation 5: Different benefits in relation to adoption strategies are observed depending on adoption scenarios. For example, big-bang transition from RD to agile leads to poorer quality in comparison to the alternatives.

Implications: Thus far, we were not able to identify a study looking at agile adoption over time, hence no related work is reported. Based on our findings (see Figures 5 and 6) it is recommended that organizations, which decide to switch from traditional approaches, should conduct the transition in an incremental way. In contrast to a big-bang transition, a number of benefits is perceived with this approach, the most highly rated ones being adaptability and increased quality of code as well as of design.

Rigid development organizations that decide only to adopt a few agile practices also experience improvements to their processes. It is also suggested that recently created organizations should build their process in an incremental, bottom-up way instead of tailoring an initially complex process (top-down). The practices that should be introduced at first are mainly related to increased interaction within development units through small teams and face-to-face communication. They were a starting point in the process of agile adoption in most of cases.

Observation 6: Not only RD practices are abandoned over time, but also agile practices. The most frequently abandoned agile practices are pair programming, test-driven development, and continuous integration with testing. Table 5 shows the frequency of abandonment of practices over time.

Implications: Due to the dynamics of market and the need of being flexible to frequent requirements change, practices such as detailed up-front architecture design, detailed management plans, extensive process documentation and

Table 5 Practices Abandonment Rate

Practice	Abandoned	Applied (total)	Abandoned %
Detailed up-front architecture design	9	18	50.00%
Detailed mgt. plans and process documentation	11	23	47.83%
The project follows a sequential flow of phases	10	24	41.67%
Big bang integration and infrequent releases	6	18	33.33%
Extensive time planning	7	22	31.82%
Up-front documentation of requirements	8	28	28.57%
Pair-programming	4	14	28.57%
Test-driven development	4	23	17.39%
Continuous integration with testing	4	34	11.76%

sequential development are no longer applicable and should be used to a minimal extent if necessary. The research has shown that organizations, which follow agile dominated processes or decided to switch to agile from traditional approaches, perceive a number of benefits and claim to follow much more reasonable and effective processes. However, some agile practices are also abandoned over time. To remind, the most significant limitations of agile are related to the lack of specific professional skills. The reason for relatively frequent (when compared to other practices) abandoment of practices such as pair-programming, test-driven development or continuous integration with testing might be strongly related to the required skills. The issues related to the application of test-driven development [19] and continuous integration with automated testing [43] have already been reported in literature. The reasons given in the studies were related to lacks of professional experience, agile coaching, professional knowledge, etc. The abandonment of practices indicates that there are challenges in their application. The purpose of this study was different and it provides too little data in its current version. Nonetheless, this observation is considerable and the problem should be further investigated by agile practitioners and researchers.

5.3 Research implications

In this research, we investigated how practices are adopted over time by practitioners. As the history and time perspective is often not considered in case studies, these classifications could be used to reflect on how agile has evolved over time when conducting longitudinal studies in agile software development. Such studies were regarded as important by Dybå and Dingsøyr [15], who highlighted that only few of such studies exist in the literature.

Looking at clusters of practice adoption, it was very evident that practitioners tailor the development processes to their needs, as several different clusters have been identified. Though it was noticeable that certain practices

appear together, referring to agile processes in practice can mean very different applications of practices. In this study, we utilized labels to reflect and refer to groupings. However, this was primarily done for the purpose of referring to clusters and approximating what they are presenting.

When looking at the whole data set (not distinguishing between clusters) the results were quite balanced. However, when looking at the individual groupings, a few prioritizations were very different. Thus, when inferring results from agile, what agile means has to be characterized on a more fine-granular level in study reporting. Otherwise, it is not possible to compare studies on agile with each other. In particular, the set of practices and the adoption strategy would be useful to report in combination with additional contextual information (cf. [32]).

We captured cost and effort drivers in the survey. However, it would be important to consider corporate measures of cost associated with organizational transformations. Given that a survey instrument was used, we could not capture accurate cost measures based on, for example, time reporting. Furthermore, important contextual events (e.g. major releases) affecting the cost could not be captured. Thus, we highlight that the research in this paper needs to be complemented by industrial case studies to provide reliable information on cost figures.

