Available online at www.sciencedirect.com
ScienceDirect
Information and Software Technology 50 (2008) 833–859
INFORMATION AND SOFTWARE TECHNOLOGY
www.elsevier.com/locate/infsof

# Empirical studies of agile software development: A systematic review

Tore Dybå *, Torgeir Dingsøyr

*SINTEF ICT, S.P. Andersensv. 15B, NO-7465 Trondheim, Norway*

Received 22 October 2007; received in revised form 22 January 2008; accepted 24 January 2008
Available online 2 February 2008

**Abstract**

Agile software development represents a major departure from traditional, plan-based approaches to software engineering. A systematic review of empirical studies of agile software development up to and including 2005 was conducted. The search strategy identified 1996 studies, of which 36 were identified as empirical studies. The studies were grouped into four themes: introduction and adoption, human and social factors, perceptions on agile methods, and comparative studies. The review investigates what is currently known about the benefits and limitations of, and the strength of evidence for, agile methods. Implications for research and practice are presented. The main implication for research is a need for more and better empirical studies of agile software development within a common research agenda. For the industrial readership, the review provides a map of findings, according to topic, that can be compared for relevance to their own settings and situations.


*Keywords:* Empirical software engineering; Evidence-based software engineering; Systematic review; Research synthesis; Agile software development; XP; Extreme programming; Scrum

**Contents**



* Corresponding author. Tel.: +47 73 59 29 47; fax: +47 73 59 29 77.
*E-mail addresses:* tore.dyba@sintef.no (T. Dybå), torgeir.dingsoyr@sintef.no (T. Dingsøyr).

0950-5849/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2008.01.006

## 1. Introduction

The issue of how software development should be organized in order to deliver faster, better, and cheaper solutions has been discussed in software engineering circles for decades. Many remedies for improvement have been suggested, from the standardization and measurement of the software process to a multitude of concrete tools, techniques, and practices.

Recently, many of the suggestions for improvement have come from experienced practitioners, who have labelled their methods *agile software development*. This movement has had a huge impact on how software is developed worldwide. However, though there are many agile methods, little is known about how these methods are carried out in practice and what their effects are.

This systematic review seeks to evaluate, synthesize, and present the empirical findings on agile software development to date, and provide an overview of topics researched, their findings, strength of the findings, and implications for research and practice. We believe this overview will be important for practitioners who want to stay up to date with the state of research, as well as for researchers who want to identify topic areas that have been researched or where research is lacking. This review will also help the scientific community that works with agile development to build a common understanding of the challenges that must be faced when investigating the effectiveness of agile methods. The results of such investigation will be relevant to the software industry.

The article is organized as follows: In Section 2, we give an overview of agile software development, identify the theoretical roots, and existing reviews. Section 3 describes the methods used for this review. Section 4 reports the findings of the review after first presenting an overview of the studies, the research methods used, the quality of the methodology, and a description of the studies in four main thematic groups. Section 5 discusses benefits and limitations, strength of evidence, and implications for research and practice. Section 6 concludes and provides recommendations for further research on agile software development.

## 2. Background – agile software development

We first describe the field of agile development, core ideas, how this field relates to other disciplines, and summarize the critique of agile development. We then summarize previous reviews of the agile literature, justify the need for this review, and state the research questions that motivated the review.

### 2.1. The field of agile software development

Methods for agile software development constitute a set of practices for software development that have been created by experienced practitioners [68]. These methods can be seen as a reaction to plan-based or traditional methods, which emphasize "a rationalized, engineering-based approach" [21,47] in which it is claimed that problems are fully specifiable and that optimal and predictable solu-

Table 1
Description of main agile development methods, with key references

| Agile method | Description | Reference |
|---|---|---|
| Crystal methodologies | A family of methods for co-located teams of different sizes and criticality: Clear, Yellow, Orange, Red, Blue. The most agile method, Crystal Clear, focuses on communication in small teams developing software that is not life-critical. Clear development has seven characteristics: frequent delivery, reflective improvement, osmotic communication, personal safety, focus, easy access to expert users, and requirements for the technical environment | [16] |
| Dynamic software development method (DSDM) | Divides projects in three phases: pre-project, project life-cycle, and post project. Nine principles underlie DSDM: user involvement, empowering the project team, frequent delivery, addressing current business needs, iterative and incremental development, allow for reversing changes, high-level scope being fixed before project starts, testing throughout the lifecycle, and efficient and effective communication | [60] |
| Feature-driven development | Combines model-driven and agile development with emphasis on initial object model, division of work in features, and iterative design for each feature. Claims to be suitable for the development of critical systems. An iteration of a feature consists of two phases: design and development | [50] |
| Lean software development | An adaptation of principles from lean production and, in particular, the Toyota production system to software development. Consists of seven principles: eliminate waste, amplify learning, decide as late as possible, deliver as fast as possible, empower the team, build integrity, and see the whole | [52] |
| Scrum | Focuses on project management in situations where it is difficult to plan ahead, with mechanisms for "empirical process control"; where feedback loops constitute the core element. Software is developed by a self-organizing team in increments (called "sprints"), starting with planning and ending with a review. Features to be implemented in the system are registered in a backlog. Then, the product owner decides which backlog items should be developed in the following sprint. Team members coordinate their work in a daily stand-up meeting. One team member, the scrum master, is in charge of solving problems that stop the team from working effectively | [56] |
| Extreme programming (XP; XP2) | Focuses on best practice for development. Consists of twelve practices: the planning game, small releases, metaphor, simple design, testing, refactoring, pair programming, collective ownership, continuous integration, 40-h week, on-site customers, and coding standards. The revised "XP2" consists of the following "primary practices": sit together, whole team, informative workspace, energized work, pair programming, stories, weekly cycle, quarterly cycle, slack, 10-minute build, continuous integration, test-first programming, and incremental design. There are also 11 "corollary practices" | [9,10] |

tions exist for every problem. The "traditionalists" are said to advocate extensive planning, codified processes, and rigorous reuse to make development an efficient and predictable activity [11].

By contrast, agile processes address the challenge of an unpredictable world by relying on "people and their creativity rather than on processes" [21,47].

Ericksson et al. [27] define agility as follows:

> agility means to strip away as much of the heaviness, commonly associated with the traditional software-development methodologies, as possible to promote quick response to changing environments, changes in user requirements, accelerated project deadlines and the like. (p. 89)

Williams and Cockburn [66] state that agile development is "about feedback and change", that agile methodologies are developed to "embrace, rather than reject, higher rates of change".

In 2001, the "agile manifesto" was written by the practitioners who proposed many of the agile development methods. The manifesto states that agile development should focus on four core values[1]:

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

In an article that describes the history of iterative and incremental development, Larman and Basili [40] identify Dynamic Systems Development Method (DSDM) [60] as the first agile method, followed by extreme programming (XP) [9], which originated from the Chrysler C3 project in 1996 [5]. In 1998, the word "agile" was used in combination with "software process" for the first time [6]. Several further methods followed, including the Crystal family of methods [16], EVO [28], Feature-Driven Development [50], Lean Development [52] and Scrum [56]. In 2004, a new version of XP appeared [10]. See Table 1 for an overview of the most referenced agile development methods, and Table 2 for a comparison of traditional and agile development.

Many have tried to explain the core ideas in agile software development, some by examining similar trends in other disciplines. Conboy and Fitzgerald [19], for example, describe agility as what is known in other fields as "flexibility" and "leanness". They refer to several sources of inspiration, primarily:

- Agile manufacturing, which was introduced by researchers from Lehigh University in an attempt for the USA to regain its competitive position in manufacturing. Key concepts in

[1] http://agilemanifesto.org

Table 2
Main differences between traditional development and agile development [47]

| | Traditional development | Agile development |
|---|---|---|
| Fundamental assumption | Systems are fully specifiable, predictable, and are built through meticulous and extensive planning | High-quality adaptive software is developed by small teams using the principles of continuous design improvement and testing based on rapid feedback and change |
| Management style | Command and control | Leadership and collaboration |
| Knowledge management | Explicit | Tacit |
| Communication | Formal | Informal |
| Development model | Life-cycle model (waterfall, spiral or some variation) | The evolutionary-delivery model |
| Desired organizational form/structure | Mechanistic (bureaucratic with high formalization), aimed at large organizations | Organic (flexible and participative encouraging cooperative social action), aimed at small and medium-sized organizations |
| Quality control | Heavy planning and strict control. Late, heavy testing | Continuous control of requirements, design and solutions. Continuous testing |

agile manufacturing are integrating customer-supplier relationships, managing change, uncertainty, complexity, utilizing human resources, and information [30,55].

- Lean development [67], which is rooted in the Toyota Production System [49] from the 1950s. Some of the core ideas in this system were to eliminate waste, achieve quality first time, and focus on problem solving.

Meso and Jain [44] have compared ideas in agile development to those in Complex Adaptive Systems by providing a theoretical lens for understanding how agile development can be used in volatile business environments. Turk et al. [64] have clarified the assumptions that underlie processes of agile development and also identifies the limitations that may arise from these assumptions. In the literature, we also find articles that trace the roots of agile development to the Soft Systems Methodology of Peter Checkland [14], New product development [63] and Ackoff's interactive planning [4].

Nerur and Balijepally [46] compare agile development to maturing design ideas in architectural design and strategic management: "the new design metaphor incorporates learning and acknowledges the connectedness of knowing and doing (thought and action), the interwoven nature of means and ends, and the need to reconcile multiple world-views" (p. 81).

However, agile development methods have also been criticized by some practitioners and academics, mainly focusing on five aspects:

1. Agile development is nothing new; such practices have been in place in software development since the 1960s [43].
2. The lack of focus on architecture is bound to engender suboptimal design-decisions [42,61].
3. There is little scientific support for many of the claims made by the agile community [42].
4. The practices in XP are rarely applicable, and are rarely applied by the book [34].
5. Agile development methods are suitable for small teams, but for larger projects, other processes are more appropriate [17].

It has also been suggested that the social values embraced by extreme programming makes agile teams make ineffective decisions, which are contrary to those that the group members desire [41].

### 2.2. Summary of previous reviews

Introductions to and overviews of agile development are given by Abrahamsson et al. [2], Cohen et al. [17], and Erickson et al. [27]. These three reports describe the state of the art and state of the practice in terms of characteristics of the various agile methods and lessons learned from applying such methods in industry. We summarize each of these previous overviews briefly.

The first review of the existing literature on agile software development was done in a technical report published by Abrahamsson et al. at VTT in 2002 [2]. The report discusses the concept of agile development, presents processes, roles, practices, and experience with 10 agile development methods, and compares the methods with respect to the phases that they support and the level of competence that they require. Only DSDM and the Rational Unified Process [38] were found to give full coverage to all phases of development, while Scrum mainly covers aspects related to project management. Abrahamsson et al. found anecdotal evidence that agile methods are "effective and suitable for many situations and environments", but state that very few empirically validated studies support these claims. The report was followed by a comparative analysis of nine agile methods in 2003 [3], where it is stated that empirical support for the suggested methods remains scarce.

Cohen et al.'s review published in 2004 [17] emphasizes the history of agile development, shows some of the roots to other disciplines, and, in particular, discusses relations between agile development and the Capability Maturity Model (CMM) [51]. They further describe the state of the art with respect to the main agile methods and their characteristics. They also describe the state of the practice, which resulted from an online discussion between 18 prac-

titioners, many of whom were involved in defining the various agile development methods. They discuss issues such as the introduction of, and project management in, agile development. They also present experiments and surveys, and seven case studies of agile development. The authors believe that agile methods will be consolidated in the future, just as object-oriented methods were consolidated. Further, they do not believe that agile methods will rule out traditional methods. Rather, they believe that agile and traditional methods will have a symbiotic relationship, in which factors such as the number of people working on a project, application domain, criticality, and innovativeness will determine which process to select.

In 2005, Erickson et al. [27] described the state of research on XP, agile software development, and agile modelling. With respect to XP, they found a small number of case studies and experience reports that promote the success of XP. The XP practice of pair programming is supported by a more well-established stream of research, and there are some studies on iterative development. Erickson et al. recommend that the other core practices in XP be studied separately in order to identify what practices are working (for recent studies of practices, see [22,26]). Further, they see challenges with matching agile software development methods with standards such as ISO, and they argue that this is an area that needs further research. There was much less research on agile modelling than on XP.

### 2.3. Objectives of this review

In a short time, agile development has attracted huge interest from the software industry. A survey in the USA and Europe reveals that 14% of companies are using agile methods, and that 49% of the companies that are aware of agile methods are interested in adopting them [1]. In just six years, the Agile[2] conference has grown to attract a larger attendance than most conferences in software engineering.

Rajlich [53] describes agile development as a paradigm shift in software engineering, which has emerged from independent sources: studies of software life cycles and iterative development. "The new paradigm brings a host of new topics into the forefront of software engineering research. These topics have been neglected in the past by researchers inspired by the old paradigm, and therefore there is a backlog of research problems to be solved." (p. 70).

No systematic review of agile software development research has previously been published. The existing reviews that were presented in the previous section only partially cover the empirical studies that exist today. Further, the previous reviews do not include any assessment of the quality of the published studies, as in this systematic review.

This means that practitioners and researchers have to rely on practitioner books in order to get an overview. We hope that this article will be useful for both groups, and that it will make clear which claims on agile software development are supported by scientific studies.

