


RESEARCH

Open Access



Back to the future: origins and directions of the “Agile Manifesto” – views of the originators

Philipp Hohl^{1*} , Jil Klünder², Arie van Bennekum³, Ryan Lockard⁴, James Gifford⁵, Jürgen Münch⁶, Michael Stupperich¹ and Kurt Schneider²

*Correspondence:

philipp.hohl@daimler.com

¹Daimler AG,

Research and Development,
Wilhelm-Runge-Str. 11, 89081 Ulm,
Germany

Full list of author information is
available at the end of the article

Abstract

In 2001, seventeen professionals set up the manifesto for agile software development. They wanted to define values and basic principles for better software development. On top of being brought into focus, the manifesto has been widely adopted by developers, in software-developing organizations and outside the world of IT. Agile principles and their implementation in practice have paved the way for radical new and innovative ways of software and product development. In parallel, the understanding of the manifesto's underlying principles evolved over time. This, in turn, may affect current and future applications of agile principles. This article presents results from a survey and an interview study in collaboration with the original contributors of the manifesto for agile software development. Furthermore, it comprises the results from a workshop with one of the original authors. This publication focuses on the origins of the manifesto, the contributors' views from today's perspective, and their outlook on future directions. We evaluated 11 responses from the survey and 14 interviews to understand the viewpoint of the contributors. They emphasize that agile methods need to be carefully selected and agile should not be seen as a silver bullet. They underline the importance of considering the variety of different practices and methods that had an influence on the manifesto. Furthermore, they mention that people should question their current understanding of “agile” and recommend reconsidering the core ideas of the manifesto.

Keywords: Manifesto for agile software development, Agile manifesto, Agile methods, Interview review

1 Introduction

Nowadays, companies are confronted with high-frequent changes due to innovations and new technology (Broy 2006). Innovations such as cloud based services, big data and digitalization are spreading into all markets and all kinds of products. A lot of innovation is achieved in the combination of software and new types of devices. This combination offers new opportunities in a fast pace, which asks one by one a response from the market. Consequently, the amount of software and connected solutions, e.g. in cars (Broy 2006), increased exponentially during the last decades. Therefore, it is a competitive advantage to develop and distribute high-quality software and connected solutions at a high pace.

Agile software development methods are a promising solution to keep pace with this progress. The rise of agile methods and practices started to get significant attention 17

years ago with the manifesto for agile software development, often referred to as “agile manifesto” although not meant to be agile itself (Thomas 2017).

A lot of publications (Highsmith 2002; Dingsøyr et al. 2012; Abrahamsson et al. 2002) which were published during the last 17 years interpret the statements from the manifesto and introduce agile practices, principles, and methods. The manifesto is usually referred to whenever developers aim at conducting agile development (Beck et al. 2001):

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

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

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

The manifesto was written by 17 software practitioners in 2001 in Snowbird, near Salt Lake County, Utah. On being asked about the motivation for setting up the manifesto, Kent Beck mentions that they wanted to “*uncover better ways of developing software*” (Beck et al. 2001). Ron Jeffries adds that the original idea “*Kent Beck had in mind with Extreme Programming, [was] to make the world safe for programmers*” (Lockard et al. 2017). A “safe” development environment is not only the motivation of Extreme Programming (XP). It is common for most agile methods, to improve the working conditions of developers. This can be seen as a main factor for the success of the manifesto for agile software development.

The rise of agile methods was welcomed with enthusiasm by many developers but also led to criticism. At least two mindsets of approaching agile development with potentially different outcomes could be observed: There are those developers who apply agile practices because they believe in the values and principles of the manifesto, and those who do it because it is seen as best practice. The manifesto for agile software development seemed to promise a more successful way of developing software by “just” following the original values and principles. This, in turn, makes the manifesto special. Some developers believed and still believe in the manifesto as the “Holy Grail” for successful software development, while others denote it as a marketing gimmick to sell intuitive development behavior within a new livery.

Nevertheless, the ongoing success of agile had several implications. Among others, new trends like scaling agile with Scrum of Scrums (SoS) or the Scaled Agile Framework (SAFe) are common talk. The original ideas of the manifesto have become more and more commercialized. Many developers and managers that are now adopting agile are not aware of the initial diversity of agile methods and the underlying principles. Scrum is often seen as the only agile practice (Klunder et al. 2017).

The so-called agile transformation of organizations is a frequently discussed topic because agile ways of working promise to make companies future-proof (Lockard and Cleff 2016). However, the developers often declare themselves “agile” when just using Scrum by the book. In this case, the meaning of agile is

wrongly interpreted, not understood, and commercialized (Klunder et al. 2017). This is one of the reasons why we conducted this survey and the interview study: In our daily work, we have often seen the trend of doing agile for the sake of doing agile and we wanted to know what the originators think about it.

A variety of agile methodologies influenced the manifesto, its values and principles. The aim of this article is to recall the past and to reflect on the present agile world. Therefore, we scrutinized the manifesto from various perspectives. We collected first-hand data to draw an undistorted picture. We combine the results of a survey, in which 11 of the 17 original contributors participated, with the outcome of an interview study with 14 of the 17 contributors. Furthermore, we gained insights from a workshop with Arie van Bennekum who is among the authors of the agile manifesto and also a co-author of this article. We explicitly indicate throughout the article when his individual perspective or opinion is presented. As secondary data, we also included insights from previously held talks at conferences of given interviews of the original authors.

The main contributions of this work consist in:

- Collect background information to identify the techniques and methods that influenced the manifesto.
- Identify the evolution of interpretations of the statements in the manifesto and future directions from the original contributors' point of view.
- Present selected challenges, prerequisites and success factors for an agile transformation.

The remainder of the article is structured as follows. Section 2 discusses related work. Section 3 defines the study approach and the research design, while Section 4 presents and discusses the findings of our research. In Section 5, we discuss the findings and conclude the publication in Section 6.

2 Background and related work

There are interesting contributions from the time before the rise of the manifesto (Larman and Basili 2003) and afterwards (Dingsøyr et al. 2012).