Bear in mind that, to the best of our knowledge, this is the first study focusing on identifying the relative priorities of agile benefits and limitations, and relating them to practice usage and adoption strategies, the findings provide valuable indications and allow to develop propositions.

5.4 Threats to Validity

Gencel and Petersen [30] proposed a classification of validity threats to be discussed in empirical software engineering studies.

Descriptive validity (Factual accuracy): This threat is concerned with the ability to objectively describe the reality as perceived by the practitioners. One threat was related to the ability of respondents to indicate the relative significance of the benefits and limitations. The respondents could find it hard to compare a number of issues related to highly different development aspects. The number of comparisons needed was reduced by voting in hierarchies [3]. To assure the understandability of the benefits and limitations an external expert reviewed the questions. For the purpose of extensive research in Poland, it was decided to translate the questionnaire into the Polish language. The risk of wrong and ambiguous translation was partially reduced thanks to feedback and language checks by two IT developers fluent in both languages. Furthermore, there is a need to have practitioners with the experience using the development processes. Overall, the experience of the participants was quite high.

Theoretical Validity; Theoretical validity is concerned with the ability to capture what we intend to capture, as well as with potential confounding factors. There might be more factors that were not part of the study. To

reduce this threat, the respondents had the opportunity to leave comments about the practices or missing benefits and limitations at the end of the questionnaire. Numerous studies (see related work in Section 2) already conducted qualitative research in these areas, providing a sound foundation for benefit and limitation identification. In addition, an external agile expert that was not involved in the research reviewed the lists and did not identify any missing items. Maturation is a threat to the theoretical validity, e.g. we cannot be sure if the practitioners read through all benefits and limitation items in their respective categories before prioritizing. To reduce this threat, we relied mainly on personal contacts. Furthermore, we tested the questionnaire to take about an hour to finalize, informing the participants prior to participation. Important confounding factors when conducting industrial studies are, for example skills of developers, roles and perspectives, scale, as well as product domain (cf. [32]). These have been captured and considered in the interpretation of the results and validity of the study (in particular external validity). Furthermore, it is important to note that no cause-effect relationships could be determined, only how the situation changes when looking at the data from different perspectives (e.g. with respect to development models).

Interpretive validity: Interpretive validity is concerned with researcher bias and the ability to draw reasonable conclusions given the data. In order to reduce this threat, an existing analysis framework [24] has been applied, providing guidance of how to visualize and interpret cumulative voting data.

Generalizability: This type of validity is concerned with the degree to which we can generalize the results. Given the intention to capture rich information about benefits and limitations in relative comparison the number of answers was limited, similar to what has been observed in previous cumulative voting studies [24,39]. Looking at the demographics, the study shows that the majority of responses come from the actual development perspective, followed by process experts. Hence, this perspective is well represented, while few persons answered from the business perspective and quality assurance perspective. Answers are evenly distributed with respect to organizational sizes, which indicates good coverage. Furthermore a high number of unique organizations has participated, assuring that the results are not biased towards a particular organizational context. Since the cross-analysis of development methods and adoption strategies with perceived benefits and limitations has not been done before, no direct comparability of the data distributions was possible. However, there was a study on agile practices adoption, which results are similar to this research [23]. The number of participants from Poland is over-represented. Given that we cannot be assured (and did not find any evidence to that account) how the Polish software industry does or does not differ from software industries in other countries, we cannot be sure to what degree the findings can be generalized. Hence, given that this is an initial exploration of how benefits and limitations are prioritized relative to each other, and to other practices, there is a need for replications to further strengthen the generalizability.

6 Conclusion

The aim of this research was to understand which practice combinations are used in industry and how these relate to agile benefits and limitations. For that four research questions have been formulated for this research, concerning which practices are combined, which strategies are followed in adopting practices over time, and which benefits/limitations result from using the practices.