The objective of the review is to answer the following research questions:

1. What is currently known about the benefits and limitations of agile software development?
2. What is the strength of the evidence in support of these findings?
3. What are the implications of these studies for the software industry and the research community?

In addition to producing substantive findings regarding agile software development, the review also aims to advance methodology for integrating diverse study types, including qualitative research, within systematic reviews of software engineering interventions. The result of this work has been reported separately [23], and is not further described here.

## 3. Review method

Informed by the established method of systematic review [31,35,36], we undertook the review in distinct stages: the development of review protocol, the identification of inclusion and exclusion criteria, a search for relevant studies, critical appraisal, data extraction, and synthesis. In the rest of this section, we describe the details of these stages and the methods used.

### 3.1. Protocol development

We developed a protocol for the systematic review by following the guidelines, procedures, and policies of the Campbell Collaboration,[3] the Cochrane Handbook for Systematic Reviews of Interventions [31], the University of York's Centre for Reviews and Dissemination's guidance for those carrying out or commissioning reviews [35], and consultation with software engineering specialists on the topic and methods. This protocol specified the research questions, search strategy, inclusion, exclusion and quality criteria, data extraction, and methods of synthesis.

### 3.2. Inclusion and exclusion criteria

Studies were eligible for inclusion in the review if they presented empirical data on agile software development and passed the minimum quality threshold (see Section 3.5). Studies of both students and professional software developers were included. Inclusion of studies was not restricted to any specific type of intervention or outcome measure. The systematic review included qualitative and quantitative research studies, published up to and including 2005. Only studies written in English were included.

[2] www.agile200X.org

[3] www.campbellcollaboration.org

Studies were excluded if their focus, or main focus, was not agile software development or if they did not present empirical data. Furthermore, as our research questions are concerned with agile development as a whole, and its underlying assumptions, studies that focused on single techniques or practices, such as pair programming, unit testing, or refactoring, were excluded. In addition to agile methods in general, we included the following specific methods: XP, Scrum, Crystal, DSDM, FDD, and Lean.

Finally, given that our focus was on empirical research, "lessons learned" papers (papers without a research question and research design) and papers merely based on expert opinion were also excluded.

### 3.3. Data sources and search strategy

The search strategy included electronic databases and hand searches of conference proceedings. The following electronic databases were searched:

- ACM Digital Library
- Compendex
- IEEE Xplore
- ISI Web of Science
- Kluwer Online
- ScienceDirect – Elsevier
- SpringerLink
- Wiley Inter Science Journal Finder

In addition, we hand-searched all volumes of the following conference proceedings for research papers:

- XP
- XP/Agile Universe
- Agile Development Conference

Fig. 1 shows the systematic review process and the number of papers identified at each stage. In stage 1, the titles, abstracts, and keywords of the articles in the included electronic databases and conference proceedings were searched using the following search terms:

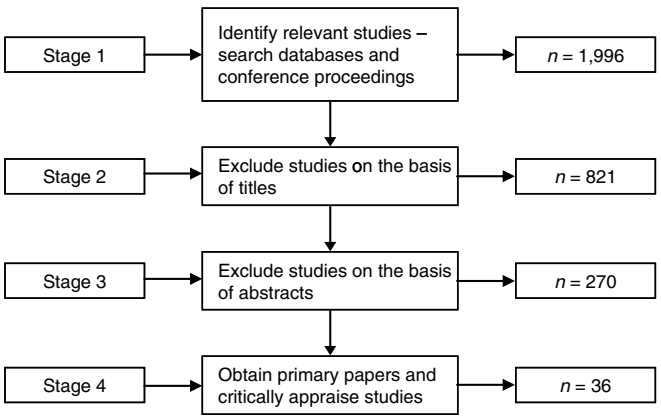


Fig. 1. Stages of the study selection process.

(1) agile AND software
(2) extreme programming
(3) xp AND software
(4) scrum AND software
(5) crystal AND software AND (clear OR orange OR red OR blue)
(6) dsdm AND software
(7) fdd AND software
(8) feature AND driven AND development AND software
(9) lean AND software AND development

All these search terms for agile articles were combined by using the Boolean "OR" operator, which entails that an article only had to include any one of the terms to be retrieved. That is, we searched:

1 OR 2 OR 3 OR 4 OR 5 OR 6 OR 7 OR 8 OR 9

Excluded from the search were editorials, prefaces, article summaries, interviews, news, reviews, correspondence, discussions, comments, reader's letters and summaries of tutorials, workshops, panels, and poster sessions. This search strategy resulted in a total of 2946 "hits" that included 1996 unduplicated citations.

### 3.4. Citation management, retrieval, and inclusion decisions

Relevant citations from stage 1 ($n = 1996$) were entered into and sorted with the aid of EndNote. They were then imported to Excel, where we recorded the source of each citation, our retrieval decision, retrieval status, and eligibility decision. For each subsequent stage, separate EndNote databases and Excel sheets were established.

At stage 2, both authors sat together and went through the titles of all studies that resulted from stage 1, to determine their relevance to the systematic review. At this stage, we excluded studies that were clearly not about agile software development, independently of whether they were empirical or not. As an example, because our search strategy included the term "xp and software", we got several "hits" on articles about Microsoft's Windows XP operating system. In addition, because we used the term "agile and software", we got several hits on articles related to agile manufacturing. Articles with titles that indicated clearly that the articles were outside the scope of this systematic review were excluded. However, titles are not always clear indicators of what an article is about. Some authors' use of "clever" or witty titles can sometimes obscure the actual content of an article. In such cases, the articles were included for review in the next stage. At this stage, 1175 articles were excluded.

At stage 3, studies were excluded if their focus, or main focus, was not agile software development or if they did not present empirical data. However, we found that abstracts were of variable quality; some abstracts were

missing, poor, and/or misleading, and several gave little indication of what was in the full article. In particular, it was not always obvious whether a study was, indeed, an empirical one. Therefore, at this stage, we included all studies that indicated some form of experience with agile development. If it was unclear from the title, abstract, and keywords whether a study conformed to the screening criteria, it was included for a detailed quality assessment (see below).

At this stage, we divided the abstracts among ourselves and a third researcher in such a way that each abstract was reviewed by two researchers independently of each other. For the 821 abstracts assessed, the number of observed agreements was 738 (89.9%). We also computed the Kappa coefficient of agreement, which corrects for chance agreement [18]. The Kappa coefficient for stage 3 assessments was 0.78, which is characterized as "substantial agreement" by Landis and Koch [39]. All disagreements were resolved by discussion that included all three researchers, before proceeding to the next stage. As a result of this discussion, another 551 articles were excluded at this stage, which left 270 articles for the detailed quality assessment.

### 3.5. Quality assessment

Each of the 270 studies that remained after stage 3 was assessed independently by both authors, according to 11 criteria. These criteria were informed by those proposed for the Critical Appraisal Skills Programme (CASP)[4] (in particular, those for assessing the quality of qualitative research [29]) and by principles of good practice for conducting empirical research in software engineering [37].

The 11 criteria covered three main issues pertaining to quality that need to be considered when appraising the studies identified in the review (see Appendix B):

- *Rigour*. Has a thorough and appropriate approach been applied to key research methods in the study?
- *Credibility*. Are the findings well-presented and meaningful?
- *Relevance*. How useful are the findings to the software industry and the research community?

We included three screening criteria that were related to the quality of the *reporting* of a study's rationale, aims, and context. Thus, each study was assessed according to whether:

1. The study reported empirical research or whether it was merely a "lessons learned" report based on expert opinion.
2. The aims and objectives were clearly reported (including a rationale for why the study was undertaken).
3. There was an adequate description of the context in which the research was carried out.

The first of these three criteria represents the minimum quality threshold of the review and was used to exclude non-empirical research papers (see Appendix B). As part of this screening process, any single-technique or single-practice papers were also identified and excluded.

Five criteria were related to the *rigour* of the research methods employed to establish the validity of data collection tools and the analysis methods, and hence the trustworthiness of the findings. Consequently, each study was assessed according to whether:

4. The research design was appropriate to address the aims of the research.
5. There was an adequate description of the sample used and the methods for identifying and recruiting the sample.
6. Any control groups were used to compare treatments.
7. Appropriate data collection methods were used and described.
8. There was an adequate description of the methods used to analyze data and whether appropriate methods for ensuring the data analysis were grounded in the data.

In addition, two criteria were related to the assessment of the *credibility* of the study methods for ensuring that the findings are valid and meaningful. In relation to this, we judged the studies according to whether:

9. The relationship between the researcher and participants was considered to an adequate degree.
10. The study provided clearly stated findings with credible results and justified conclusions.

The final criterion was related to the assessment of the *relevance* of the study for the software industry at large and the research community. Thus, we judged the studies according to whether:

11. They provided value for research or practice.

Taken together, these 11 criteria provided a measure of the extent to which we could be confident that a particular study's findings could make a valuable contribution to the review. Each of the 11 criteria was graded on a dichotomous ("yes" or "no") scale. Again, only criterion 1 was used as the basis for including or excluding a study.

Of the 270 articles assessed for quality, the number of observed agreements regarding inclusion/exclusion based on the screening criterion was 255 (94.4%). The corresponding Kappa coefficient was 0.79. Again, all disagreements were resolved by discussion that included all three researchers. At this stage, another 234 lessons-learned or single-practice articles were excluded, leaving 33 primary and 3 secondary studies

[4] www.phru.nhs.uk/Pages/PHD/CASP.htm

Table 3
Quality criteria

| |
|---|
| 1. Is the paper based on research (or is it merely a "lessons learned" report based on expert opinion)? |
| 2. Is there a clear statement of the aims of the research? |
| 3. Is there an adequate description of the context in which the research was carried out? |
| 4. Was the research design appropriate to address the aims of the research? |
| 5. Was the recruitment strategy appropriate to the aims of the research? |
| 6. Was there a control group with which to compare treatments? |
| 7. Was the data collected in a way that addressed the research issue? |
| 8. Was the data analysis sufficiently rigorous? |
| 9. Has the relationship between researcher and participants been considered to an adequate degree? |
| 10. Is there a clear statement of findings? |
| 11. Is the study of value for research or practice? |

for data extraction and synthesis. A summary of the quality assessment criteria for these studies is presented in Table 3.

### 3.6. Data extraction

During this stage, data was extracted from each of the 33 primary studies included in this systematic review according to a predefined extraction form (see Appendix C). This form enabled us to record full details of the articles under review and to be specific about how each of them addressed our research questions.

When we piloted the extraction process we found that extracting data was hindered by the way some of the primary studies were reported. Due to this, we also found that we differed too much in what we actually extracted for independent extraction to be meaningful. As a consequence, all data from all primary studies were extracted by both authors in consensus meetings.

The aims, settings, research methods descriptions, findings, and conclusions, as reported by the authors of the primary studies, were copied verbatim into NVivo, from QSR Software,[5] a specialist software package for undertaking the qualitative analysis of textual data.

### 3.7. Synthesis of findings

Meta-ethnographic methods were used to synthesize the data extracted from the primary studies [48]. The first stage of the synthesis was to identify the main concepts from each primary study, using the original author's terms. The key concepts were then organized in tabular form to enable comparison across studies and the reciprocal translation of findings into higher-order interpretations. This process is analogous to the method of constant comparison used in qualitative data analysis [45,62]. When we identified differences in findings, we investigated whether these could be explained by the differences in methods or characteristics of the study setting.

In a meta-ethnographic synthesis, studies can relate to one another in one of three ways: they may be directly comparable as reciprocal translations; they may stand in opposition to one another as refutational translations; or taken together they may represent a line of argument [13]. Table 4 shows Noblit and Hare's seven-step process for conducting a meta-ethnography.

This process of reciprocal and refutational translation and synthesis of studies achieved three things with respect to answering our overarching question about the benefits and limitations of agile software development. First, it identified a set of higher-order interpretations, or themes, which recurred across studies. Second, it documented that agile software development contains both positive and negative dimensions. Finally, it highlighted gaps in the evidence about the applicability of agile methods to software development.

## 4. Results

We identified 36 empirical studies on agile software development. Thirty-three are primary studies (S1-S33) and three are secondary studies (S34-S36); see Appendix A. In what follows, we discuss the primary studies. These cover a range of research topics, were done with a multitude of research methods, and were performed in settings that ranged from professional projects to university courses. Key data, along with a description of the domain in which each primary study was conducted, is presented in Appendix D.

We categorized the studies into four main groups: (1) introduction and adoption, (2) human and social factors, (3) customer and developer perceptions, and (4) comparative studies. Three studies did not fit into any of these categories. They provide baseline data on various aspects of agile development (S1, S8, S11).

We now describe characteristics of the studies, describe the research methods applied, and assess the quality of the studies. Then, we present the studies included in the four categories mentioned above.

### 4.1. Overview of studies

With respect to the kinds of agile method that have been studied, we see from Table 5 that 25 (76%) of the studies in this review were done on XP. Studies on agility in general come next, with five (15%) of the studies. Scrum and Lean

Table 4
Seven phases of meta-ethnography (Noblit and Hare [48])

| |
|---|
| 1. Getting started |
| 2. Deciding what is relevant to the initial interest |
| 3. Reading the studies |
| 4. Determining how the studies are related |
| 5. Translating the studies into one another |
| 6. Synthesizing translations |
| 7. Expressing the synthesis |

[5] See http://www.qsrinternational.com/

Table 5
Studies after type of agile method used in the study

| Agile method | Number | Percent |
|---|---|---|
| XP | 25 | 76 |
| General[a] | 5 | 15 |
| Scrum | 1 | 3 |
| Lean software development | 1 | 3 |
| Other[b] | 1 | 3 |
| Total | 33 | 100 |

[a] "General" refers to studies on agility in general.
[b] "Other" refers to a company-internal agile method.