2.1 Before the manifesto

Larman and Basili (2003) describe the history of Iterative and Incremental Software development (IID), starting in the pre-70s and ending up with the manifesto in 2001. They describe the relationship between IID and agile methods as follows: "In February 2001, a group of 17 process experts [...] interested in promoting modern, simple IID methods and principles met in Utah to discuss common ground" ((Larman and Basili 2003), p. 54). In the publication by Larman and Basili (2003) they structured the agile history into decades. According to the research of Larman and Basili (2003), the agile mindset started in the 1930s with the idea of "plan-do-study-act" cycles. They mention several projects, such as NASA's project Mercury (the first human spaceflight program of the United States) or the software development for the US Navy's helicopter-to-ship weapon system, which all applied IID practices. They point out that practices such as short iterations and test-first development were already used in the Mercury project. These practices remained present in the agile methods such as Scrum or XP (Larman and Basili 2003).

In the seventies, Royce (1970) published an article which is considered as the basis for the waterfall-model. In his article Royce describes his personal views of managing large software developments and what is “*necessary to transform a risky development process into one that will provide the desired product*” ((Royce 1970), p. 335). His approach aims at reaching the condition of working software delivered on-time and within costs (Royce 1970). Royce proposes to use the waterfall model’s phases with an iterative relationship between successive phases. The process benefits by scoping down the development process to manageable limits (Royce 1970). He furthermore suggests using prototypes to get an early simulation of the final product. In addition, he presents first reflections about iterative development, feedback and adaption (Larman and Basili 2003; Royce 1970).

In the 1980s, many approaches of incremental software development were presented (Wallis 1984; Swartout and Balzer 1982), such as Boehm’s spiral model (Boehm 1986). The spiral model is a risk-driven approach to software development. According to Boehm, the spiral model resulted from a “*risk-management variant of the waterfall model*” ((Boehm 1986), p. 23). The development process is divided into several circles which comprise steps like risk analysis, circle planning and reviews. As he points out, “*the model holds that each cycle involves a progression through the same sequence of steps*” ((Boehm 1986), p. 24). He concludes that the risk-driven approach is more adaptable than document-driven (waterfall) and code-driven (evolutionary development) approaches (Boehm 1986).

The work of Boehm and others present approaches at that time which led to the occurrence of Rapid Iterative Production Prototyping (RIPP) in the mid-to-late eighties (Buragga and Zaman 2013). This project management methodology was a property of Du Pont Information Engineering Association, the consulting business unit of Du Pont Co. (Margolis 1988). Scott Shultz, who was a manager at Du Pont at that time, states: “*What users tell you they want isn’t always what they really want*” ((Margolis 1988), p. 110). Direct customer collaboration led to a definite plus with clear benefits (Margolis 1988).

James Martin learned about the RIPP methodology from Scott Shultz at Du Pont (Larman 2004). Based on the success of RIPP, Martin promoted the iterative and time-boxed development himself, which led to the creation of the Rapid Application Development (RAD) (Larman 2004). The RAD approach included a quick and iterative delivery with a small team of highly skilled developers (Abbas et al. 2008). In addition, RAD set the basis for the *Dynamic Systems Development Method* (DSDM), which had a wide user audience in north-west Europe such as the United Kingdom, the Netherlands, Sweden, and Denmark.

Several decades of software development and best practices from development approaches and applications in industry ended up in the meeting in Utah, where the most suitable techniques were combined within the manifesto for agile software development.

2.2 After the manifesto

A variety of publications (Dingsøy et al. 2012; Theocharis et al. 2015) consider the evolution of agile software development after 2001. Diverse theoretical approaches and numerous points of view on the manifesto, the practices used, the implementation of the principles, or human involvement have led to a wide range of publications within the last 16 years.

Dingsøy et al. (2012) present a literature review of the evolution in the agile community. They present an overview of 1551 papers on agile development until 2010. This reflects the still increasing interest in adapting agile methods whose applications are considered in many different contexts and combinations.

Theocharis et al. (2015) conducted a literature study in 2015 resulting in a variety of different development approaches. The identified approaches often come from a combination of traditional approaches and agile methods. They called those combinations hybrid approaches and identify the method Water-Scrum-Fall (Theocharis et al. 2015) as the most widespread one. Water-Scrum-Fall is the combination of Scrum and the waterfall model. This model is seen as one approach to the adaption of agile elements into traditional development approaches.

Kuhrmann et al. (2017) set up the HELENA study, an international study on the use of Hybrid dEveLopmENt Approaches in software systems development¹. The HELENA study started in 2017 to figure out how hybrid approaches have spread into the development of software worldwide. The HELENA team consisting of more than 80 researchers from all over the world investigates the distribution of hybrid software and system development in practice. The HELENA study is ongoing research and final results have not been presented yet.

In 2002, Abrahamsson et al. (2002) published a literature review aiming at the analysis of agile methods. They analyzed ten methods such as Extreme Programming and Scrum which are characterized as “being agile” with respect to the authors’ definition. Furthermore, they compared the similarities and differences of these methods. In contrast to our approach, Abrahamsson et al. (2002) based their research on publications instead of asking the original contributors whether the agile teams are “agile” according to the original contributors’ understanding.

Pathak and Saha (2013) reviewed different agile methods and compared them with conventional process methods. Furthermore, they analyzed pros and cons of applying these agile processes to research projects. They also pointed out difficulties in adapting agile methods. The contribution focused on existing literature in that field instead of presenting the thoughts of those who started “thinking agile”.

In 2002, Highsmith (2002) described the most important principles of agile development from his point of view, such as delivering value to the customer and focusing on individual developers and their skills. He also presented all major agile methods such as Scrum or Extreme Programming. Highsmith (2002) proposed a method of transforming a company into a truly agile organization.

3 Research method

This section presents the data collection consisting of a survey, an interview study, a workshop with one original contributor, and data provided by the survey participants.

3.1 Research questions

The aim of this study is to preserve the initial thoughts of the original contributors of the manifesto and to elicit their viewpoint of the future of the manifesto and agile development in general. We are interested in answering the following research questions:

RQ 01: *What influences and intentions led to the manifesto for agile software development?*

This question focuses on the agile practices and methods that had an influence on the manifesto. Furthermore, it will clarify the original contribution to the manifesto.