The aim was achieved through a targeted survey based on existing evidence and a multidimensional data analysis. The mean for obtaining data is a web-based questionnaire with an interactive board with practices and time indication sliders (to capture applied development models and practices adoption strategies) and hierarchical cumulative voting (to measure the relative significance of benefits and limitations). The data analysis is supported by hierarchical cluster analysis and an extended hierarchical voting analysis framework (EHV-F).

In total, 45 practitioners have been successfully surveyed. The commonness of the use of 21 development practices was investigated. The relative significance of agile adoption benefits (32 factors in 10 categories) and limitations (23 factors in 7 categories) was measured with respect to a global view (all respondents and perspectives), different agile adoption strategies as well as distinguished development models. Four research questions have been explored.

RQ1: Which practice combinations are currently used by practitioners? The respondents were asked to define their development processes in terms of practices that they applied. A hierarchical clustering algorithm was applied in order to classify practices that are commonly applied together. The most popular combinations are as follows: 1) Face-to-Face communication facilitates Frequent planning and reporting and Prioritized list of requirements development and management, 2) Iteration planning meeting and Iteration review and retrospectives are a strongly linked combination, both facilitated by small self-organizing cross-functional teams, 3) Continuous integration with testing facilitates Short iterations and releases. Four different ways of combining practices were distinguished: 1) agile process with a few rigid development practices; 2) hybrid, balanced process; 3) agile process and 4) rigid development process with a few agile elements. It is apparent that agile is frequently combined with rigid development approaches, however, agile often dominates over rigid approaches.

RQ2: Which strategies were followed in adopting agile practices over time? The development models were analyzed in a graphical way in order to observe agile adoption strategies. Different adoption strategies towards agile development models were distinguished. The patterns were: 1) a great incremental transition from regid development towards agile; 2) a great big-bang transition from rigid develoipment towards agile; 3) incremental adoption of agile practices to a rigid process; 4) tailoring of a complex hybrid process (bottom-

down) and 5) developing an initial simple agile process by adopting other practices (bottom-up).

RQ3: Which are the most significant/insignificant benefits of agile practice usage? The most significant benefits externally visible to the customer were value, relationship with the customer, and product quality. The most significant benefits within the company were improved knowledge and learning, employee satisfaction, social skill development, and feedback and confidence. On the detailed leve, agile makes people feel purposeful, improves knowledge transfer and learning between team members, leads to early requirements validation due to frequent feedback, and improves turn-around time for fixing bugs. As product quality is important, when transitioning from rigid to agile development a big-bang transition should be avoided as it leads to poorer quality in comparison to the alternatives.

RQ4: Which are the most significant/insignificant limitations of agile practice usage? Professional skills specific demands, scalability and lack of suitability for specific product domains are the main limitations of agile practice usage. Agile requires sufficiently trained managers and high qualifications from all team members. It provides limited support for developing safety-critical systems and is not applicable to large teams. We also observed that not all agile practices may lead to the desired benefits. The most commonly discarded practices that were tried in the studied companies were pair programming, test-driven development, and continuous integration with testing. Hence, there is a need to understand why companies did not consider the practices to be useful enough to continue using them.

Further studies are needed in order to further substantiate the findings. In particular, we propose to continue the line of research in the following ways: (1) continue the research and extend it to a broader population in order to achieve increased generalizability of the results; (2) address this research to new adopters as well as there is little information of what organizations expect from agile adoption, and what they actually achieve over time (short-term vs. long-term); (3) focus on the limitations of agile adoption found through this study as significant (e.g. professional skills specific demands, agile scalability) and research for solutions that would help to overcome these challenges.

Acknowledgements The work also funded partially by ELLIIT, the Strategic Area for ICT research, funded by the Swedish Government. We also would like to thank all participants of this study.

References

1. Andrew Begel and Nachiappan Nagappan. Usage and perceptions of agile software development in an industrial context: An exploratory study. In *Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on*, pages 255–264. IEEE, 2007.
2. Patrik Berander and Anneliese Andrews. Requirements prioritization. In *Engineering and managing software requirements*, pages 69–94. Springer, 2005.