Software Development were studied in only one empirical research article each.

If we look at the level of experience of the employees who perform agile development in the reviewed studies (Appendix D), we see that 24 (73%) of the studies that investigated agile projects dealt with employees who are beginners (less than a year of experience in agile development). Four (12%) studies dealt with mature agile development teams (at least one year of experience in agile development). Two studies did not indicate whether it was a beginner or mature team that was studied and for three studies (surveys) this classification was not applicable.

Most studies (24 of 33, 73%) dealt with professional software developers. The remaining nine (27%) were conducted in a university setting. Most projects were of short duration and were completed in small teams.

Table 6 gives an overview of the studies according to publication channel. We see that the conferences XP and Agile Development have the largest number of studies.

Table 6
Distribution of studies after publication channel and occurrence

| Publication channel | Type | Number | Percent |
|---|---|---|---|
| XP 200X | Conference | 5 | 15 |
| Agile Development Conference | Conference | 5 | 15 |
| IEEE Software | Journal | 3 | 9 |
| Profes | Conference | 2 | 6 |
| ASEE FEC | Conference | 1 | 3 |
| Computer Supported Cooperative Work | Journal | 1 | 3 |
| CSMR | Conference | 1 | 3 |
| Empirical Software Engineering | Journal | 1 | 3 |
| ENCBS | Conference | 1 | 3 |
| ENC | Conference | 1 | 3 |
| EuroMicro | Conference | 1 | 3 |
| EuroSPI | Conference | 1 | 3 |
| HICCS | Conference | 1 | 3 |
| HSSE | Conference | 1 | 3 |
| IASTED ICSEA | Conference | 1 | 3 |
| ICSE | Conference | 1 | 3 |
| ISESE | Conference | 1 | 3 |
| ITIICT | Conference | 1 | 3 |
| Journal of database management | Journal | 1 | 3 |
| Metrics | Conference | 1 | 3 |
| Software Quality Journal | Journal | 1 | 3 |
| XP/Agile Universe | Conference | 1 | 3 |
| Total | | 33 | 100 |

Table 7
Studies by research method

| Research method | Number | Percent |
|---|---|---|
| Single-case | 13 | 39 |
| Multiple-case | 11 | 33 |
| Survey | 4 | 12 |
| Experiment | 3 | 9 |
| Mixed | 2 | 6 |
| Total | 33 | 100 |

Most of the studies were published in conferences (26 of 33, 79%), while seven (21%) appeared in scientific journals.

Regarding the year of publication, we found no empirical studies of agile software development prior to 2001. However, from 2001 we found a steady increase of studies with one empirical research study published in 2001, one in 2002, three in 2003, 12 in 2004, and 16 published in 2005.

### 4.2. Research methods

The number and percentage of publications using each research method is listed in Table 7. Information on which studies belong to which category is given in Appendix D. Of the 13 single-case studies, nine were done in projects in industry. The material for the other four studies was taken from projects where students did the development. Interestingly, three of these studies took their data from the same project. Only one of the single-case studies in industry was done on a mature development team.

For the 11 multiple-case studies, all were done in industry, but only three of the studies were on mature teams. The number of cases varied from two to three.

Three of the four surveys were done on employees in software companies, while one was done on students. The three experiments were all done on students, with team sizes ranging from three to 16. For the two mixed-method studies, Melnik and Maurer (S22) reported on a survey amongst students in addition to interviews and notes from discussions. The study by Baskerville et al. (S3), reported on 10 case studies in companies, in combination with findings from group discussions in a "discovery colloquium" that was inspired by principles in action research [8].

### 4.3. Methodological quality

As mentioned in Section 3, we chose to assess each of the primary studies according to 11 quality criteria based on the Critical Appraisal Skills Programme (CASP)[6] and by principles of good practice for conducting empirical research in software engineering (e.g., [37]). A summary of the questions used to assess the quality of these studies is presented in Table 3. The detailed subcriteria are presented in Appendix B. Both authors rated each criterion of each study independently. When discrepancies arose,

[6] www.phru.nhs.uk/casp/casp.htm.

Table 8
Quality assessment

| Study | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Research | Aim | Context | R. design | Sampling | Ctrl. Grp | Data coll. | Data anal | Reflexivity | Findings | Value | |
| S1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 7 |
| S2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 10 |
| S3 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 6 |
| S4 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 9 |
| S5 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 9 |
| S6 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 9 |
| S7 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 9 |
| S8 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 7 |
| S9 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 9 |
| S10 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 8 |
| S11 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 6 |
| S12 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 9 |
| S13 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 7 |
| S14 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 9 |
| S15 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 9 |
| S16 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 8 |
| S17 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 8 |
| S18 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 7 |
| S19 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 8 |
| S20 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 8 |
| S21 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 8 |
| S22 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 9 |
| S23 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 4 |
| S24 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 8 |
| S25 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 6 |
| S26 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 6 |
| S27 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 8 |
| S28 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 10 |
| S29 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 9 |
| S30 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 9 |
| S31 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 6 |
| S32 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 9 |
| S33 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 9 |
| Total | 33 | 31 | 33 | 30 | 8 | 10 | 26 | 25 | 1 | 33 | 33 | |

these were discussed and the study was reread to determine the final scores of each criterion.

Taken together, these 11 criteria provide a measure of the extent to which we can be confident that a particular study's findings can make a valuable contribution to the review. The grading of each of the 11 criteria was done on a dichotomous ("yes" or "no") scale. The results of the quality assessment are shown in Table 8, in which a "1" indicates "yes" (or OK) to the question, while "0" indicates "no" (or not OK).

Because we only included research papers in this review, all included studies were rated as OK on the first screening criterion. However, two of the included studies still did not have a clear statement of the aims of the research. All studies had some form of description of the context in which the research was carried out. For three of the studies, the chosen research design did not seem appropriate to the aims of the research. As many as 25 out of the 33 primary studies did not have a recruitment strategy that seemed appropriate for the aims stated for the research. Ten of the studies included one or more groups with which to compare agile methods. As many as seven and eight studies, respectively, did not adequately describe their data collection and data analysis procedures. In only one study was the recognition of any possibility of researcher bias mentioned.

We frequently found the following: methods were not well described; issues of bias, validity, and reliability were not always addressed; and methods of data collection and analysis were often not explained well. None of the studies got a full score on the quality assessment and only two studies got one negative answer. Twenty-one studies were rated at two or three negative answers, while 10 studies were rated as having four or more negative answers. The highest number of negative answers was seven.

### 4.4. Introduction and adoption of agile development methods

Several studies addressed how agile development methods are introduced and adopted in companies; see Table 9. We characterized these studies as falling into three broad groups: those that discuss introduction and adoption, those that discuss how the development process is changed, and

Table 9
Study aims for studies on the introduction and adoption of agile development methods

| Study | Study aim |
|---|---|
| S2 | Understand what differentiates waterfall and XP development, and examine the impact of knowledge creation on the adoption of XP |
| S9 | Study why and how XP is adopted and used in everyday software production |
| S12 | Study the integration of agile teams into stage-gate software product development |
| S23 | Test the applicability of lean techniques to developing software |
| S29 | Study how an agile process affects collaboration in a software development organization |
| S30 | Study the introducing of a process based on XP in an evolutionary and maintenance-based software development environment |
| S31 | Understand how newcomers practice extreme programming, and how practice is improved over time |

those that discuss how knowledge and projects are managed.

However, some researchers argue that there is nothing new about agile methods. Hilkka et al. (S9) studied two development organizations in Finland, and concluded that XP is "old wine in new bottles". XP "formalizes several habits that appear naturally (...) close customer involvement, short release cycles, cyclical development, and fast response to change requests". In a company-internal development department, the researchers found that:

> the tools and techniques of XP had been employed for more than 10 years and had been applied in a quite systematic fashion, though the company had never made a deliberate decision to use XP. (p. 52)

Another "new economy" company was more aware of developments in the field:

> the XP process had more or less emerged as a novel way of solving time and budget constraints. The developers were aware of XP practices, but did not choose to engage in it by the book. (p. 52)

However, most studies treated agile development as something "new" and that consequently requires introduction and adoption.

#### 4.4.1. Introduction and adoption

Svensson and Höst (S30) present the results of introducing a process based on XP to a large software development company. The process was introduced to a pilot team that worked for eight months. Svensson and Höst concluded that the introduction of the process proved difficult, due to the complexity of the organization. They advise companies that want to introduce agile development methods to assess existing processes, with the following goals in mind: determining what to introduce; clarifying terminology to simplify communication with the rest of the company; avoiding underestimating the effort needed to introduce and adapt XP; and introducing the practice of continuous testing early, because it takes time and effort to introduce this properly.

In contrast, Bahli and Zeid (S2) studied how a Canadian organization shifted from a waterfall process to XP, and found that "even though team members had no prior experience with XP (except one week of training), they found the model easy to use and useful to develop information systems". A development manager described the shift as follows:

> The first week was tough, no one of my guys have a strong experience with XP. But they quickly caught up and we got quite good results. A lot of work is needed to master XP but we are getting there. (p. 8)

The study reports that the development team found using the waterfall model to be an "unpleasant experience", while XP was found to be "beneficial and a good move from management". The XP project was delivered a bit less late (50% time-overrun, versus 60% for the traditional), and at a significantly reduced cost overrun (25%, compared to 50% cost overrun for the traditional project).

Bahli and Zeid claim that the adoption of XP was facilitated by "a high degree of knowledge creation enabled by mutual development of information systems developers and users".

Karlström and Runeson (S29) found that XP teams experienced improved communication, but were perceived by other teams as more isolated. Their study is described in detail below.

#### 4.4.2. Development process

Tessem (S31) set up a project with researchers and students to learn more about how practices in XP work. The project lasted for three weeks and had two deliveries (three planned, but reduced because of "severe underestimation in the beginning"). Six people worked on the project. The experience of the people varied from programming experience only from university courses to people experienced with professional development. The aim of the project was to develop a web application for group work in university courses. Tessem reports on experience with key practices of XP. Pair programming was found to be a "positive experience, enhancing learning and also leading to higher quality". However, three of the programmers also reported it to be "extremely inefficient", "very exhausting", and "a waste of time". Towards the end of the project, single programming was used to a greater extent than pair programming. Tessem suggests that there is a connection between this shift in programming methods and a higher occurrence of problems in

the end. Frequent partner changes are suggested as a way to achieve optimal learning and to increase collective code ownership. Further, the on-site customer role was perceived as "very valuable by all programmers". Tessem also found that test-first programming contributed to "higher quality in code", while the project struggled to get functional tests running.

A study by Svensson and Höst (S29) also provides insight into the development process, but focused primarily on how agile development affects customer collaboration. This study was done in a software development company in Sweden with 250 software developers, who were responsible for over 30 software systems. A modified process was introduced that mainly followed XP. The researchers found that having the customer on-site enabled better collaboration with the customer, because it provided an arena for detailed discussions.

Concepts from lean development were introduced in a large company's information systems department in a study organized by Middleton (S23). The techniques were tried on two two-person teams that were maintaining a financial and management information system. The teams were instructed to change their work practice, so that it involved shorter feedback cycles and completing work before taking on more. In the beginning, many errors were discovered in the work, which led to a time of "frustration and low productivity". One of the teams made fewer errors over time, but the other team continued to make a high number of errors, which also led to an early termination of the study. According to Middleton, this was because one person in the fault-prone team felt overqualified for the work and was not willing to discuss his work with others, and was "unable to produce work without errors". There was no infrastructure in the company to handle this problem. Although the experiment was short and only successful for one team, Middleton claimed that "by moving responsibility for measuring quality from the manger to the workers, a much quicker and more thorough response to defects was obtained".

Hilkka et al. (S9) found that in the cases they studied, XP worked best with experienced developers with domain and tool knowledge. The tools facilitated fast delivery and easy modification of prototypes. In addition, continuous feedback was found to be a key factor for success.

#### 4.4.3. Knowledge and project management

The study by Bahli and Zeid (S2) examined knowledge sharing in an XP project and a traditional project. They found that when the XP model was used, the creation of tacit knowledge improved as a result of frequent contacts:

> Because the XP model's main characteristics are short iterations with small releases and rapid feedback, close user participation, constant communication and coordination and collective ownership, knowledge and the capability to create and utilize knowledge among the development team members are eminent. (p. 4)

Hilkka et al. also underline the importance of skilled team members with solid domain knowledge: "without these kinds of persons, the chosen approach would probably have little possibility to succeed" (S9).

Karlström and Runeson (S12) studied the feasibility of applying agile methods in large software development projects, using stage-gate project management models. They report findings from introductory trials with XP in three companies: ABB, Ericsson Microwave Systems, and Vodaphone Group. They found that the engineers were motivated by the principles of agile development, but that the managers were initially afraid and needed to be trained. They also found that as a result of using agile development, the engineers focused on past and current releases, while the managers focused increasingly on current and future releases. A potential problem was that technical issues were raised too early for management.