RQ 02: *What do the original authors think about the manifesto today?*

This question examines if there are parts of the manifesto the original authors would skip or change today. It helps to clarify whether the interpretations of the manifesto are still valid or represent the original ideas.

RQ 03: *What do the original authors think about the manifesto's future?*

This question deals with the necessity of new concepts and adjustments of the manifesto. According to the initial idea of being “agile”, it is necessary to learn and adapt according to the given context. Therefore, it is important to rethink the original ideas and to examine their influence on today's software development in practice.

RQ 04: *What do the original authors think about the future of agile software development?*

This question focus on the future directions of agile software development. Furthermore, it presents necessary steps to preserve the origin of agile in the future.

3.2 Data collection

The data collection and analysis bases on four different data sources, which are presented in this section. An overview can be found in Table 1. The survey and the interview study were set up at the same time with the same goal to preserve the original agile mind-set². The workshop resulted from the survey and aims at providing deeper insights into the agile movement. In order to gain a consistent picture of the ideas, we combined the studies.

3.2.1 Survey

Research design. According to Yin (2009), the type of the research questions defines the usage of different research methods. The defined research questions are mainly “what”-questions which are used in exploratory studies. This class of research question are knowledge questions resulting in a clearer understanding of the phenomena (Easterbrook et al. 2008). According to Forza (2002), surveys can be divided into three different groups: exploratory surveys, descriptive surveys, and explanatory surveys.

Our study is founded on a qualitative and a descriptive survey design to obtain knowledge about the manifesto. The survey aims at describing and explaining the phenomenon of the manifesto by using a questionnaire for data collection. A characteristic of survey research is the selection of a representative sample from a well-defined population (Easterbrook et al. 2008). The sample selection is explicitly limited to the 17 original authors of the manifesto because we wanted to obtain first-hand information. With this selection, the sample size is sufficient to represent the population of interest.

Table 1 Overview of data sources

Data Source	Initials of executing authors	Initials of analyzing authors	Number of participants	Further information
Survey	P.H., J.K.	P.H., J.K.	11	Section 3.2.1
Interview study	R.L., J.G.	P.H., J.K.	14	Section 3.2.2
Workshop	P.H., J.K., A.vB.	P.H., J.K.	3	Section 3.2.3
Secondary data	-	P.H., J.K.	-	Section 3.2.4

The survey questions are classified as open questions according to Dresch et al. (2015). The number of questions is limited to five in order to shorten time to answer the questionnaire. This increases the respondent involvement and thus reduces the amount of missing data (Forza 2002). Furthermore, the participants are asked for permission to use the given answers and information anonymously in this publication.

All survey questions are directly mapped to the research questions.

1. Why did you participate in the manifesto for agile software development 16 years ago? (*aims at answering RQ1*)
2. What was your original contribution to the manifesto for agile software development? (*aims at answering RQ1*)
3. Which parts of the manifesto would you skip or change today? (*aims at answering RQ2*)
4. Are there necessary new concepts and adjustments for the manifesto? (*aims at answering RQ3*)
5. What trends will shape the future of agile development from your perspective? (*aims at answering RQ4*)

Forza (2002) recommends conducting a pilot test of the questionnaire with colleagues, industry experts and target respondents. A pilot test was conducted with another researcher and an industry expert. The feedback they provided led to some modifications of the questionnaire to make the questions more comprehensible for the participants.

Execution. The survey was conducted in January and February 2017. We sent an invitation letter and the questionnaire to all contributors of the manifesto. In total, we received 11 answers.

In order to collect the data, we chose two different ways to get in touch with the original contributors and to send them the questionnaire. First, we searched for email addresses on the private blogs and homepages. We directly contacted Martin Fowler and Ron Jeffries via personal email. Both provided answers to the questionnaire. To get in contact with the other 15 participants, Researcher 1 signed up for a recruiter membership on LinkedIn³. With the recruiter profile, it was possible to get in contact with the original contributors and to send them the questionnaire via LinkedIn. The recruiter membership contract was canceled after one month. Within that month, we received answers from 9 participants.

In most of the responses, the items of the questionnaire were directly answered. The length of the provided answers varied from one sentence per question to multiple pages to answer the complete questionnaire.

We were invited by one participant to a Skype™ call to discuss the questions face-to-face. The following interview was a one-hour Skype™ conference in which two researchers and one contributor participated. The interview was recorded by both researchers, and the meeting minutes were subsequently combined.

Two other participants from the study referred to the ongoing project “Agile Uprising Agile Manifesto Review” conducted by Ryan Lockard and James Gifford (2016) (cf. Section 3.2.2).

Data analysis. The collected qualitative data was analyzed using categorization and data coding. According to the classification of Stol et al. (2016), the coding concepts of *Straussian Grounded Theory* were used. We used the presented three coding phases: open

coding, axial coding, and selective coding (Stol et al. 2016; Corbin and Strauss 1990). The interpretive process of open coding generates categories and concepts by breaking down the data analytically. The written answers were chunked into small, logically related phrases and tagged with a code. Each code represents a certain theme, whereas the codes form a hierarchy of codes and sub-codes in the axial coding. In the selective coding, the central categories were defined. This leads to the final result, a formalized body of knowledge (Runeson and Höst 2009). To analyze, categorize and sort the collected data, we used the tagging and coding functionality in the reference management and citation tool Citavi.

3.2.2 Interview study

The interview study was mainly conducted by Ryan Lockard and James Gifford as part of the “manifesto authors review” in a podcast (Lockard 2016) for the Agile Uprising network. The network focuses on the advancement of the agile mindset and global professional networking between leading agilists. The network purely focuses on the advancements of the agile craft removing external influences from sponsors, partners and other organizations (Lockard 2016).

Research design. Since October 2016, Ryan Lockard and James Gifford have performed an interview study (Lockard 2016), asking the original contributors about the reasons for participating in the meeting that ended up with the manifesto.