3. Patrik Berander and Per Jönsson. Hierarchical cumulative voting (hcv)prioritization of requirements in hierarchies. *International Journal of Software Engineering and Knowledge Engineering*, 16(06):819–849, 2006.
4. Barry Boehm. Project termination doesn't equal project failure. *Computer*, 33(9):94–96, 2000.
5. Barry Boehm. Get ready for agile methods, with care. *Computer*, 35(1):64–69, 2002.
6. Barry Boehm and Richard Turner. Observations on balancing discipline and agility. In *Agile Development Conference, 2003. ADC 2003. Proceedings of the*, pages 32–39. IEEE, 2003.
7. Barry Boehm and Richard Turner. Using risk to balance agile and plan-driven methods. *Computer*, 36(6):57–66, 2003.
8. Frederick P Brooks Jr. No silver bullet-essence and accidents of software engineering. *IEEE computer*, 20(4):10–19, 1987.
9. Robert N Charette. Why software fails. *IEEE spectrum*, 42(9):36, 2005.
10. Tsun Chow and Dac-Buu Cao. A survey study of critical success factors in agile software projects. *Journal of Systems and Software*, 81(6):961–971, 2008.
11. Danny T Connors. Software development methodologies and traditional and modern information systems. *ACM SIGSOFT Software Engineering Notes*, 17(2):43–49, 1992.
12. Helene Dahlberg, Francisco Solano Ruiz, and Carl Magnus Olsson. The role of extreme programming in a plan-driven organization. In *The Transfer and Diffusion of Information Technology for Organizational Resilience*, pages 291–312. Springer, 2006.
13. Carsten Dogs and Timo Klimmer. *An evaluation of the usage of agile core practices*. PhD thesis, Masters thesis, Blekinge Institute of Technology, Sweden, 2004.
14. David E Drehmer and Sasa M Dekleva. A note on the evolution of software engineering practices. *Journal of Systems and Software*, 57(1):1–7, 2001.
15. Tore Dybå and Torgeir Dingsøyr. Empirical studies of agile software development: A systematic review. *Information and software technology*, 50(9):833–859, 2008.
16. Holmes Finch. Comparison of distance measures in cluster analysis with dichotomous data. *Journal of Data Science*, 3(1):85–100, 2005.
17. Andrew Forward and Timothy C Lethbridge. A taxonomy of software types to facilitate search and evidence-based software engineering. In *Proceedings of the 2008 conference of the center for advanced studies on collaborative research: meeting of minds*, page 14. ACM, 2008.
18. Martin Fowler and Jim Highsmith. The agile manifesto. *Software Development*, 9(8):28–35, 2001.
19. Matt Ganis, David Leip, Fred Grossman, and Joe Bergin. Introducing agile development (xp) into a corporate webmaster environment-an experience report. In *Agile Conference, 2005. Proceedings*, pages 145–152. IEEE, 2005.
20. Judy H Gray, Iain L Densten, and James C Sarros. Size matters: Organisational culture in small, medium, and large australian organisations. *Journal of Small Business & Entrepreneurship*, 17(1):31–46, 2003.
21. Tomohiro Hayata and Jianchao Han. A hybrid model for it project with scrum. In *Service Operations, Logistics, and Informatics (SOLI), 2011 IEEE International Conference on*, pages 285–290. IEEE, 2011.
22. Michael Hirsch. Moving from a plan driven culture to agile development. In *International Conference on Software Engineering (ICSE 2005)*, volume 27, page 38, 2005.
23. Narendra Kurapati, Venkata Sarath Chandra Manyam, and Kai Petersen. Agile software development practice adoption survey. In *Agile Processes in Software Engineering and Extreme Programming*, pages 16–30. Springer, 2012.
24. Ludwik Kuzniarz and Lefteris Angelis. Empirical extension of a classification framework for addressing consistency in model based development. *Information & Software Technology*, 53(3):214–229, 2011.
25. Maarit Laanti, Outi Salo, and Pekka Abrahamsson. Agile methods rapidly replacing traditional methods at nokia: A survey of opinions on agile transformation. *Information & Software Technology*, 53(3):276–290, 2011.
26. Sridhar Nerur, RadhaKanta Mahapatra, and George Mangalaraj. Challenges of migrating to agile methodologies. *Communications of the ACM*, 48(5):72–78, 2005.
27. Linda Dailey Paulson. Adapting methodologies for doing software right. *IT Professional*, 3(4):13–15, 2001.