In the study by Tessem (S31), the planning game practice of XP was used to estimate the size of work. Estimates made by the project team at the beginning of the project were about one third of what turned out to be correct, which is explained by both the team's lack of estimation experience and coarse user stories. Estimates improved towards the end of the project. Several of the study participants mentioned that during the project, there were not enough discussions on design and architecture.

In a study by Svensson and Höst (S30), the planning game activity was found to have a positive effect on collaboration within the company, because it provided the organization with better insight into the software development process.

### 4.5. Human and social factors

Several studies examined various human and social factors related to agile development; see Table 10. Three broad topics were investigated: the impact of organizational culture, how collaborative work takes place in agile development, and what characterizes agile development teams.

#### 4.5.1. Organizational culture

Robinson and Sharp (S26) found that XP has the ability to thrive in radically different organizational settings in an ethnographically informed study of three companies in the UK. The companies were studied with respect to three factors: organizational type, organizational structure, and physical and temporal settings. These factors are described in Table 11.

Case A was a large multinational bank with an XP team that was "a small part of the bank's software development activities". Case B was a medium-sized company that produces content security software, using only XP. Case C was a small start-up company that had used XP since the beginning to develop web-based intelligent advertisements.

Robinson and Sharp (S26) found that, despite the variations in organization type, structure, and physical setting, XP was working well. They list a number of consequences

Table 10
Study aims for studies on human and social factors

| Study | Study aim |
|---|---|
| S5 | Study how the social behaviour of individuals reflects beliefs and understanding with regard to software development |
| S16 | Evaluate key aspects of XP in a real-world setting as well as contemporary practices of software engineering |
| S24 | Study the characteristics of XP teams |
| S25 | Explore the social side of practices in mature XP teams |
| S26 | Explore the impact of organizational culture on XP practice in mature teams |
| S27 | Gain insight into the culture and community of XP |
| S33 | Explore whether it is possible to identify roles within an extreme programming team that has associated personality characteristics |

Table 11
Differences in organization type, structure, and physical setting for the three cases studied in S26

| | Organization type | Organizational structure | Physical setting |
|---|---|---|---|
| Case A | Hierarchical | XP team collaborated with two customer representatives | Large, open-plan floor with workstations |
| Case B | Collaborative | Internal "on-site" customers, outside testers | Open-plan space with groups of workstations for pair programming, meeting rooms and cubicles |
| Case C | Little or no central control | Minimal organizational structure | Open-plan office with pair programming area |

for development that were generated by the organizational culture.

Case C is further described in another publication (S27). Here, the development team is described as:

> a self-managing, self-organizing community with a culture that emphasized shared responsibility. There was a rhythm to life that enabled people to organize their work tasks in a way that gave them common ownership of the work product and control over how it was achieved. The rhythm was comfortable and relaxed, yet purposeful and productive. (p. 373)

In case C, the authors claim that the organization seemed to behave in an agile fashion. They found no signs of such normal software development artefacts as modelling techniques, requirements documents, or minutes of meetings. The working mode in this company resembles descriptions of communities of practice in the literature on knowledge management [65].

The organizational culture affected how XP was carried out, with respect to behaviour, beliefs, attitudes and values, organizational structure, as well as physical and temporal settings.

#### 4.5.2. Collaborative work

Collaborative work in XP development has been studied from three angles: the role of conversation in collaborative work, how progress is tracked, and how work is standardized.

With respect to conversation, Robinson and Sharp (S25) describe pairing as a process of:

> purposeful talk where they [two developers] discuss, explore, negotiate and progress the task at hand. This talk has a complex structure with identifiable episodes of exploration, creation, fixing & refining, overlaid with explaining, justifying & scrutinising. (p. 102)

Paring is described as intense and stressful, and one pair's conversation would frequently spread to other pairs. Mackenzie and Monk (S16) also emphasize the importance of conversations, claiming that it constitutes "talking code into existence".

With respect to tracking progress, Robinson and Sharp described it as happening on two levels: the daily rhythm and the rhythm oriented around the iteration. Progress was communicated in daily stand-up meetings, and teams would often have ceremonies around releasing code. One team studied by Robinson and Sharp used a toy cow that was tilted to make a 'moo' sound when new code was released. Chong (S5) reports similar findings, stating that "XP makes developing software visually and aurally available".

Studies of collaborative work also find that the work patterns are standardized. Chong (S5) observed that:

> shared understandings manifested themselves in the consistent, uniform patterns of work in which the team members engaged ... the XP framework required that they work at the same time, in the same place, and in largely the same ways. (p. 8)

One practice that standardizes work in XP is the planning game, which is described in use by Mackenzie and Monk (S16):

> the card game knit together in a rule-governed process a very disparate set of work processes and relations involving management, the customer or client and all the members of the software development team. (p. 114)

Mackenzie and Monk claimed that the process spans the usual boundaries between project managers and software developers.

#### 4.5.3. Team characteristics

Robinson and Sharp (S24) claim that agile development teams have faith in their own abilities, show respect and responsibility, establish trust, and preserve the quality of working life. Young et al. (S33) used a technique called "repertory grid analysis" to identify good personality characteristics for members of XP development teams.

Faith in one's own abilities was observed to have two aspects in the study by Robinson and Sharp (S24): believing that the team was capable of achieving the tasks at hand, and understanding what the limitations were. The team received feedback on their beliefs from successfully executing code, from a satisfied customer, and from support and encouragement from each other.

Preserving the quality of working life was observed through constructive discussions in the planning game, taking into account the needs of individuals in pair programming, and adhering to 40-h work-weeks. In addition, one team took regular breaks and identified several ways to relieve developers in hectic periods.

Respect for one's team members and a sense of responsibility were manifested via the way in which work was assigned; active agreement was required. "Individuals clearly felt that they had the respect of their fellow team members and were therefore empowered to take on responsibility in this way".

In the study by Robinson and Sharp (S24), trust was found to be pervasive:

> The nature of the trust relationship here transcends the immediate business of two individuals pairing and is persistent. It also applies across pairs (and sub-teams), with each pair trusting the others to do their part, and it extends beyond the 12 practices. (p. 146)

Young et al. (S33) investigated what personality traits it is beneficial for team members to possess in agile development. They discussed the traits of roles such as team leader, technical lead, architect, good (XP) team member, and bad team member. Good XP team members are described as "analytical, with good interpersonal skills and a passion for extending his knowledge base (and passing this on to others)."

### 4.6. Perceptions on agile methods

Several studies have investigated how agile methods are perceived by different groups. We describe findings from studies that examined the perceptions amongst customers, developers, and university students. Table 12 gives an overview of the aims of these studies.

#### 4.6.1. Customer perceptions

Several aspects of customer perceptions are discussed in the literature on agile development. Some have addressed how satisfied customers are with agile methods, others describe the customer role, and some focus on the collaboration between a customer and the development team.

With respect to the customer's satisfaction with agile development methods, Ilieva et al. (S10) studied the introduction of an agile method based on XP and the Personal Software Process [32]. They state that the customer had constant control over the development process, which was "highly praised by the customer at the project sign-off". In addition, Mann and Maurer (S17) found, in a study on the impact of Scrum on overtime and customer satisfaction, that customers believed that the daily meetings kept them up to date and that planning meetings were helpful to "reduce the confusion about what should be developed". The attitude of the customers was found to change from "one of ambivalence to becoming involved". The customers stated that their satisfaction with the project that was based on XP was greater than with previous projects at the company.

However, Mann and Maurer stress that the customer should be trained in the Scrum process so that they will understand the new expectations that the developers will have of them.

The role of the customer is also the focus in the study by Martin et al. (S19), on three XP projects with on-site customers. In all cases, they found that the customer was under stress and committed working long hours, although

Table 12
Study aims for the perceptions of customers, developers, and students

| Study | Study aim |
|---|---|
| S2 | Understand what differentiates waterfall and XP development, and examine the impact of knowledge creation on the adoption of XP |
| S10 | Compare empirical findings of an agile method based on XP with a baseline, with respect to productivity, defect rates, cost, and schedule deviations |
| S13 | Provide empirical data on customer effort usage, customer perceptions, and project team perceptions of on-site customer in an XP project |
| S17 | Study the impact of Scrum on overtime, as well as on customer and developer experience |
| S18 | Compare the job satisfaction of developers that use XP practices with developers that do not use XP practices |
| S19 | Provide empirical data on the role of on-site customer role in XP |
| S20 | Study the role of the customer in projects using XP and outsourcing |
| S21 | Study student perceptions of agile practices |
| S22 | Study how students perceive agile development methods |

all the customers were supported by an acceptance team, various technical advisors, or senior personnel:

> The existing XP Customer practices appears to be achieving excellent results, but they also appear to be unsustainable, and so constitute a great risk to XP projects, especially in long or high pressure projects (p. 12)

Martin et al. (S20) also studied the role of the customer in outsourced projects, and found that this was challenging because the customer was required to become acclimatized to the different cultures or organizations of the developers.

Koskela and Abrahamsson (S13) analyzed the role of the customer in an XP project and found that most of the time was spent on participating in planning game sessions and acceptance testing, followed by retrospective sessions at the end of release cycles.

#### 4.6.2. Developer perceptions

Mannaro et al. (S18) surveyed the job satisfaction amongst employees in software companies that used XP and companies that did not use agile development methods. One hundred and twenty-two people completed a web-based questionnaire. The bulk of these were from Europe and the United States.

Ninety-five percent of the employees who used XP answered that they would like their company to continue using their current development process, while the number for the employees in companies that did not use agile development methods was 40%. In addition, the employees in the companies that used XP were significantly more willing to use the development process in the future than the employees in companies that did not use XP. Further, Mannaro et al. (S18) claimed that employees who use XP have greater job satisfaction, feel that the job environment is more comfortable, and believe that their productivity is higher. In particular, 73% of the employees who used pair programming claim that this practice speeds up the software development process.

In the study by Ilieva et al. (S10), developers found pair programming to be "a very useful style of working as everyone was strictly conforming to the coding standards". However, the authors also note that working 40 h a week in pairs requires a lot of concentration, and that as a result, the developers became exhausted.

Mann and Maurer (S17) found that the introduction of Scrum led to a reduction of overtime, and all developers recommended the use of Scrum in future projects. The developers were more satisfied with the product, and saw that the Scrum process fostered more customer involvement and communication. One developer said that "the Scrum process is giving me confidence that we are developing the software that the customer wants".

A study by Bahli and Zeid (S2) used the Technology Acceptance Model [54] to study the adoption of XP in a company that develops medical information systems. They found that employees saw XP as easy to use and useful, and that employees intended to use this development process in the future.

#### 4.6.3. Student perceptions

Melnik and Maurer (S21, S22) report on student perceptions of agile development methods in two studies, one from 2002 and one from 2005. They found that 240 students who responded to a survey at the Southern Alberta Institute of Technology and at the University of Calgary in Canada were "very enthusiastic about core agile practices". The findings were consistent across educational programmes.

The students found that working in agile teams helped them to develop professional skills such as communication, commitment, cooperation, and adaptability. Seventy-eight percent of the respondents stated that they believe that XP improves the productivity of small teams. This figure is comparable to the findings of Mannaro et al. (S18) on pair programming and productivity for employees in software companies.

Further, 76% of the respondents believed that using XP improved the quality of code, and 65% would recommend using XP to any company for which they may work in the future. Of those that recommended using XP to their future employers, a large number preferred to work in pairs.

In the 2002 study, Melnik and Maurer (S21) present qualitative findings on perceptions of XP in general, pair programming, test-first design, and the planning game. Most students found pair programming to be helpful, but some expressed concern when the the members of the pair had different levels of competence. One student stated that "There was a huge difference in skill level in my pair, so we weren't very productive when I wasn't driving". In addition, test-first design was difficult for many students. The authors believe that this is because design in itself is very difficult, and writing the tests first forces students to make design decisions early.

### 4.7. Comparative studies

One third (11) of the reviewed primary studies provided some form of comparison of agile development against an alternative; see Table 13. Using our interpretations as a basis, these comparisons can be grouped into four higher-order comparative topics: project management, productivity, product quality, and team characteristics. Non-comparative studies that mention one or more of these issues are also included in this section.

#### 4.7.1. Project management

The management of software projects has long been a matter of interest. Agile methods have reinforced this interest, because many conventional ideas about management are challenged by such methods. Ceschi et al. (S4) found, in their survey of plan-based and agile companies, that agile methods improved the management of the development process as well as relationships with the customer.

Table 13
Study aims for comparative studies

| Study | Study aim |
|---|---|
| S3 | Understand how and why Internet-speed software development differs from traditional software development |
| S4 | Compare agile methods with plan-based methods with respect to project management practices |
| S6 | Compare the effectiveness of incremental and evolutionary process models for technology development projects |
| S7 | Compare differences in resource utilization and efficiency in products developed using sequential, incremental, evolutionary, and extreme programming |
| S10 | Compare empirical findings of an agile method based on XP with a baseline, with respect to productivity, defect rates, cost, and schedule deviations |
| S12 | Study the integration of agile teams into stage-gate software product development |
| S14 | Compare the effectiveness of XP with traditional development, with respect to pre- and post-release quality, programmer productivity, customer satisfaction, and team moral |
| S15 | Compare XP with a traditional approach with respect to quality |
| S18 | Compare the job satisfaction of developers that use XP practices with developers that do not use XP practices |
| S28 | Compare agile and document-driven approaches in managing uncertainty in software development |
| S32 | Compare plan-driven and agile development with respect to team cohesion and product quality |

In particular, they found that companies that use agile methods prefer to organize their processes in more releases and that the managers of such companies are more satisfied with the way they plan their projects than are plan-based companies. Moreover, Baskerville et al. (S3) found that "Internet-speed development" project management differs from that of traditional development in that "Projects do not begin or end, but are an ongoing operation more akin to operations management." (Baskerville et al. (S3), p. 77).