The study has been designed as an exploratory semi-structured interview. For the interviews, an interview guide has been designed. This guide has been developed in an iterative way. It was pilot-tested and adjusted to the problems which have arisen. The structure of the guide is divided into three parts: (1) before the meeting, (2) at the meeting and (3) afterwards (cf. Appendix A)

Execution. The interview study took place from October 2016 till March 2017. Getting in touch with the contributors was an iterative process. Ryan Lockard and James Gifford first contacted three of them, namely John Kern, Alistair Cockburn and Bob Martin. In the interview, they asked them to share the idea of the podcast with other contributors. This way, another batch of three contributors to continue was created. In total, 14 contributors were interviewed. Two of them have been interviewed in person (John Kern and Alistair Cockburn), the others have been interviewed via Skype™. Skype™ sessions and face-to-face interviews were recorded and afterwards published in the uprising podcast channel.

The average duration for an interview is 50 min, whereas the shortest interview, a mashup of interviews published by Scrum.org, is 17 min and the longest 1 h and 18 min. A complete overview of the conducted interviews with their lengths can be found in Table 2.

Data analysis. The data analysis was conducted by Researcher 1 and Researcher 2 by coding the meeting minutes. The coding process followed the coding concept presented in Section 3.2.1. The results have been analyzed according to the research questions.

3.2.3 Workshop

After having answered the questionnaire, Arie van Bennekum invited Researcher 1 and Researcher 2 for a 1-day workshop to discuss the origins and future directions of the manifesto. He furthermore suggested, to discuss the future of the agile movement in general. He organized a workshop to discuss these topics in detail.

Table 2 Overview of the interviews conducted as part of the agile uprising podcast

Participant	Interviewer	Publication Date	Length	Ref.
Arie van Bennekum	Ryan Lockard, Andy Cleff	2017-02-10	50 min	Lockard and Cleff (2016)
John Kern	Ryan Lockard	2017-02-11	57 min	Lockard (2016a)
Martin Fowler	Ryan Lockard	2017-02-11	55 min	Lockard (2017)
Stephen Mellor	Ryan Lockard, James Gifford	2017-02-19	48 min	Lockard and Gifford (2017)
Ken Schwaber	Mashup of interviews published by Scrum.org	2017-03-06	17 min	Scrum.org (2017)
Jim Highsmith	Ryan Lockard, James Gifford	2017-03-10	43 min	Lockard and Gifford (2017c)
Ron Jeffries	Ryan Lockard, Jason Cusack, Troy Lightfoot	2017-03-10	52 min	Lockard et al. (2017)
Mike Beedle	Ryan Lockard, James Gifford	2017-03-10	49 min	Lockard and Gifford (2016a)
Brian Marrick	Ryan Lockard, James Gifford	2017-03-10	78 min	Lockard and Gifford (2016c)
Jeff Sutherland	Ryan Lockard, Andy Cleff	2017-03-10	45 min	Lockard and Gifford (2017a)
James Grenning	Ryan Lockard, James Gifford	2017-03-10	53 min	Lockard and Gifford (2017b)
Andy Hunt	Ryan Lockard, James Gifford	2017-03-10	66 min	Lockard and Gifford (2016b)
Alistair Cockburn	Ryan Lockard	2017-03-11	50 min	Lockard (2016b)
Robert Martin	Ryan Lockard, James Gifford	2017-03-31	48 min	Lockard and Gifford (2016)

The workshop took place in Oktober 2017 in Rotterdam, Netherlands. The workshop was separated in two sessions of 2, 5 h each. In the first session, Researcher 1 and Researcher 2 presented the results from the survey and discussed the findings with Arie van Bennekum. In the second session, Arie van Bennekum presented his view on the future of the agile movement and the importance of agile transformation. Researcher 1 and Researcher 2 wrote meeting minutes and merged them after the workshop.

The data analysis was done by Researcher 1 and Researcher 2, according to the previously identified codes from the data analysis of the survey.

3.2.4 Secondary data

Some participants of the survey referred to previously held presentations at conferences, and interviews about the manifesto in previous publications. We used this information as a secondary data source. Therefore, we also transcribed the talk “Agile is Dead” from the conference “GOTO Amsterdam” given by Dave Thomas (2017) in 2015. Furthermore, we included an interview with Kent Beck (2011) from 2011. To analyze, categorize and sort the collected data, we used the tagging and coding functionality in the reference management and citation tool Citavi. The secondary data sets are limited to the recommendations given by the survey participants.

3.3 Threats to validity

In accordance with Wohlin et al. (2012), we consider four commonly used criteria for validity.

Construct validity. The construction of the questionnaire aims at preventing a wrong interpretation of the questions. To minimize the risk of misunderstandings, a pilot test was conducted and a short explanation was given to the participants. The semi-structured Skype™ interview was based on the questionnaire.

All conducted interviews were guided by pre-defined open questions (cf. Appendix A). They all started with a description of the interviewee’s background and the reason for attending the meeting in Snowbird in 2001. The focus of the interview is

set on the meeting itself. Furthermore, it gives an outlook on agile development in future.

For data triangulation, we extended our study with data sources, which were provided by participants of the survey. This data includes a presentation at a conference (Thomas 2017) and an interview (Denning 2011).

Recall bias may have led to misinterpretations of statements of both data sources. We reduced this threat by interpreting and discussing the results with at least two researchers.

Internal validity. The internal validity is significant when causal relations are examined (Wohlin et al. 2012). This was not relevant for our mainly descriptive survey (Wohlin et al. 2012). However, we have to mention that we promised the participants to anonymize the answers from our survey. These statements are therefore marked as “Anonymous” in this publication.

External validity. The results of the survey may not be generalized. However, the findings could be interesting for other people in order to learn about the manifesto. Nowadays, the agile mindset is spreading into all businesses and industries. It could be a benefit to know the basics and the origin of the agile mindset. Of course, the results of our survey are subjective impressions of 11 of the 17 authors.

The result of the interview study do not necessarily represent the opinion of all original authors. But we do not claim to generalize the results. We only present a subjective overview of the meeting in Snowbird and future directions of the manifesto.

Conclusion validity. We try to retain the original agile knowledge for people who want to learn more about the original contributor’s view on the manifesto for agile software development. We do not draw conclusions from the acquired data.

Reliability. The main goal of reliability is to minimize errors and biases in the survey. We took several steps to improve reliability. The questionnaire was pilot-tested by a colleague and an industry expert. The questions were explained and additional information was given.