28. Kai Petersen. Implementing lean and agile software development in industry, phd thesis, doctoral dissertation series no. 2010:04. Technical report, Blekinge Institute of Technology, 2007.
29. Kai Petersen. Is lean agile and agile lean. *A comparison between two software development paradigms, Modern software engineering concepts and practices: advanced approaches, IGI Global*, pages 19–46, 2011.
30. Kai Petersen and Cigdem Gencel. Worldviews, research methods, and their relationship to validity in empirical software engineering research. In *Proceedings of the Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement (IWSM-Mensura 2013)*, 2013.
31. Kai Petersen and Claes Wohlin. A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case. *Journal of Systems and Software*, 82(9):1479–1490, 2009.
32. Kai Petersen and Claes Wohlin. Context in industrial software engineering research. In *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, pages 401–404. IEEE Computer Society, 2009.
33. Kai Petersen and Claes Wohlin. The effect of moving from a plan-driven to an incremental software development approach with agile practices. *Empirical Software Engineering*, 15(6):654–693, 2010.
34. Kai Petersen, Claes Wohlin, and Dejan Baca. The waterfall model in large-scale development. In *Product-Focused Software Process Improvement*, pages 386–400. Springer, 2009.
35. Minna Pikkarainen, Outi Salo, Raija Kuusela, and Pekka Abrahamsson. Strengths and barriers behind the successful agile deployment - insights from the three software intensive companies in finland. *Empirical Software Engineering*, 17(6):675–702, 2012.
36. Alcides Quispe, Maira Marques, Luis Silvestre, Sergio F Ochoa, and Romain Robbes. Requirements engineering practices in very small software enterprises: A diagnostic study. In *Chilean Computer Science Society (SCCC), 2010 XXIX International Conference of the*, pages 81–87. IEEE, 2010.
37. K. Rinkevics and Richard Torkar. Equality in cumulative voting: A systematic review with an improvement proposal. *Information & Software Technology*, 55(2):267–287, 2013.
38. Linda Rising and Norman S Janoff. The scrum software development process for small teams. *Software, IEEE*, 17(4):26–32, 2000.
39. Per Rovegard, Lefteris Angelis, and Claes Wohlin. An empirical study on views of importance of change impact analysis issues. *Software Engineering, IEEE Transactions on*, 34(4):516–530, 2008.
40. Hossein Saiedian and R Dale. Requirements engineering: making the connection between the software developer and customer. *Information and Software Technology*, 42(6):419–428, 2000.
41. James Shore et al. *The art of agile development*. O'Reilly, 2008.
42. Ian Sommerville. *Software Engineering (9th edition)*. Addison Wesley Professional, 2010.
43. Megan Sumrell. From waterfall to agile-how does a qa team transition? In *Agile Conference (AGILE), 2007*, pages 291–295. IEEE, 2007.
44. SH VanderLeest and A Buter. Escape the waterfall: Agile for aerospace. In *Digital Avionics Systems Conference, 2009. DASC'09. IEEE/AIAA 28th*, pages 6–D. IEEE, 2009.
45. John Viega and Gary McGraw. *Building secure software: how to avoid security problems the right way*. Pearson Education, 2001.
46. LEOR Vijayasarathy and Dan Turk. Agile software development: A survey of early adopters. *Journal of Information Technology Management*, 19(2):1–8, 2008.
47. Dave West, Tom Grant, M Gerush, and D DSilva. Agile development: Mainstream adoption has changed agility. *Forrester Research*, 2010.
48. Laurie Williams. Agile software development methodologies and practices. *Advances in Computers*, 80:1–44, 2010.
49. Claes Wohlin and Rafael Prikladnicki. Systematic literature reviews in software engineering. *Information & Software Technology*, 55(6):919–920, 2013.