Karlström and Runeson (S12) studied how traditional stage-gate project management could be combined with agile methods. In a case study of three large companies, they found that agile methods give the stage-gate model powerful tools for microplanning, day-to-day work control, and reporting on progress. They also found that they were able to communicate much more effectively when using the working software and face-to-face meetings of agile methods than when using written documents. In turn, the stage-gate model provided the agile methods with a means to coordinate with other development teams and to communicate with marketing and senior management. Their conclusion was that it is feasible to integrate agile methods with stage-gate project management to improve cost control, product functionality, and on-time delivery.

A central concern for agile methods is to attend to the real needs of the customer, which are often not stated explicitly in a more or less complete requirements specification. Thus, Dagnino et al. (S6) compared and contrasted the use of an evolutionary agile approach with a more traditional incremental approach in two different technology development projects. They showed that by planning in detail only the features and requirements to be implemented in a specific cycle, the agile team was more able to incorporate changes in requirements at a later stage with less impact on the project. In addition, by delivering in-progress software to the customer more frequently, the agile team was able to demonstrate business value more quickly and more often than the traditional, iterative team. Combined with continuous feedback by the customer, this lead to a sharp increase in customer satisfaction on the agile project.

Similarly, Ceschi et al. (S4) found that the tighter links between the customer and the development team resulted in agile companies being more satisfied with their customer relationships than plan-based companies. Furthermore, Sillitti et al.'s (S28) survey of project managers found that companies that use agile methods are more customer-centric and flexible than document-driven ones, and that companies that use agile methods seem to have a more satisfactory relationship with the customer.

However, with respect to human resource management, Baskerville et al. (S3) concluded that compared to traditional development, team members of agile teams are less interchangeable, and more difficult to describe and identify.

#### 4.7.2. Productivity

Four studies compared the productivity of agile teams with the productivity of teams using traditional development methods (S7, S10, S14, S32); see Table 14. Ilieva et al. (S10) compared the productivity of two similar projects, one of which used traditional methods and the other of which used XP. They measured the productivity for three iterations of each project. Overall, the results showed a 42% increase in productivity for the agile team. The increase in productivity was largest for the first iteration, while there was virtually no difference in productivity for the last iteration.

The case study by Layman et al. (S14) compared an old release developed with traditional methods with a new

Table 14
Comparisons of productivity

| Study | $\text{Productivity}_{\text{TRAD}}$ | $\text{Productivity}_{\text{AGILE}}$ | Productivity gain (%) |
|---|---|---|---|
| S7 | 3 LOC/h[a] | 13.1 LOC/h | 337 |
| S10 | 3.8 LOC/h | 5.4 LOC/h | 42 |
| S14 | 300 LOC/month | 440 LOC/month | 46 |
| S32 | 157 LOC/engineer[b] | 88 LOC/engineer | −44 |

[a] V-model.
[b] Comparisons were made between two one-semester courses; however, the actual hours worked by the members of the teams were not measured.

release developed with agile methods. The results showed a 46% increase in productivity for the new agile release compared with the old, traditional release. However, the agile team had notably greater domain and programming language expertise and project manager experience, because three of the team members on the new release had previously worked on the old release.

Dalcher et al. (S7) performed an experiment in which fifteen software teams developed comparable software products using four different development approaches (V-model, incremental, evolutionary, and XP). The greatest difference in productivity was between the V-model teams and the XP teams, with the XP teams being, on average, 337% more productive than the V-model teams. However, this productivity gain was due to the XP team delivering 3.5 times more lines of code without delivering more functionality.

Contrary to the studies by Dalcher et al. (S7), Ilieva et al. (S10), and Layman et al. (S14), Wellington et al. (S32) found a 44% decrease in productivity for an XP team compared with a traditional team. Furthermore, Svensson and Höst (S30) found no change in overall productivity when comparing results from before and after the introduction of an agile process. However, they did find evidence that when the agile process was introduced, the team improved their productivity during the first iterations.

In addition, Mannaro et al. (S18) asked their subjects whether the team's productivity had increased significantly as a result of the development process that was used. On a scale from 1 (Strongly Disagree) to 6 (Strongly Agree), the mean for the non-XP developers was 3.78, while the mean for the XP developers was one scale point higher (4.75). Similarly, 78% of Melnik and Maurer's (S22) respondents either believed or believed strongly that using XP improves the productivity of small teams.

#### 4.7.3. Product quality

Several aspects of product quality were examined by the studies in this review. For example, comparing the results for a new release of a project to those for an old release, Layman et al. (S14) found a 65% improvement in pre-release quality and a 35% improvement in post-release quality. Ilieva et al. (S10) found 13% fewer defects reported by the customer or by the quality assurance team in an XP project than in a non-XP project.

In Wellington et al.'s (S32) study, the XP team's code scored consistently better on the quality metrics used than the traditional team. In addition, the quality of the code delivered by the XP team was significantly greater than that delivered by the traditional team. However, both teams agreed that the traditional team had developed a better and much more consistent user interface.

Macias et al. (S15) measured the internal and external quality of the products developed by 10 XP teams and 10 traditional teams. However, in contrast to Layman et al. and Wellington et al., they found no difference in either internal or external quality between the XP teams and the traditional teams.

With respect to product size, the XP model teams in Dalcher et al.'s (S7) study delivered 3.5 times more lines of code than the V-model teams. This is in sharp contrast to Wellington et al.'s (S32) results, which showed that the traditional team delivered 78% more lines of code than the XP team. However, in contrast to both Dalcher et al. and Wellington et al., Macias et al. (S15) found no difference in product size between the XP teams and the traditional teams.

#### 4.7.4. Work practices and job satisfaction

A few studies made qualitative comparisons of social behaviour. Chong (S5), for example, performed an ethnographic study to compare the work routines and work practices of the software developers on an XP team and a non-XP team. Chong's observations suggest that certain features of XP promote greater uniformity in work routine and work practice across individual team members and that, consequently, XP provides a framework for standardizing the work of software development and making it more visible and accessible to the members of a software development team.

An important part of the XP philosophy is to increase overall team cohesion by making everyone in the team responsible for the source code. However, Wellington et al.'s (S32) study of team cohesion and individuals' attachment to the project in XP and non-XP teams yielded equal or higher scores for every aspect of cohesion for the non-XP teams. However, at the same time, the study indicated a lack of cohesion across subteams for the non-XP team (the XP team was not divided into subteams).

The point of departure for Mannaro et al. (S18) was the importance of job satisfaction for the effectiveness of the software development process. Consequently, they performed a survey to compare the job satisfaction of developers that used XP practices with that of developers that did not use them. The results of their study showed that the developers viewed XP practices favourably and indicated that developers who use XP practices are more comfortable with their job environment and more satisfied with their jobs than developers that do not use XP practices.

## 5. Discussion

The present review identified a greater number of studies than did previous reviews. Abrahamsson et al. (S34) wrote in their 2002 review that the existing evidence consists mainly of practitioners' success stories. Cohen et al. (S35) found seven case studies on agile development in their 2004 report, we included none of these in our final set of studies, because they were either lessons learned studies or single-practice studies. Further, Erickson et al.'s (S36) 2004 review found four "case studies and lessons learned reports", none of which we included in our review. This systematic review shows that there are many more empiri-

cal studies on agile development methods in general than have previously been acknowledged. In contrast to the previous reviews, this review used an explicit search strategy combined with explicit inclusion and exclusion criteria.

We now address our research questions, starting by discussing what we found regarding the benefits and limitations of agile software development. The second subsection discusses the strength of evidence of these findings, while the third subsection discusses the implications of the findings for research and practice. Finally, we discuss the limitations of this systematic review.

### 5.1. Benefits and limitations of agile development

The studies that address the introduction and adoption of agile methods do not provide a unified view of current practice, but offer a broad picture of experience and some contradictory findings. XP was found to be difficult to introduce in a complex organization, yet seemingly easy in other types of organizations. This is consistent with earlier findings that suggest that agile development methods are more suitable for small teams than for larger projects [17]. It is likely that the ease with which XP can be introduced will depend on how interwoven software development is with the other functions of the organization. Most studies reported that agile development practices are easy to adopt and work well. Benefits were reported in the following areas: customer collaboration, work processes for handling defects, learning in pair programming, thinking ahead for management, focusing on current work for engineers, and estimation. With respect to limitations, the lean development technique did not work well for one of the teams trying it out, pair programming was seen as inefficient, and some claimed that XP works best with experienced development teams. A further limitation that was reported by one of the studies, which has also been repeatedly mentioned in the literature [42,61], was the lack of attention to design and architectural issues.

A recurring theme in studies on agile development is what we have called human and social factors and how these factors affect, and are affected by, agile development methods. A benefit of XP was that it thrived in radically different environments; in organizations that varied from having a hierarchical structure to little or no central control. In addition, customer involvement and physical settings varied greatly for the successful XP teams studied. It seems to be possible to adopt XP in various organizational settings. Further, conversation, standardization, and the tracking of progress have been studied and are described as mechanisms for creating awareness within teams and organizations. In addition, studies of XP indicate that successful teams manage to balance a high level of individual autonomy with a high level of team autonomy and corporate responsibility. They have faith in their own abilities and preserve the quality of their working lives. Good interpersonal skills and trust were found to be important characteristic for a successful XP team.

Many studies have sought to identify how agile methods are perceived by different groups. Studies on customer perceptions report that *customers* are satisfied with the opportunities for feedback and responding to changes. However, we also found that the role of on-site customer can be stressful and cannot be sustained for a long period. *Developers* are mostly satisfied with agile methods. Companies that use XP have reported that their employees are more satisfied with their job and that they are more satisfied with the product. There were mixed findings regarding the effectiveness of pair programming and several developers regard it as an exhausting practice, because it requires heavy concentration. *University students* perceive agile methods as providing them with relevant training for their future work and believe that these methods improve the productivity in teams. However, they reported that pair programming was difficult when there was a large skill differential between the members of the pairs. In addition, test-first development was reported to be difficult for many students.

The group of comparative studies, in which variations of traditional development are compared to variations of agile development, is very interesting. It has been found that traditional and agile development methods are accompanied by differing practices of project management. Some studies suggest benefits in projects that use agile methods because changes are incorporated more easily and business value is demonstrated more efficiently. In addition, we found that it is also possible to combine agile project management with overall traditional principles, such as the stage-gate project management model. A limitation that was mentioned is that team members are less interchangeable in agile teams, which has consequences for how projects are managed. With respect to the productivity of agile and traditional teams, three of the four comparative studies that address this issue found that using XP results in increased productivity in terms of LOC/h. However, none of these studies had an appropriate recruitment strategy to ensure an unbiased comparison. There are also findings from several of the non-comparative studies that indicate that the subjects themselves believe that the productivity increases with the use of agile methods.

With respect to product quality, most studies report increased code quality when agile methods are used, but, again, none of these studies had an appropriate recruitment strategy to ensure an unbiased comparison. The size of the end product seems not to be correlated with the method of development used. Different studies have reported larger, smaller, and equal sizes of end product for traditional versus agile methods. The effect on work practices and job satisfaction of using agile and traditional methods has not been established conclusively. Some studies have found that work practice is more standardized when agile methods are used and that job satisfaction is greater. However, a study of team cohesion did not find any improvement of cohesion in an XP team.

Table 15
Definitions used for grading the strength of evidence [7]

| | |
|---|---|
| High | Further research is very unlikely to change our confidence in the estimate of effect |
| Moderate | Further research is likely to have an important impact on our confidence in the estimate of effect and may change the estimate |
| Low | Further research is very likely to have an important impact on our confidence in the estimate of effect and is likely to change the estimate |
| Very low | Any estimate of effect is very uncertain |

### 5.2. Strength of evidence

Several systems exist for making judgments about the strength of evidence in systematic reviews (see [7] for an overview). Most of these systems suggest that the strength of evidence can be based on a hierarchy with evidence from systematic reviews and randomized experiments at the top of the hierarchy and evidence from observational studies and expert opinion at the bottom of the hierarchy [36]. The inherent weakness with evidence hierarchies is that randomized experiments are not always feasible and that, in some instances, observational studies may provide better evidence.

To cope with the weaknesses of evidence hierarchies, we used the GRADE (Gradingof Recommendations Assessment, Development and Evaluation) workinggroup definitions to grade the overall strength of the evidences high, moderate, low, or very low [7] (see Table 15). According to GRADE, the strength of evidence can be determined on the basis of the combination of four key elements, i.e., in addition to study design, study quality, consistency, and directness are also evaluated. The GRADE system initially categorizes evidence concerning study design by assigning randomized experiments a *high* grade and observational studies a *low* grade. However, by considering the quality, consistency, and directness of the studies in the evidence base, the initial overall grade could be increased or decreased, i.e., evidence from inconsistent, low-quality experiments may be assigned a low grade, while strong or very strong evidence of association from two or more high-quality observational studies may be assigned a high grade [7].