Reliability may be affected by answers that are not provided. The overall picture of the manifesto may be incomplete due to the missing opinions from six original authors.

Another influence is the phenomenon of “recall bias”. In the dictionary of epidemiology, recall bias is defined as “systematic error due to differences in accuracy or completeness of recall to memory of past events or experiences” ((Porta et al. 2014), p. 240). Both the meeting and the creation of the manifesto were conducted 17 years ago. Some interviewees mentioned that they could not remember every single detail from the meeting.

Yet another factor that may influence reliability are the researchers themselves, by misinterpretation and errors in transcriptions. We tried to avoid the “halo effect” (Porta et al. 2014), which would influence the interpretation of the findings by the previously gained knowledge. The fact that two researchers were conducting the survey leads to reciprocal control.

4 Results: past, present and future of the agile manifesto

4.1 Research question 1: influences and intentions

What influences and intentions led to the manifesto for agile software development?

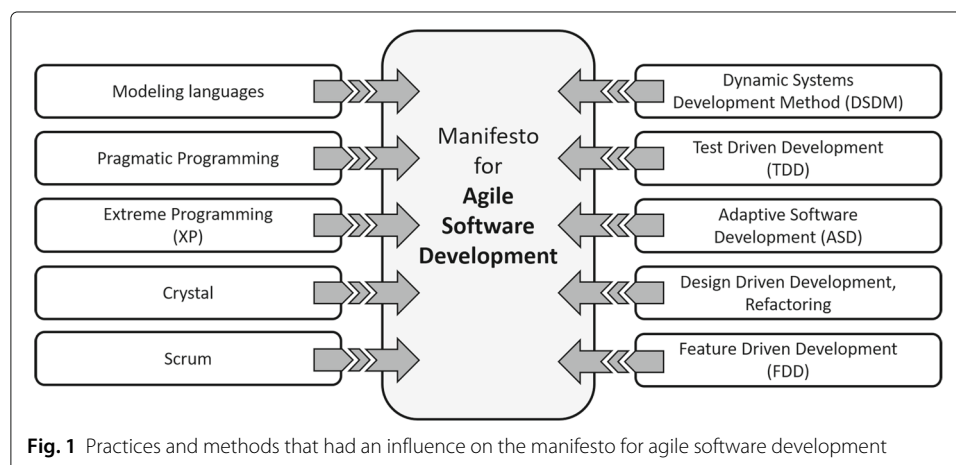
In 1999, Kent Beck invited to a special conference he called the XP leadership conference. Bob (Robert) Martin, Martin Fowler, Ron Jeffries, Ward Cunningham, Kent

Beck and several others attended this conference (Lockard and Gifford 2016). At the conference, they talked about lightweight processes, realizing that they needed to set up another meeting in order to focus on a broader approach for better software development (Lockard et al. 2017).

In 2001, they set up a meeting to find more effective development methods. They wanted to search for common ground (Lockard and Cleff 2016; Lockard and Gifford 2016b). Some of the participants had developed their own working methods and lightweight processes. Highsmith (2017c) did consulting and trainings in structured methodologies and realized that his way of working varied from what he taught. Hunt (2016b) realized that all his clients were working exactly the same (wrong) way and had the same problems in their projects, such as dissatisfied customers or broken deadlines. At that time, Thomas and Hunt published their book “The Pragmatic Programmer: From Journeyman to Master” (Hunt and Thomas 2000).

When the original contributors of the manifesto met in 2001, they tried to identify areas the used methodologies had in common (Lockard and Gifford 2017b). At the beginning of the meeting in 2001, everyone gave a short presentation of his development practice. The echoes of the participants in the manifesto and the principles are still perceptible. Jeffries (2017) describes it as a general feeling that they had similar methodologies, and gave them a higher-level guideline to follow. Highsmith (2017c) mentions that the success of the manifesto is based on the fact that they focused on the similarities of the methodologies for software development. In his opinion, this is the reason why agile is still valid. It is not based on processes but on values and principles.

The results of our study are shown in Fig 1. It visualizes the practices and methods that had an influence on the manifesto for agile software development. Arie van Bennekum pointed out, “*they all share two common treats: the aim to diminish bureaucracy and a focus on valuable deliverables*” (Lockard and Cleff 2016). The following section describes the methods which had an influence on the agile manifesto. The statements of the originators shall help to sharpen the reader’s understanding that there are more methods than SCRUM and XP.



4.1.1 *Dynamic Systems Development Method (DSDM)*

DSDM, first published in February 1995, is an agile framework developed and supported by the DSDM Consortium for dynamic, non-predictable projects. It focuses on early delivery in an iterative development process with close customer collaboration. Nowadays, it also includes project management (the Agile PM Framework). It consists of seven phases, for instance preparation of a project and deployment to the customer. One practice of DSDM is “time-boxing”. The project is sliced into time-boxes which could themselves be sliced into smaller time-boxes. Requirements are prioritized per time-box (DSDM Consortium 2018). DSDM has its origin in the RUGs, the “Rapid Application Development user groups”. Therefore, the Intellectual Property of DSDM is a community owned Intellectual Property.

Arie van Bennekum mentions that at the time they were “*writing the manifesto, the other guys receive DSDM as quite heavy, [and] it did not get that much attention in the US*” (Lockard and Cleff 2016). This is also apparent in Mike Beedle’s statement that he “*didn’t know DSDM*” (Lockard and Gifford 2016a) at that time. Also Jeff Sutherland mentions that he “*did not know anything about DSDM, [and] [...] just met Arie the first time at the meeting*” (Lockard and Gifford 2017a). Alistair Cockburn remembers that “*Arie van Bennekum had to describe DSDM*” at the beginning of the meeting.

4.1.2 *Test Driven Development (TDD)*

TDD helps to control the software development process by using tests. The procedure is cyclic and starts by writing a test, which consequently fails in the first attempt due to missing code. In the next step, the developer implements as much code as necessary in order to fulfill the test criteria. Afterwards, code and tests get refactored. Kent Beck presented this incremental method in detail in 2003 (Beck 2003). TDD was part of the XP practice of test-first, which was already presented in 1999 (Copeland 2001).