A Surveyed Practices

The following practices have been surveyed:

1. *Up-front documentation of requirements:* Requirements are fully documented, and as complete as possible
2. *Detailed up-front architecture design:* Detailed and extensive design of entire software architecture, documentation-heavy
3. *Big bang integration and infrequent releases:* Most of the implementation artifacts, such as code and modules, are integrated at once before milestones; releases are done at least once per 4 months or more
4. *Extensive time planning:* Fully specified Gantt charts with fixed milestones for the entire project, complete Work Breakdown Structure decomposition,
5. *The project follows a sequential flow of phases:* When one phase is finished, it is closed, e.g. 0) requirements, 1) design, 2) construction, 3) testing, 4) delivery.
6. *Detailed management plans and process documentation:* Formal and plan-based quality, risk, resource, change and configuration management
7. *Repeatable development process:* The process is formally defined and followed, it is repeatable, its flexibility and abilities to adapt to projects are limited
8. *Face-to-face communication:* Team sits together, open space office facilitating interaction, video conferences if the team is distributed
9. *Technical excellence:* Ongoing refactoring of the code, simplest solution design, following coding standards
10. *Small self-organizing cross-functional teams:* The team is independent, takes full responsibility; small with no more than 10 members
11. *On-site customer:* Continuous user involvement in the development process, customer can be consulted anytime if it is needed
12. *Frequent planning/reporting:* Everyday meetings to discuss what was and what is going to be done, reporting the progress on a burn-down chart, burn-up chart, informative workshops
13. *Prioritized list of requirements:* Tasks in a backlog as features/user stories - short statements of the functionality, system metaphors - stories about how the system works
14. *Pair-programming:* Two developers work together at one workstation, they switch ro
15. *Time boxing:* Fixed time of iterations, meetings, working hours limited to 40h weekly - employees hardly ever work overtime, no "death march" projects, team members avoid burnout etc.
16. *Continuous integration with testing:* Software is built frequently, even few times a day, accompanied with testing (e.g. ten-minute builds, automated unit, regression, etc.)
17. *Short iterations and releases:* Frequent releases of the software, at most 3-4 months, early and continuous delivery of partial but fully functional software
18. *Iteration reviews, retrospectives:* The entire team participates in selecting features to be implemented in the following iteration, estimating resources required to implement them, consensus based, e.g. planning game, the Wideband Delphi Estimation, planning poker etc.
19. *Iteration reviews, retrospectives:* Meetings after each iteration to review the project, discuss threats to process efficiency, modify and improve, build up the software development process
20. *Test-driven development:* Writing automated test cases for functionalities and then implementing the tested functionalities until the tests are passed successfully
21. *Collective code ownership:* Everybody in the team can change the code of other developers in case of maintenance, bug-fixing or other development activities

B Overview of Internal Benefits Derived from Literature

The following list shows the benefit categories surveyed (categories 1 to 10) and the items within the categories.