Regarding study design, there were only three experiments in the review (two randomized trials), while the remaining primary studies were observational. That there are few experiments is natural because we included only studies that addressed agile methods as a whole and excluded ones that investigated specific practices in isolation. This is consistent with Shadish et al.'s comments that experiments are best used to investigate specific cause-effect phenomena [57]. Consequently, our initial categorization of the total evidence in this review based on study design is *low*. We now consider the quality, consistency, and directness of the studies in the evidence base.

With respect to the quality of the studies, methods were not, in general, described well; issues of bias, validity, and reliability were not always addressed; and methods of data collection and analysis were often not explained well (see Section 4.3). As many as 25 out of the 33 primary studies did not have a recruitment strategy that seemed appropriate for the aims stated for the research and 23 of the studies did *not* use other groups or baselines with which to compare their findings. Furthermore, in only one study was the possibility of researcher bias mentioned (see Table 9). Using these findings as a basis, we conclude that there are serious limitations to the quality of the studies that inevitably increasesthe risk of bias or confounding. Hence, we must be circumspect about the studies' reliability.

With respect to consistency, i.e., the similarity of estimates of effect across studies, we found differences in both the direction of effects and the size of the differences in effects, i.e., we found no consistent evidence of association from two or more studies with no plausible confounders nor did we find direct evidence from studies with no major threats to validity. These inconsistencies might by due to imprecise or sparse data, and reporting bias.

With respect to directness, i.e., the extent to which the people, interventions, and outcome measures are similar to those of interest, we found that most studies were concerned with XP. This leaves an uncertainty about the directness of evidence for other agile methods. However, given that most of the studies regarding XP were performed with student subjects or professionals who had little or no experience in agile development, this also raises an issue regarding the directness of evidence for XP. In addition, very few studies provided direct comparisons of interventions; hence, we had to make comparisons across studies. However, such indirect comparisons leave greater uncertainty than direct comparisons because of all the other differences between studies that can affect the results. Our judgment is thus that there are major uncertainties about the directness of the included studies.

Combining the four components of study design, study quality, consistency, and directness, we find that the strength of the evidence in the current review regarding the benefits and limitations of agile methods, and for decisions related to their adoption, is *very low*. Hence, any estimate of effect that is based on evidence of agile software development from current research is very uncertain. This is consistent with criticisms that have been raised regarding the sparse scientific support for many of the claims made by the agile community [42].

### 5.3. Implications for research and practice

This systematic review has a number of implications for research and practice. For research, the review shows a clear need for more empirical studies of agile development

methods. Agile development has had a deep impact on the software industry in recent years. In our opinion, this should lead to a greater interest amongst researchers as to what has driven the trend and what the effects are of the changes that emerge in response to the adoption of agile development.

This review also shows that, with rare exceptions, only XP has been studied. Hence, research on other agile approaches that are popular in industry should be a priority when designing future studies. In our opinion, management-oriented approaches, such as Scrum, are clearly the most under-researched compared to their popularity in industry.

Another striking finding is that only one research group in the world has studied mature agile development teams. If we want to investigate the potential of agile methods, we clearly need to direct more resources towards investigating the practices of mature teams.

The review shows that a range of research methods have been applied. We need to employ both flexible and fixed research designs if we are to gain a deeper understanding of agile development. Edmondson and McManus [25] argue that the research design needs to fit the current state of theory and research. They divide this state into three categories: nascent, intermediate, and mature; see Table 16. For agile software development, we believe the current state of theory and research on methods is clearly nascent, which suggests a need for exploratory qualitative studies. Rajlich [53] phrased it as a "backlog of research problems to be solved".

Other areas of research on agile software development, such as studies of particular practices, like pair programming, or areas that connect well to existing streams of software engineering research, might be described as being at an intermediate, or even a mature, state.

A major challenge is to increase the quality of studies on agile software development. In [58], Sjøberg et al. discuss measures to increase the quality of empirical studies in software engineering in general. Recently, Höst and Runeson [33] have suggested a checklist to use in case studies in software engineering. The recent special issue of *Information and Software Technology* on qualitative software engineering research [20] provides many useful examples of approaches for study designs, data collection, and analysis that should be relevant for future studies of agile software development. The state of research with respect to controlled experiments has been described thoroughly in a survey by Sjøberg et al. [59].

In order to increase the usefulness of the research for industry and to provide a sufficient number of studies of high quality on subtopics related to agile development, we think that researchers in the field should collaborate to determine a common research agenda. It lies beyond the scope of this article to suggest such an agenda, but we hope that the synthesis of research presented herein may provide the inspiration to create one.

For practitioners, this review shows that many promising studies of the use of agile methods have been reported. Although serious limitations have been identified, e.g., that the role of on-site customer seems to be unsustainable for long periods and that it is difficult to introduce agile methods into large and complex projects, the results of the review suggest that it is possible to achieve improved job satisfaction, productivity, and increased customer satisfaction.

The strongest, and probably most relevant, evidence for practice is from the studies of mature agile teams, which suggests that it is necessary to focus on human and social factors in order to succeed. Specifically, it seems that a high level of individual autonomy must be balanced with a high level of team autonomy and corporate responsibility. It also seems important to staff agile teams with people that have faith in their own abilities combined with good interpersonal skills and trust.

Evidence also suggests that instead of abandoning traditional project management principles, one should rather take advantage of these principles, such as state-gate project management models, and combine them with agile project management. The evidence also suggests that agile methods not necessarily are the best choice for large projects. Thus, consistent with recommendations provided by others [11,12,15], we suggest that practitioners carefully study their projects' characteristics and compare them with the relevant agile methods' required characteristics.

Due to the limited number and relatively poor quality of the primary studies in this review, it is impossible to offer more definitive and detailed advice. Rather, this review provides an overview of research carried out to date, which must be critically appraised by companies in order to identify similarities and differences between the studies reported and their own situation. A particular important aid in this appraisal is the description of the context of the studies in

Table 16
Categories of methodological fit in field research [25]

| State of prior theory and research | Nascent | Intermediate | Mature |
|---|---|---|---|
| Research questions | Open-ended inquiry about a phenomenon of interest | Proposed relationships between new and established constructs | Focused questions and/or hypotheses relating existing constructs |
| Type of data collected | Qualitative, initially open-ended data that need to be interpreted for meaning | Hybrid (both qualitative and quantitative) | Quantitative data: focused measures where extent or amount is meaningful |
| Theoretical contribution | A suggestive theory, often an invitation for further work on the issue or set of issues opened up by the study | A provisional theory, often one that integrates previously separate bodies of work | A supported theory that may add specificity, new mechanisms, or new boundaries to existing theories |

this review (Appendix D). A further aid would be to apply the principles of evidence-based software engineering in order to support and improve the decisions about what methods and technologies to employ [24].

The review clearly shows the need for more research in order to determine the situations in which advice on agile development that has been offered by practitioners may suitably be applied. We would like to urge companies to participate in research projects in the future, in order to target research goals that are relevant for the software industry. Action research is one such way of organizing collaboration between industry and researchers that would be highly relevant for a nascent field such as agile software development.

### 5.4. Limitations of this review

The main limitations of the review are bias in the selection of publications and inaccuracy in data extraction. To help to ensure that the process of selection was unbiased, we developed a research protocol in advance that defined the research questions. Using these questions as a basis, we identified keywords and search terms that would enable us to identify the relevant literature. However, it is important to recognize that software engineering keywords are not standardized and that they can be both discipline- and language-specific. Therefore, due to our choice of keywords and search strings, there is a risk that relevant studies were omitted. To avoid selection bias, we piloted every part of the review process, and in particular, the search strategy and citation management procedure, in order to clarify weaknesses and refine the selection process. Furthermore, since our focus was on empirical research, we excluded "lessons learned" papers and papers that were based merely on expert opinion. If the review had included this literature, the current study could, in principle, have provided more data. In that event, it might have been possible to draw more general conclusions. To further ensure the unbiased selection of articles, a multistage process was utilized that involved three researchers who documented the reasons for inclusion/exclusion at every step, as described in Section 3 and also as suggested by Kitchenham [36].

When we piloted the data extraction process, we found that several articles lacked sufficient details about the design and findings of a study and that, due to this, we differed too much in what we actually extracted. As a consequence, all data from all the 33 primary studies were extracted by the two authors in consensus meetings according to a predefined extraction form (Appendix C). However, we often found that the extraction process was hindered by the way some of the primary studies were reported. Many articles lacked sufficient information for us to be able to document them satisfactorily in the extraction form. More specifically, we frequently found that methods were not described adequately, that issues of bias and validity were not always addressed, that methods of data collection and analysis were often not explained well, and that samples and study settings were often not described well. There is therefore a possibility that the extraction process may have resulted in some inaccuracy in the data.

## 6. Conclusion

We identified 1996 studies from searches of the literature, of which 36 were found to be research studies of acceptable rigour, credibility, and relevance. Thirty-three of the 36 studies identified were primary studies, while three were secondary studies.

The studies fell into four thematic groups: introduction and adoption, human and social factors, perceptions of agile methods, and comparative studies. We identified a number of reported benefits and limitations of agile development within each of these themes. However, the strength of evidence is very low, which makes it difficult to offer specific advice to industry. Consequently, we advise readers from industry to use this article as a map of findings according to topic, which they can use to investigate relevant studies further and compare the settings in the studies to their own situation.

The studies investigated XP almost exclusively, and only a few of the studies on XP were done on mature development teams. A clear finding of the review is that we need to increase both the number and the quality of studies on agile software development. In particular, agile project management methods, such as Scrum, which are popular in industry, warrant further attention. We see that there is a backlog of research issues to be addressed. In this context, there is a clear need to establish a common research agenda for agile software development and for future field studies to pay more attention to the fit between their research methods and the state of prior work.

## Acknowledgements

The work in this paper was supported by the Research Council of Norway through the project Evidence-Based Software Engineering (181685/I30). We are grateful to Geir K. Hanssen at SINTEF ICT, who participated in selecting and assessing the studies included in this review. We are also grateful to Chris Wright for proofreading the paper.

## Appendix A. Studies included in the review

[S1] P. Abrahamsson, J. Koskela, Extreme programming: a survey of empirical data from a controlled case study, in: Proceedings – 2004 International Symposium on Empirical Software Engineering, ISESE 2004, Aug 19–20 2004, Redondo Beach, CA, United States, 2004.

[S2] B. Bahli, E.S.A. Zeid, The role of knowledge creation in adopting extreme programming model: an empirical study, in: ITI 3rd International Confer-

ence on Information and Communications Technology: Enabling Technologies for the New Knowledge Society, 2005.

[S3] R. Baskerville, B. Ramesh, L. Levine, J. Pries-Heje, S. Slaughter, Is internet-speed software development different? IEEE Software 20(6) (2003) 70–77.

[S4] M. Ceschi, A. Sillitti, G. Succi, S. De Panfilis, Project management in plan-based and agile companies, IEEE Software 22(3) (2005) 21–27.

[S5] J. Chong, Social behaviours on XP and non-XP teams: a comparative study, in: Proceedings of the Agile Development Conference (ADC'05), 2005.

[S6] A. Dagnino, K. Smiley, H. Srikanth, A.I. Anton, L. Williams, Experiences in applying agile software development practices in new product development, in: Proceedings of the 8th IASTED International Conference on Software Engineering and Applications, November 9–11, Cambridge, MA, United States, 2004.

[S7] D. Dalcher, O. Benediktsson, H. Thorbergsson, Development life cycle management: a multiproject experiment, in: Proceedings of the 12th International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05), 2005.

[S8] A. Fruhling, K. Tyser, G.-J. De Vreede, Experiences with extreme programming in telehealth: developing and implementing a biosecurity health care application, in: Proceedings of the 38th Hawaii International Conference on System Sciences (HICCS), Hawaii, USA, 2005.

[S9] M.-R. Hilkka, T. Tuure, R. Matti, Is extreme programming just old wine in new bottles: a comparison of two cases, Journal of Database Management 16(4) (2005) 41–61.

[S10] S. Ilieva, P. Ivanov, E. Stefanova, Analyses of an agile methodology implementation, in: Proceedings 30th Euromicro Conference, IEEE Computer Society Press, 2004, pp. 326–333.

[S11] T. Jokela, P. Abrahamsson, Usability assessment of an extreme programming project: close co-operation with the customer does not equal to good usability, in: Product Focused Software Process Improvement, Lecture Notes in Computer Science, vol. 3009, Springer Verlag, Berlin, 2004, pp. 393–407.

[S12] D. Karlström, P. Runeson, Combining agile methods with stage-gate project management, IEEE Software 22(3) (2005) 43–49.

[S13] J. Koskela, P. Abrahamsson, On-site customer in an XP project: empirical results from a case study, in: T. Dingsøyr (Ed.), Software Process Improvement, Proceedings, Lecture Notes in Computer Science, vol. 3281, Springer-Verlag, Berlin, 2004, pp. 1–11.

[S14] L. Layman, L. Williams, L. Cunningham, Exploring extreme programming in context: an industrial case study, Agile Development Conference, 2004.