James Grenning states that people “*are somehow religious about*” (Lockard and Gifford 2017b) TDD. As he points out, it “*is more like being an engineer. It’s a problem solving technique one should use when it is appropriate*” (Lockard and Gifford 2017b). Ron Jeffries mentions that for refactoring it is necessary to implement the TDD principles and to be able to ship it within one increment (Lockard et al. 2017). Bob Martin refers to his first pair programming and Test Driven Development session with Kent Beck. He admits that he has “*never seen something like this before, [...] [and he has] never seen anybody writing code in this crazy, nutty, high-speed style*” (Lockard and Gifford 2016).

4.1.3 *Adaptive Software Development (ASD)*

In 2000, Jim Highsmith published his book “Adaptive Software Development” (Highsmith 2000). He mentions that he was writing the “*book and there was a chapter in there, about complexity theorem and systems*” (Lockard and Gifford 2017c). He remembers that he “*started looking into that, as a way of really [...] become more adaptive. That’s where adaptive software development came from.*” (Lockard and Gifford 2017c). ASD provides a collaborative approach to manage complex systems by an adaptive culture in which change and uncertainty are assumed to be the natural state. ASD builds a frame containing only a few guidelines. It consists of speculating, collaborating and learning. Speculating includes the planning process and collaborating aims towards implementation. Teamwork is one key characteristic of the

collaborating phase. Learning cycles ensure quality and help in integrating so-called “learning loops”.

4.1.4 Design Driven Development (D3), Refactoring

D3 aims at creating innovative requirements to build better solutions. It is based on the following assumptions: Design fulfillment is very important for success. Design is an event randomly occurring during the conception process. This is the key to innovation. The best design is always the result of continuous facilitation and refinement. In 1999, Martin Fowler published his book “Refactoring” (Fowler 1999) to improve the design of any existing code. He states that badly designed software can be reworked into a good one by refactoring the code continuously. He provides principles of refactoring which go hand in hand by setting up required tests to validate the refactored code (Lockard 2017).

Martin Fowler concludes that *“there are still the very basics of refactoring which are not understood by people”* (Lockard 2017). He is disappointed with that fact because, for him, it *“was the most mind-blowing thing”* (Lockard 2017).

4.1.5 Scrum

Scrum itself is not a software development technique, but a framework for managing various kinds of projects. Scrum provides the separation of the whole project into sprints (Schwaber 2004). A sprint is a defined period of time in which to implement a given set of requirements and make them ready for potential delivery. It consists of several core activities, artifacts and roles which need to be extended by suitable practices. The core activities describe meetings during the sprints, like a daily Scrum or a retrospective (Schwaber 2004). The widely used artifacts comprising a product backlog consist of all requirements which have not been implemented of the project, the sprint backlog which defines all tasks for the ongoing sprint, and a product increment which is the sum of all already finished product-backlog-elements (Schwaber 2004). There are mainly three roles involved: the Product Owner defining the requirements, the development team fulfilling the Product Owner’s requests and the Scrum Master helping to create a good atmosphere for the team and to preserve agility (Schwaber 2004).

Ron Jeffries mentions that *“Jeff Sutherland had used Scrum since the middle 80s”* (Lockard et al. 2017). Jeff Sutherland mentions that he *“hired Ken [Schwaber] for a huge banking project”* (Lockard and Gifford 2017a) and *“wrote a paper 1995 [with him] to kick it off”* (Lockard and Gifford 2017a). He emphasizes that he was *“driving Scrum before, starting from 1993”* (Lockard and Gifford 2017a). His intention was to *“teach the style of Scrum to help the people to improve and accelerate”* (Lockard and Gifford 2017a). Mike Beedle was the first adopter of the Scrum method. He mentions that using Scrum *“was like a miracle drug”* (Lockard and Gifford 2016a). He remembers that they had a *“project which was nearly three years late but [...] saved the project with Scrum”* (Lockard and Gifford 2016a). Arie van Bennekum points out that nowadays people do often not *“know that there were more agile methods than Scrum”* (Lockard and Cleff 2016).

4.1.6 Crystal

Crystal, developed by Alistair Cockburn, is a family of software development methods in agile development. Each “member of this family” usually has a color, with the

basic variant being called “Crystal Clear”. Crystal is based on a couple of principles like passive knowledge transfer, continuous delivery, frequent releases and automated testing. Crystal is configured for one project and needs to be adapted for another one. Cockburn (2005)

Alistair Cockburn came up with crystal in the 90s. He mentions that he realized that *“different projects need different things and [...] [it is necessary to] characterize a situation which calls for a certain type of a behavior”* (Lockard 2016b). To distinguish the situations, he *“used crystal as a way to separate them out”* (Lockard 2016b) and assigned different colors to them. However, Jeff Sutherland mentions that he *“never had any exposure to Crystal”* (Lockard and Gifford 2017a). In contrast, Mike Beedle states that he had *“heard about crystal from Alistair”* (Lockard and Gifford 2016a). He further admits that there were only *“few instances of crystal”* (Lockard and Gifford 2016a). In addition, Ron Jeffries maintains that *“Crystal never seemed as a method that is competing in any way”* (Lockard et al. 2017).

4.1.7 Extreme Programming (XP)

In 2000, Kent Beck published his first edition of his book “Extreme Programming explained” (Beck 2000), which marked the starting point for the success of XP. In XP, several practices are described. The most famous and widespread one is pair programming, besides small releases and continuous integration.

Ron Jeffries remembers that Kent Beck invited him *“to help coaching, which turned out to be the first Extreme Programming installation”* (Lockard et al. 2017). He mentions that it was the effort of Kent Beck who put together the development process, defined *“what should be done, and named it Extreme Programming”* (Lockard et al. 2017). He further argues that one reason why XP became well-known is based on the fact that Kent Beck did not brand mark it (Lockard et al. 2017).

Bob Martin got in touch with XP, when he *“stumbled across Kent Beck’s writings on Extreme Programming”* (Lockard and Gifford 2016). He recognizes that *“this is great, except for that ridiculous testing thing, the whole notion of test first, that’s nonsense, but the rest of it, the rest of it sounds pretty good”* (Lockard and Gifford 2016). Bob Martin recalls that they *“made a mistake in the early days of XP, when we put that high emphasize on pair programming”* (Lockard and Gifford 2016). He emphasizes that *“in realistic agile teams it happen less 50% of all time”* (Lockard and Gifford 2016).