1. *Knowledge and Learning:* Agile results in improved knowledge transfer and helps to achieve the teams a common understanding.
   1.1. Agile improves knowledge transfer and learning between team members (e.g. how to do good testing, how to write good user stories, domain knowledge) as communication is improved (e.g. frequent contacts, exchange of programmers) [35, 29, 31, 46].
   1.2. Agile helps teams to achieve a common understanding (e.g. shared vision of the product requirements, the development process to be used) among team members [35, 25, 15, 31, 1].
2. *Social Skills Development:* Agile helps to develop socials skills such as communication, cooperation, adaptability.
   2.1. Agile helps to develop communication skills (e.g. presentation, constructive discussion, negotiations etc.) [15].
   2.2. Agile helps to develop cooperation-related skills (e.g. ability to help each other, solve problems together) [15, 31].
   2.3. Agile helps to develop adaptability skills (e.g. ability to adapt to changing conditions/change in plans) [15].
3. *Employee Satisfaction:* Agile improves the quality of working life. Employees feel more comfortable, relaxed, satisfied and purposeful.
   3.1. Agile leads to empowerment of the engineers and thus increased motivation due to its technical focus (e.g. frequent milestones and short iterations) [31].
   3.2. Agile makes people feel comfortable and relaxed, yet purposeful and productive (e.g. due to common ownership of product, 40 hour week, discussion of issues) [15, 31].
   3.3. Agile increases confidence and satisfaction (from perspective of the internal organization) of the product being developed as it gives the confidence that the developed software is what the customer desires (e.g. the team is capable to achieve at hand and quickly understand where the limitations were) [35, 15, 31, 46].
   3.4. Agile makes people feel purposeful as the team is a self-organizing community with a culture that emphasizes shared responsibility (e.g. team members have influence on decisions, there is a need to collaborate in order to achieve goals) [15].
4. F*eedback and confidence:* Agile increases the confidence of development team that what they are doing is appropriate. The feedback from customer is frequent and decisions are verified very soon.
   4.1. Agile allows to verify design decisions very soon due to frequent feedback (e.g. feedback from integration tests) [31].
   4.2. Agile increases the awareness of what has to be developed thanks to close cooperation with the customer (e.g. by on-site customers) [15, 29, 33].
   4.3. Agile leads to soon requirements validation due to frequent feedback on their work (e.g. feedback from customers after software presentations) [1, 46].
5. *Quality:* Agile results in improved quality of software products.
   5.1. Agile results in higher quality and reuse of code (e.g. ongoing refactoring, test-first programming, programming in pairs etc.) [25, 1, 31, 46].
   5.2. Agile leads to improved quality of design, architecture and performance as design decisions are verified very soon [1].
6. *Verification and validation processes improvement:* Agile leads to improved, more efficient verification and validation processes.
   6.1. Agile leads to improved turnaround time for fixing bugs due to early discovery and handling of issues and thus following development relies on verified work [1, 29].
   6.2. Agile provides much quicker and more thorough response to defects as responsibility for measuring quality is moved from the managers to the developers [15].
   6.3. Agile leads to more efficient use of testers' time as in small teams testing and design can be easily parallelized due to short ways of communication between designers and testers (instant testing) [31].
7. *Planning and estimations:* Agile allows for more accurate planning and estimations based on real data.
   7.1. Agile allows accurate small scope estimations as requirements in requirements packages are precise [31].

7.2. Agile allows more efficient way of project planning as the processes are organized in more releases and managers are more satisfied with the way they plan their projects (rather than are RD companies) [15].
7.3. Agile increases the ability to forecast (e.g. releases, iterations) based on real data as the process is more dynamic [1,46].
8. *Monitoring and controlling:* Agile results in more efficient ways of monitoring and controlling of the development process.
8.1. Agile methods complement the stage-gate management model with powerful tools for micro-planning, day-to-day work control, reporting on progress [15].
8.2. Agile methods result in improved resource management, product functionality, on-time delivery and quality control due to real-time project tracking [15].
8.3. Agile leads to increased process control, transparency and quality through continuous integration and small manageable tasks [1,31].
8.4. Agile allows for continuous improvement and refinement of the development process (e.g. at the end of each iteration, release) [1,31].
8.5. Agile allows to fail cheaply as each fail is detected on every small step (e.g. end of iteration, daily meetings) [1,46].
9. *Adaptability:* Agile is adaptable to different environments as well as other software development approaches.
9.1. Agile thrives in radically different environments (e.g. from hierarchical structure to little or no central control, customer involvement and physical setting varied greatly) [15].
9.2. Agile methods are adaptable and it is possible to combine agile project management with overall traditional principles (e.g. RD approaches) [15].
10. *Decreased effort:* Agile leads to much more efficient development process due to reduction of effort of work, design etc.
10.1. Agile decreases effort related to changing requirements as the process is flexible to incorporate changes in requirements at a later stage with less impact on the project (e.g. features and requirements that are planned in detail are only these to be implemented in a specific cycle) [15].
10.2. Agile helps to decrease effort for redesign due to very soon architectural design decisions verification (e.g. frequent feedback, software demos) since the later changes are incorporated, the more effort is generated [31].
10.3. Agile decreases effort of overall work due to reduction of the waste of not used work (requirements documented, components implemented) as it is reduced as small packages started are always implemented [25,31].
10.4. Agile decreases effort of rework caused by faults as testing priorities are made more clear due to prioritized features, and that testers as well as designers work closely together [31].
10.5. Agile decreases the effort of extensive documenting since small teams with people having different roles only require small amounts of documentation as it is replaced with direct communication facilitating learning and understanding for each other [31].