[S15] F. Macias, M. Holcombe, M. Gheorghe, A formal experiment comparing extreme programming with traditional software construction, in: Proceedings of the Fourth Mexican International Conference on Computer Science (ENC 2003), 2003.

[S16] A. Mackenzie, S. Monk, From Cards to Code: How Extreme Programming Re-Embodies Programming as a Collective Practice, Computer Supported Cooperative Work, vol. 13, 2004, pp. 91–117.

[S17] C. Mann, F. Maurer, A case study on the impact of scrum on overtime and customer satisfaction, Agile Development Conference, 2005.

[S18] K. Mannaro, M. Melis, and M. Marchesi, Empirical analysis on the satisfaction of IT employees comparing XP practices with other software development methodologies, in: Extreme Programming and Agile Processes in Software Engineering, Proceedings, Lecture Notes in Computer Science, vol. 3092, Springer Verlag, 2004, pp. 166–174.

[S19] A. Martin, R. Biddle, J. Noble, The XP customer role in practice: three studies, Agile Development Conference, 2004.

[S20] A. Martin, R. Biddle, J. Noble, When XP met outsourcing, in Extreme Programming and Agile Processes in Software Engineering, Proceedings, Lecture Notes in Computer Science, vol. 3092, Springer Verlag, Berlin, 2004, pp. 51–59.

[S21] G. Melnik, F. Maurer, Perceptions of agile practices: a student survey, in: Proceedings, eXtreme Programming/Agile Universe 2002, Lecture Notes in Computer Science, vol. 2418, Springer Verlag, 2002, pp. 241–250.

[S22] G. Melnik, F. Maurer, A cross-program investigation of student's perceptions of agile methods, in: International Conference on Software Engineering (ICSE), St. Louis, MI, USA, 2005.

[S23] P. Middleton, Lean software development: two case studies, Software Quality Journal 9(4) (2001) 241–252.

[S24] H. Robinson, H. Sharp, The characteristics of XP teams, in: Extreme Programming and Agile Processes in Software Engineering, Lecture Notes in Computer Science, vol. 3092, Springer Verlag, Berlin, 2004, pp. 139–147.

[S25] H. Robinson, H. Sharp, The social side of technical practices, in: Extreme Progamming and Agile Processes in Software Engineering, Lecture Notes in Computer Science, vol. 3556, Springer Verlag, Berlin, 2005, pp. 100–108.

[S26] H.S. Robinson, Organisational culture and XP: three case studies, in: Proceedings of the Agile Conference (ADC'05), 2005.

[S27] H. Sharp H. Robinson, An ethnographic study of XP practice, Empirical Software Engineering, 9(4) (2004) 353–375.

[S28] A. Sillitti, M. Ceschi, B. Russo, G. Succi, Managing uncertainty in requirements: a survey in documentation-driven and agile companies, in: Proceedings of the 11th International Software Metrics Symposium (METRICS), 2005.

[S29] H. Svensson, M. Höst, Introducing agile process in a software maintenance and evolution organization, in: Ninth European Conference on Software Maintenance and Reengineering (CSMR'05), 2005.
[S30] H. Svensson, M. Höst, Views from an organization on how agile development affects its collaboration with a software development team, Lecture Notes in Compuer Science, vol. 3547, Springer Verlag, Berlin, 2005, pp. 487–501.
[S31] Tessem, Experiences in learning xp practices: a qualitative study, in: XP 2003, vol. 2675, Springer Verlag, Berlin, 2003, pp. 131–137.
[S32] C.A. Wellington, T. Briggs, C.D. Girard, Comparison of student experiences with plan-driven and agile methodologies, in: Proceeedings of the 35th ASEE/IEEE Frontiers in Education Conference, 2005.
[S33] S.M. Young, H.M. Edwards, S. Mcdonald, J.B. Thompson, Personality characteristics in an XP team: A repertory grid study, in: Proceedings of Human and Social Factors of Software Engineering (HSSE), St. Louis, MI, USA, 2005.
[S34] P. Abrahamsson, O. Salo, J. Ronkainen, J. Warsta, Agile software development methods: Review and analysis, VTT Technical report, 2002.
[S35] D. Cohen, M. Lindvall, P. Costa, An introduction to agile methods, in: M. V. Zelkowitz (Ed.), Advances in Computers, Advances in Software Engineering, vol. 62, Elsevier, Amsterdam, 2004.
[S36] J. Erickson, K. Lyytinen, K. Siau, Agile modeling, Agile software development, and extreme programming: the state of research, Journal of Database Management 16(4) (2005) 88–100.

## Appendix B. Quality assessment form

### Screening questions:

| | |
|---|---|
| **1. Is this a research paper?**<br>*Consider:*<br>*—Is the paper based on research (or is it merely a "lessons learned" report based on expert opinion?* | ☐ Yes ☐ No |
| **2. Is there a clear statement of the aims of the research?**<br>*Consider:*<br>*—Is there a rationale for why the study was undertaken?*<br>*—Is the study's focus or main focus on Agile Software Development?*<br>*—Does the study present empirical data?*<br>*—Is there a clear statement of the study's primary outcome (i.e. time-to-market, cost, or product or process quality)?* | ☐ Yes ☐ No |
| **3. Is there an adequate description of the context in which the research was carried out?**<br>*Consider whether the researcher has identified:*<br>*—The industry in which products are used (e.g. banking, telecommunications, consumer goods, travel, etc)*<br>*—The nature of the software development organization (e.g. in-house department or independent software supplier)*<br>*—The skills and experience of software staff (e.g. with a language, a method, a tool, an application domain)*<br>*—The type of software products used (e.g. a design tool, a compiler)*<br>*—The software processes being used (e.g. a company standard process, the quality assurance procedures, the configuration management process)* | ☐ Yes ☐ No |

**If question 1, or both of questions 2 and 3, receive a "No" response do not continue with the quality assessment.**

### Detailed questions:

| | |
|---|---|
| **Research design**<br>4. Was the research design appropriate to address the aims of the research?<br>*Consider:*<br>– *Has the researcher justified the research design (e.g. have they discussed how they decided which methods to use)?* | ☐ Yes ☐ No |

| | |
|---|---|
| **Sampling**<br>5. Was the recruitment strategy appropriate to the aims of the research?<br>*Consider:*<br>*—Has the researcher explained how the participants or cases were identified and selected?*<br>*—Are the cases defined and described precisely?*<br>*—Were the cases representative of a defined population?*<br>*—Have the researchers explained why the participants or cases they selected were the most appropriate to provide access to the type of knowledge sought by the study?*<br>*—Was the sample size sufficiently large?* | ☐ Yes ☐ No |
| **Control group**<br>6. Was there a control group with which to compare treatments?<br>*Consider:*<br>*—How were the controls selected?*<br>*—Were they representative of a defined population?*<br>*—Was there anything special about the controls?*<br>*—Was the non-response high? Could non-respondents be different in any way?* | ☐ Yes ☐ No |
| **Data collection**<br>7. Was the data collected in a way that addressed the research issue?<br>*Consider:*<br>*—Were all measures clearly defined (e.g. unit and counting rules)?*<br>*—Is it clear how data was collected (e.g. semi-structured interviews, focus group etc.)?*<br>*—Has the researcher justified the methods that were chosen?*<br>*—Has the researcher made the methods explicit (e.g. is there an indication of how interviews were conducted, did they use an interview guide)?*<br>*—If the methods were modified during the study, has the researcher explained how and why?*<br>*—Whether the form of the data is clear (e.g. tape recording, video material, notes etc.)*<br>*—Whether quality control methods were used to ensure completeness and accuracy of data collection* | ☐ Yes ☐ No |
| **Data analysis**<br>8. Was the data analysis sufficiently rigorous?<br>*Consider:*<br>*—Was there an in-depth description of the analysis process?*<br>*—If thematic analysis was used, is it clear how the categories/ themes were derived from the data?*<br>*—Has sufficient data been presented to support the findings?*<br>*—To what extent has contradictory data been taken into account?*<br>*—Whether quality control methods were used to verify the results* | ☐ Yes ☐ No |
| **Reflexivity (research partnership relations/recognition of researcher bias)**<br>9. Has the relationship between researcher and participants been considered adequately?<br>*Consider:*<br>*—Did the researcher critically examine their own role, potential bias and influence during the formulation of research questions, sample recruitment, data collection, and analysis and selection of data for presentation?*<br>*—How the researcher responded to events during the study and whether they considered the implications of any changes in the research design.* | ☐ Yes ☐ No |
| **Findings**<br>10. Is there a clear statement of findings?<br>*Consider:*<br>*—Are the findings explicit (e.g. magnitude of effect)?*<br>*—Has an adequate discussion of the evidence, both for and against the researcher's arguments, been demonstrated?*<br>*—Has the researcher discussed the credibility of their findings (e.g. triangulation, respondent validation, more than one analyst)?*<br>*—Are limitations of the study discussed explicitly?*<br>*—Are the findings discussed in relation to the original research questions?*<br>*—Are the conclusions justified by the results?* | ☐ Yes ☐ No |

| **Value of the research** | |
|---|---|
| 11. Is the study of value for research or practice?<br>*Consider:*<br>*—Does the researcher discuss the contribution the study makes to existing knowledge or understanding (e.g. do they consider the findings in relation to current practice or relevant research-based literature)?*<br>*—Does the research identify new areas in which research is necessary?*<br>*—Does the researcher discuss whether or how the findings can be transferred to other populations, or consider other ways in which the research can be used?* | ☐ Yes ☐ No |

## Appendix C. Data extraction form

| | | |
|---|---|---|
| *Study description* | | |
| 1. | Study identifier | Unique id for the study |
| 2. | Date of data extraction | |
| 3. | Bibliographic reference | Author, year, title, source |
| 4. | Type of article | Journal article, conference paper, workshop paper, book section |
| 5. | Study aims | What were the aims of the study? |
| 6. | Objectives | What were the objectives? |
| 7. | Design of study | Qualitative, quantitative (experiment, survey, case study, action research) |
| 8. | Research hypothesis | Statement of hypotheses, if any |
| 9. | Definition of agile software development given in study | Verbatim from the study |
| 10. | Sample description | Size, students, professionals (age, education, experience) |
| 11. | Setting of study | Industry, in-house/supplier, products and processes used |
| 12. | Control group | Yes, no (number of groups, sample size) |
| 13. | Data collection | How was the data obtained? (questionnaires, interviews, forms) |
| 14. | Data analysis | How was the data analyzed? (qualitative, quantitative) |
| *Study findings* | | |
| 1. | Findings and conclusions | What were the findings and conclusions? (verbatim from the study) |
| 2. | Validity | Limitations, threats to validity |
| 3. | Relevance | Research, practice |

## Appendix D. Overview of primary studies

| ID | Research method | Agile method | Agile experience | Professional/ Student | Project duration | Team size | Domain, comment |
|---|---|---|---|---|---|---|---|
| S1 | Singlecase | XP | Beginner | Student | 8,4 weeks | 4 | Research prototype developed |
| S2 | Multicase | XP | Beginner | Professional | 1 year | 9 | Medical information systems |
| S3 | Mixed | General | – | Professional | – | NA | Web development |
| S4 | Survey | General | NA | Professional | – | NA | NA |
| S5 | Multicase | XP | Beginner | Professional | – | 7–12 | Mid-size software start-up |
| S6 | Multicase | Other | Beginner | Professional | 2700 h | 5 | Industrial automation |
| S7 | Experiment | XP | Beginner | Student | 1 year | 3–4 | NA |
| S8 | Singlecase | XP | Beginner | Professional | 21 months | 4 | Medical information systems |
| S9 | Multicase | XP | Beginner | Professional | NA/18 months | 6/4 | Factory system + communication system |
| S10 | Singlecase | XP | Beginner | Professional | 900 h | 4 | Financial software |
| S11 | Singlecase | XP | Beginner | Student | 8,4 weeks | 4 | Research prototype developed |
| S12 | Multicase | General | Beginner | Professional | – | – | Industrial automation/Defence/Telecom |
| S13 | Singlecase | XP | Beginner | Student | 8,4 weeks | 4 | Research prototype developed |
| S14 | Singlecase | XP | Beginner | Professional | 3,5 months | 10 | Airline company software |
| S15 | Experiment | XP | Beginner | Student | 1 semester | 4–5 | NA |
| S16 | Singlecase | XP | Beginner | Professional | – | 6–12 | Knowledge management software |
| S17 | Singlecase | Scrum | Beginner | Professional | 22 months | 4–6 | Oil and gas software |
| S18 | Survey | XP | NA | Professional | NA | NA | NA |

(*continued on next page*)

**Appendix C.** (*continued*)

| ID | Research method | Agile method | Agile experience | Professional/ Student | Project duration | Team size | Domain, comment |
|---|---|---|---|---|---|---|---|
| S19 | Multicase | XP | Beginner | Professional | 15/45/18 | 11/8/16 | – |
| S20 | Multicase | XP | Beginner | Professional | 15 months/18+ months | 11/60 | – |
| S21 | Survey | XP | Beginner | Student | NA | NA | NA |
| S22 | Mixed | General | Beginner | Student | – | NA | NA |
| S23 | Multicase | LSD | Beginner | Professional | 3 days | 2 | Financial system |
| S24 | Multicase | XP | Mature | Professional | – | 8/23 | Web applications/document software |
| S25 | Multicase | XP | Mature | Professional | – | 7/23/8 | Mid-size software start-up |
| S26 | Multicase | XP | Mature | Professional | – | 12/20/8 | Banking/Content security software/web-applications |
| S27 | Singlecase | XP | Mature | Professional | – | 10 | – |
| S28 | Survey | General | NA | Professional | – | NA | NA |
| S29 | Singlecase | XP | Beginner | Professional | – | – | Software house |
| S30 | Singlecase | XP | Beginner | Professional | – | – | Software maintenance and evolution |
| S31 | Singlecase | XP | Beginner | Student | 3 weeks | 6 | Educational software |
| S32 | Experiment | XP | Beginner | Student | 1 semester | 16 | NA |
| S33 | Singlecase | XP | – | Professional | – | 6 | Software house |

Several numbers for a study in the columns 'Project duration' and 'Team size' indicate that the study included several teams.