4.1.8 Pragmatic programming

In their book “The Pragmatic Programmer: From Journeyman to Master” (Hunt and Thomas 2000), Dave Thomas and Andy Hunt defined software development as a craft. They give special insights into the topic, and tips based on their experience. First, they advise the programmer to care about the craft. A lot of tips represent what later reappears in the agile methodologies. Two examples are tip 16 *“Use prototype to learn”* and tip 62 *“Test early, test often, test automatically”* (Hunt and Thomas 2000). Finally, they outline a pragmatic way for the efficient, profitable development of high-quality products based on easily understandable tips.

Andy Hunt and Dave Thomas were looking at pragmatic programming which *“is considered not really as a methodology but just more common sense”* [Anonymous]. The word *“methodology was a bit of a dirty word at the time and we were understandably cautious”*

[Anonymous]. They both *“felt the role as the Pragmatic Programmers was to help defend the interests of the ordinary developer”* [Anonymous].

4.1.9 Modeling languages

Steve (Stephen) Mellor created a modeling language that was the forerunner to UML (Shlaer and Mellor 1988). To model components, sequence diagrams, collaboration diagrams, state-transition diagrams, and a combination of state charts and activity diagrams are used. The modeling language then seamlessly translated the component models into code. Code components are compiled into an executable component that is deployed (Mellor and Balcer 2002).

James Grenning remembers that *“Steve was model driven, so he really wanted to build a model and then we can convert them out into code”* (Lockard and Gifford 2017b). He further points out that *“the abstraction that Steve brought in was executing. It doesn’t matter if it is code or model or whatever”* (Lockard and Gifford 2017b).

Steve Mellor emphasizes that it is essential *“to think about a model is some high-level abstract view of something that takes away certain details, such as processors, data structures, that kind of stuff”* (Lockard and Gifford 2017). He identifies the *“underlying reason why peoples say [that models are bad] [...], because they think that models are just sketches and then you write code. And that was not the view that I took, not at all. The model was in fact the code, it just had an abstractive way, sort of relevance need to be filled out separately, not added, but filled out worked on separately”* (Lockard and Gifford 2017).

4.1.10 Feature Driven Development (FDD)

FDD was first described by Jeff De Luca. The FDD process was influenced by Peter Coad and Jon Kern through their collaboration in writing the book *“Java modeling in color with UML”* (Coad et al. 1999). The process of FDD is model driven, and in the development, this is done in short iterations. The process is built on a core set of best practices. The main idea is to decompose complex functions until each sub-problem is small enough to be called a feature.

Speaking about his role in the meeting, John Kern mentions that he *“was nobody, but Peter Coad was, [and the reason why he participated] is because [...] [he] was working with Peter Coad, publishing books on Feature Driven Development”* (Lockard 2016a).

4.2 Research question 2: Today’s views on the manifesto

What do the original authors think about the manifesto today?

4.2.1 The manifesto was a “big deal”

Alistair Cockburn (2016b) compares the meeting in Snowbird and the creation of the manifesto to the NATO Conference of software engineering in 1968. However, most of the authors in 2001 thought that their manifesto would not be as successful as it has since become. Jim Highsmith (2017c) admits that the success of the manifesto for agile software development took off faster than all authors of the manifesto anticipated. Mike Beedle (2016a) adds that he never foresaw the adaption of agile development practices in other contexts such as leadership or sales. As he recognizes in retrospect, the implications of the manifesto were getting out of control. It took James Grenning (2017b) a couple of years to realize that it was such a “big deal”. He mentions that none of the original authors

actually thought that somebody would care about the manifesto. Steve Mellor emphasizes that in retrospect *“it makes sense, [but] at that time, I was surprised”* (Lockard and Gifford 2017). In hindsight, he mentions *“that the manifesto galvanizes [...] [those], who wanted to [...] code and not to document [and] who wanted [...] to talk to users separately from the structures of the organization. And as a result, this become a thing”* (Lockard and Gifford 2017).

4.2.2 *The need for changing the wording of the manifesto*

The manifesto's wording is universal. This is one reason that led to the tremendous success of the manifesto and makes it still applicable and successful today [Anonymous]. One participant mentions that *“in total, everybody is still pretty happy with the manifesto”* [Anonymous]. As the participant points out, the concepts behind the manifesto remain solid. Bob Martin concludes that *“the original values and the original ideas have survived”* (Lockard and Gifford 2016). Nevertheless, Andy Hunt (2016b) emphasizes that it is crucial to follow the statements as written in the manifesto in order to generate successful software development. Bob Martin points out that the manifesto is not a *“living document”* (Lockard and Gifford 2016). He also emphasizes that he would not change anything in the original wording of the manifesto. He admits that there may be new concepts and adjustments for industry, but the manifesto is understood as a document in time and not an ongoing recipe. In retrospect, Ron Jeffries would add one more sentence to the manifesto. At the end he would write: *“And we really mean it, particular working software”* (Lockard and Gifford 2017a). Furthermore, Arie van Bennekum would like to change the wording of the manifesto by replacing the word *“software with solutions”* (Lockard and Cleff 2016).

4.2.3 *The difference between “doing agile” and “being agile”*

Bob Martin mentions that many organizations think they are “doing agile” when they are not (Lockard and Gifford 2016). Arie van Bennekum points out that there is a huge difference between “doing agile” and “being agile”. He defines “doing” as *“just some rituals but without significant change to support the real agile approach as end-to-end, business integration, value focus, and team empowerment”* (Hohl 2017). In contrast, he defines that *“being agile means [that] the individual, team, [and] organization have changed”* (Hohl 2017). This is *“when they overcome the old paradigms”* (Hohl 2017). These old paradigms hinder an agile approach and the agile transformation. Impediments are *“often hidden in management layers and bureaucracy”* (Hohl 2017).