C Overview of Limitations Derived from Literature

The following list shows the limitation categories surveyed (categories 11 to 17) and the items within the categories.

11. *Employee Dissatisfaction:* Agile results in decreased motivation and increased stress level of the employees.
11.1. Agile decreases developers motivation as they are distracted and feel uncomfortable when they have to report progress of work (e.g. work that has not been tested in an integrated fashion) as a matter of frequent meetings [1].
11.2. Agile causes increase of stress level of developers due to frequent deadlines and milestones [31].

12. *Professional Skills Specific Demands:* Agile is very demanding when considering professional skills in order to be successful.
    12.1. Agile requires a comparable level of qualifications of the developers, otherwise the development will be ineffective (e.g. delays in development of modules) [15,31].
    12.2. Agile requires high qualifications from the all team members along with the leader to be effective (e.g. poorly conducted meetings, lack of knowledge of prioritization techniques etc.) [35,1,31,33].
    12.3. Agile requires sufficiently trained managers (on agile development) as the empowerment of engineers makes managers afraid initially (the teams should be self-organized rather than centrally managed) [35,1,31,46].
    12.4. Agile makes team members less interchangeable (what has consequences for how projects are managed) and more difficult to describe and identify (agile teams are not only build of skilled developers, members complement each other, synergy is created) [15,1].
13. *Limitation to Specific Product Domains:* Agile is limited to safety-critical and legacy systems.
    13.1. Agile provides limited support for developing safety-critical software (e.g. military or healthcare domain where the software needs to be of the highest quality possible). [33].
    13.2. Agile provides limited support for development of legacy systems [1].
14. *Scalability:* Agile is poorly scalable and not applicable to large development teams.
    14.1. Agile is poorly scalable and thus provides limited support for distributed development environments [33].
    14.2. Agile is not applicable to large teams and thus limited for development large-scale development [35,25,1,31,33,46].
15. *Increased Testing Effort:* Agile results in increased testing effort. Continuous testing demands are very high.
    15.1. Agile requires much effort to compensate shortage of project testing due to test coverage reduction within projects as a result of lack of independent testing [35,31].
    15.2. Agile requires much effort to produce big amounts of testing documentation [31].
    15.3. Agile requires much effort for continuous testing as creating an integrated test environment is hard for different platforms and system dependencies [31,33].
16. *Increased Process Effort:* Agile methods lead to increased effort of project management, software maintenance, packaging etc.
    16.1. Agile significantly increases the effort of configuration management due to the need of coordination of the high number of internal releases [31].
    16.2. Agile methods cause increase of effort of packaging as it is still viewed from a technical point of view, but not from a commercial point of view [31].
    16.3. Agile increases the development effort and delays as dependencies rooted in implementation details are hard to identify and not covered in the anatomy plan (lack of documentation) [31].
    16.4. Agile increases the management overhead due to a high number of teams requiring much coordination and communication between (issues with coordination of work, priority lists between agile and non-agile teams due to different approaches, phases, methodology etc.) [31].
    16.5. Agile increases the maintenance effort as many different versions have to be supported and test environments for different versions have to be created (e.g. as result of the ability of releasing many releases to market) [31].
17. *Lead Time and Efficiency:* Agile is often inefficient. The inefficiency is mainly related to development, communication overhead and delays.
    17.1. Agile is inefficient as there is a handover from requirements to design due to complex decision processes as well as some decisions are ineffective due to social values embraced by agile as they are contrary to those that group members desire [35,31].
    17.2. Agile results in delays of getting functional testing running due to workload of frequent integration [31].
    17.3. Agile provides inefficient way of reporting progress on frequent meetings (e.g. if team is inexperienced or more than 8-10 people) [1].
    17.4. Agile results in reduced efficiency of development as close developers cooperation leads to exhaustion [15,31].

17.5. Agile makes the inter-team communication inefficient (e.g. especially when teams follow different software development process and have to co-operate within the same project) [1].