## References


[1] North American and European Enterprise Software and Services Survey, Business Technographics Ed., 2005.

[2] P. Abrahamsson, O. Salo, J. Ronkainen, J. Warsta, Agile software development methods: review and analysis, VTT Technical report, 2002.

[3] P. Abrahamsson, J. Warsta, M.T. Siponen, J. Ronkainen, New directions on agile methods: a comparative analysis, in: Proceedings of the 25th International Conference on Software Engineering (ICSE'03), IEEE Press, 2003.

[4] R.L. Ackoff, Alternative types of planning, in: Ackoff's Best: His Classic Writings on Management, Wiley, New York, 1999, pp. 104–114.

[5] A. Anderson, R. Beattie, K. Beck, D. Bryant, M. Dearment, M. Fowler, M. Fronczak, R. Garzaniti, D. Gore, B. Hacker, C. Hendrickson, R. Jeffries, D. Joppie, D. Kim, P. Kowalsky, D. Mueller, T. Murasky, R. Nutter, A. Pantea, D. Thomas, Chrysler goes to extremes, Distributed Computing Magazine (October) (1998) 24–28.

[6] M. Aoyama, Web-based agile software development, IEEE Software 15 (6) (1998) 56–65.

[7] D. Atkins, D. Best, P.A. Briss, M. Eccles, Y. Falck-Ytter, S. Flottorp, G.H. Guyatt, R.T. Harbour, M.C. Haugh, D. Henry, S. Hill, R. Jaeschke, G. Leng, A. Liberati, N. Magrini, J. Mason, P. Middleton, J. Mrukowicz, D. O'connell, A. D Oxman, B. Phillips, H.J. Schünemann, T.T.-T. Edejer, H. Varonen, G.E. Vist, J.W. Williams Jr., Z. Stephanie, Grading quality of evidence and strength of recommendations, BMJ 328 (1490) (2004).

[8] D. Avison, F. Lau, M. Myers, P.A. Nielsen, Action research, Communications of the ACM 42 (1) (1999) 94–97.

[9] K. Beck, Extreme Programming Explained: Embrace Change, Addison-Wesley, 2000, ISBN 0-201-61641-6.

[10] K. Beck, Extreme Programming Explained: Embrace Chage, second ed., Addison-Wesley, 2004, ISBN 978-0321278654.

[11] B. Boehm, Get ready for agile methods, with care, IEEE Computer 35 (1) (2002) 64–69.

[12] B. Boehm, R. Turner, Balancing Agility and Discipline: A Guide for the Perplexed, Addison-Wesley, Boston, 2003, ISBN 978-0321186126.

[13] N. Britten, R. Campbell, C. Pope, J. Donovan, M. Morgan, R. Pill, Using meta ethnography to synthesise qualitative research: a worked example, Journal of Health Services Research and Policy 7 (4) (2002) 209–215.

[14] P. Checkland, J. Scholes, Soft Systems Methodology in Action, Wiley, Chichester, 1990, ISBN 0-471-98605-4.

[15] A. Cockburn, Selecting a project's methodology, IEEE Software 17 (4) (2000) 64–71.

[16] A. Cockburn, Crystal Clear: A Human-Powered Methodology for Small Teams, Addison-Wesley, 2004, ISBN 0-201-69947-8.

[17] D. Cohen, M. Lindvall, P. Costa, An introduction to agile methods, in: M.V. Zelkowitz (Ed.), Advances in Computers, Advances in Software Engineering, vol. 62, Elsevier, Amsterdam, 2004.

[18] J. Cohen, A coefficient of agreement for nominal scales, Educational and Psychological Measurement 20 (1960) 37–46.

[19] K. Conboy, B. Fitzgerald, Toward a conceptual framework of agile methods: a study of agility in different disciplines, in: Proceedings of XP/Agile Universe, Springer Verlag, 2004.

[20] Y. Dittrich, M. John, J. Singer, B. Tessem, For the special issue on qualitative software engineering research, Information and Software Technology 49 (6) (2007) 531–539.

[21] T. Dybå, Improvisation in small software organizations, IEEE Software 17 (5) (2000) 82–87.

[22] T. Dybå, E. Arisholm, D. Sjøberg, J. Hannay, F. Shull, Are two heads better than one? On the effectiveness of pair-programming, IEEE Software 24 (6) (2007) 10–13.

[23] T. Dybå, T. Dingsøyr, G.K. Hanssen, Applying systematic reviews to diverse study types: an experience report, in: Proceedings of the 1st International Symposium on Empirical Software Engineering and Measurement (ESEM'07), IEEE Computer Society, Madrid, Spain, 2007, pp. 225–234.

[24] T. Dybå, B. Kitchenham, M. Jørgensen, Evidence-based software engineering for practitioners, IEEE Software 22 (1) (2005) 58–65.

[25] A.C. Edmondson, S.E. Mcmanus, Methodological fit in management field research, Academy of Management Review 32 (4) (2007) 1155–1179.

[26] H. Erdogmus, M. Morisio, M. Torchiano, On the effectiveness of the test-first approach to programming, IEEE Transactions on Software Engineering 31 (3) (2005) 226–237.

[27] J. Erickson, K. Lyytinen, K. Siau, Agile Modeling, Agile software development, and extreme programming: the state of research, Journal of Database Management 16 (4) (2005) 88–100.

[28] T. Gilb, Competitive Engineering: A Handbook for Systems Engineering, Requirements Engineering, and Software, Elsevier Butterworth-Heinemann, Oxford, 2005, ISBN 0-7506-6507-6.

[29] T. Greenhalgh, How to Read a Paper, second ed., BMJ Publishing Group, London, 2001.

[30] A. Gunasekaran, Agile manufacturing: A framework for research and development, International Journal of Production Economics 62 (1–2) (1999) 87–105.

[31] J.P.T. Higgins, S. Green (Eds.), Cochrane Handbook for Systematic Reviews of Interventions, Version 5.0.0 (updated February 2008), The Cochrane Collaboration, 2008. Available from: <www.cochrane-handbook.org>.
[32] W.S. Humphrey, PSP: A Self-Improvement Process for Software Engineers, Addison-Wesley, 2005, ISBN 978-0321305497.
[33] M. Höst, P. Runeson, Checklists for software engineering case study research, in: Proceedings of the First International Symposium on Empirical Software Engineering and Measurement, IEEE, Madrid, Spain, 2007, pp. 479–481.
[34] G. Keefer, Extreme Programming Considered Harmful for Reliable Software Development 2.0, AVOCA GmbH, Online Report, 2003.
[35] K.S. Khan, G. Ter Riet, J. Glanville, A.J. Sowden, J. Kleijnen, Undertaking Systematic Review of Research on Effectiveness, CRD's Guidance for those Carrying Out or Commissioning Reviews, CRD Report Number 4, second ed., NHS Centre for Reviews and Dissemination, University of York, 2001.
[36] B.A. Kitchenham, Guidelines for performing Systematic Literature Reviews in Software Engineering Version 2.3, Keele University and University of Durham, EBSE Technical Report, 2007.
[37] B.A. Kitchenham, S.L. Pfleeger, L.M. Pickard, P.W. Jones, D.C. Hoaglin, K. El Emam, J. Rosenberg, Preliminary guidelines for empirical research in software engineering, IEEE Transactions on Software Engineering 28 (8) (2002) 721–734.
[38] P. Krutchen, The Rational Unified Process: An Introduction, third ed., Addison-Wesley, Boston, 2003.
[39] J.R. Landis, G.G. Koch, The measurement of observer agreement for categorical data, Biometrics 33 (1) (1977) 159–174.
[40] C. Larman, V.R. Basili, Iterative and incremental development: a brief history, IEEE Computer 36 (6) (2003) 47–56.
[41] J. Mcavoy, T. Butler, The impact of the Abilene Paradox on double-loop learning in an agile team, Information and Software Technology 49 (6) (2007) 552–563.
[42] P. Mcbreen, Questioning Extreme Programming, Pearson Education, Boston, MA, USA, 2003, ISBN 0-201-84457-5.
[43] H. Merisalo-Rantanen, T. Tuure, R. Matti, Is extreme programming just old wine in new bottles: a comparison of two cases, Journal of Database Management 16 (4) (2005) 41–61.
[44] P. Meso, R. Jain, Agile software development: adaptive systems principles and best practices, Information Systems Management 23 (3) (2006) 19–30.
[45] M.B. Miles, M. Huberman, Qualitative Data Analysis: An Expanded Sourcebook, second ed., Sage Publications, 1994, ISBN 0803955405.
[46] S. Nerur, V. Balijepally, Theoretical reflections on agile development methodologies, Communications of the ACM 50 (3) (2007) 79–83.
[47] S. Nerur, R. Mahapatra, G. Mangalaraj, Challenges of migrating to agile methodologies, Communications of the ACM (May) (2005) 72–78.
[48] G.W. Noblit, R.D. Hare, Meta-Ethnography: Synthesizing Qualitative Studies, Sage Publications, London, 1988.
[49] T. Ohno, Toyota Production System: Beyond Large-scale Production, Productivity Press, New York, USA, 1988, ISBN 0-915299-14-3.
[50] S.R. Palmer, J.M. Felsing, A Practical Guide to Feature-driven Development, Prentice Hall, Upper Saddle River, NJ, 2002, ISBN 0-13-067615-2.
[51] M.C. Paulk, C.V. Weber, B. Curtis, M.B. Chrissis, The Capability Maturity Model: Guidelines for Improving the Software Process, Addison-Wesley, Boston, 1995, ISBN 0-201-54664-7.
[52] M. Poppendieck, T. Poppendieck, Lean Software Development – An Agile Toolkit for Software Development Managers, Addison-Wesley, Boston, 2003, ISBN 0-321-15078-3.
[53] V. Rajlich, Changing the paradigm of software engineering, Communications of the ACM 49 (8) (2006) 67–70.
[54] C.K. Riemenschneider, B.C. Hardgrave, F.D. Davis, Explaining software developer acceptance of methodologies: a comparison of five theoretical models, IEEE Transactions on Software Engineering 28 (12) (2002) 1135–1145.
[55] L.M. Sanchez, R. Nagi, A review of agile manufacturing systems, International Journal of Production Research 39 (16) (2001) 3561–3600.
[56] K. Schwaber, M. Beedle, Agile Software Development with Scrum, Prentice Hall, Upper Saddle River, 2001.
[57] W.R. Shadish, T.D. Cook, D.T. Campbell, Experimental and Quasi-Experimental Designs for Generalized Causal Inference, Houghton Mifflin Company, Boston, 2002.
[58] D. Sjøberg, T. Dybå, M. Jørgensen, The Future of Empirical Methods in Software Engineering Research, in: Future of Software Engineering (FOSE'07), IEEE, 2007, pp. 358–378.
[59] D. Sjøberg, J.E. Hannay, O. Hansen, V.B. Kampenes, A. Karahasanovic, N.-K. Liborg, A.C. Rekdal, A survey of controlled experiments in software engineering, IEEE Transactions on Software Engineering 31 (9) (2005) 733–753.
[60] J. Stapleton, DSDM: Business Focused Development, second ed., Pearson Education, 2003, ISBN 978-0321112248.
[61] M. Stephens, D. Rosenberg, Extreme Programming Refactored: The Case Against XP, Apress, Berkeley, CA, 2003, ISBN 1-59059-096-1.
[62] A. Strauss, J. Corbin, Basics of Qualitative Research, second ed., Sage Publications, 1998, ISBN 0-8039-5939-7.
[63] H. Takeuchi, I. Nonaka, The new product development game, Harvard Business Review (January) (1986) 137–146.
[64] D. Turk, R. France, B. Rumpe, Assumptions underlying agile software-development processes, Journal of Database Management 16 (4) (2005) 62–87.
[65] E. Wenger, Communities of Practice: Learning, Meaning and Identity, Cambridge University Press, Cambridge, UK, 1998, ISBN 0-521-43017-8.
[66] L. Williams, A. Cockburn, Agile software development: it's about feedback and change, IEEE Computer 36 (6) (2003) 39–43.
[67] J.P. Womack, D.T. Jones, D. Roos, The Machine that Changed the World: The Story of Lean Production – Toyota's Secret Weapon in the Global Car Wars that is Now Revolutionizing World Industry, Free Press, 2007, ISBN 978-0743299794.
[68] P. Ågerfalk, B. Fitzgerald, Flexible and distributed software processes: old petunias in new bowls? Communications of the ACM 49 (10) (2006) 27–34.