4.2.4 *Common misinterpretations of the agile manifesto*

Bob Martin (2016) discovers the problem that the adoption and the understanding of the agile manifesto is different. Bob Martin (2016) points out that the misinterpretation and false implementation of agile is caused by politics, imagination, and economic interest. Martin Fowler (2017) identifies missing support from the business side due to missing knowledge. He mentions that this leads to the unpleasant effect that the agile ideas are often misinterpreted. Furthermore, misinterpretation leads to pressure on the programmers instead of creating a motivating and productive working environment. Forcing programmers to change is not possible and will end up in developers resignation [Anonymous]. Bob Martin (2016) remembers that they also made a mistake in the early days of XP when they put high emphasis on pair programming by pushing it too hard.

He emphasizes that the time for developing software in pairs should be kept at a reasonable amount of time (Lockard and Gifford 2016). In realistically minded agile teams, it is always a trade-off between specific needs and context. Another participant mentioned that a context-specific selection of practices is necessary [Anonymous]. Andy Hunt adds that *“the rationale behind many of the practices is how to get quick and reliable feedback on what you are doing”* (Lockard and Gifford 2016b). Even though there is a lot of misinterpretation identified, Bob Martin (2016) positively argues that the idea behind the manifesto is still valid.

4.2.5 Poor implementation of agile

When Kent Beck invented XP, he intended to make the world safer for programmers. Ron Jeffries believes that nowadays, *“agile is not making the world safer for programmers”* (Lockard et al. 2017). One participant [Anonymous] identifies management pressure as one reason for a lot of careless software “craftsmanship” on the developer’s level. Jeff Sutherland points out that there is a *“lot of sloppy software craftsmanship [because of] management pressure”* (Lockard and Gifford 2017a). This often creates a huge mess since the agile approach is not tailored to the specific context. Jeff Sutherland (2017a) mentions that there are many teams outside without working software. According to Arie van Bennekum, often *“teams deliver [software] and then use the following sprint for test”* (Hohl 2017). In addition, Mike Beedle is disappointed because many people *“maybe just have misunderstood [the concepts of agile] and doing [it] on the cheap”* (Lockard and Gifford 2016a). The problem he points out is that *“people want solutions and people want an execution plan”* (Lockard and Gifford 2016a). Organizations which want to retain their hierarchy often try to introduce Scrum by the book to speed up development [Anonymous]. Furthermore, Mike Beedle states that *“agile is about the understanding, what makes sense”* (Lockard and Gifford 2016a). John Kern emphasizes to view agile methods and practices in a more differentiated manner. He states: *“I hope the hack that someone building software that can kill you is not following the same process as someone designing some start up bullshit webpage. Please tell me not to follow the same methodology. Whatever it is. To me that’s the difference. Rightsizing the mind-set and be as agile as you could be in your work. Fail safe better be developed with a much more strict process.”* (Lockard 2016a)

Another possible reason why agile is poorly implemented is identified by Ron Jeffries (2017). He claims that agile has *“become a management approach”*, and it is sold at the highest management levels. Arie van Bennekum emphasizes that agile *“has not only become a management approach”* (Hohl 2017). He points out the unpleasant fact that agile is *“used by managers because they score points at the board of directors, shareholders and others who have no idea, but have heard the word and trend”* (Hohl 2017). Brian Marick (2016c) mentions that before the agile movement, traditional programmers and the management were separated. He mentions that *“in agile [...] both cultures come together”* (Lockard and Gifford 2016c). However, *“today they split again”* (Lockard and Gifford 2016c). This trend can be seen at *“agile conferences, where the programmers do not go to the conferences anymore”* (Lockard and Gifford 2016c).

4.2.6 A lot of people claim to be agile

Ron Jeffries (2017) thinks that the *“blood has been squeezed out of agile”*. Since “agile” is so popular, everyone wants to ride along. Another participant [Anonymous] emphasizes

this by stating that a lot of people believe they are doing agile with a 3-h stand-up meeting and a backlog. According to Andy Hunt (2016b), this is how they pretend to be agile. Mike Beedle (2016a) and Arie van Bennekum (2017) conclude that agile approaches often do not work, because the only reason to implement them is that *“it is fashionable to be agile”*. They Lockard and Gifford (2016a); Lockard and Cleff (2016) identify the problem of too many people professing to know about agile and that many people misinterpret the basic claim of the manifesto. They mentioned the example of documentation.

The manifesto does not prohibit documents or processes. The users should just value them less than individuals or working software. Mike Beedle (2016a) mentions that even Scrum is based on processes and has a backlog, which is a document. Arie van Bennekum mentions that it is important to say that *“there is no basic issue against documentation”* (Hohl 2017). He points out that the *“basic issue is [the use of] documents [which is] one of the old paradigms”* (Hohl 2017). He recommends the use of “Information Radiators”. *“Information Radiators do cover information and can be perceived as documentation or a document”* (Hohl 2017). From his point of view, the way to document like this leads to the fact that the document or documentation *“is not the hindering product as in old processes”* (Hohl 2017).

4.2.7 Preserve the agile origin

Martin Fowler (2017) mentions that a lot of people actively working in an agile way are not satisfied. Furthermore, Ron Jeffries (2017) believes *“that all good ideas get watered down in their fashion. Over time, when the idea is getting popular, it is just watered down the ripples lower and lower. Some people get it, some people not”* (Lockard et al. 2017). The original manifesto is developer-focused. Martin Fowler (2017) mentions that *“the real benefits are [...] [still on the] technical level”*. Therefore, the manifesto does not claim to support executives and business owners [Anonymous].

4.2.8 Agile certifications and agile coaches

What comes along with the spreading of the agile ideas are certifications like the *Scrum Master*. Bob Martin denotes the idea of certification introduced by Ken Schwaber as a *“great service [...] [and] marketing scheme that just opened the flood gates”* (Lockard and Gifford 2016) for the agile movement. Its drawback is a massive number of people who attend a 2-day “agile” course. In the end, they get a certification which makes them agile in their opinion (Lockard and Gifford 2016). There is a big issue with certified agile coaches who are unable to overcome stereotypical thinking [Anonymous]. According to Arie van Bennekum (2016), the agile movement was getting so big that there was a need for coaches to help organizations to overcome the old paradigms and replace them by new ones in order to be agile. He also mentions certifications by non-profit organizations as a benefit (Lockard and Cleff 2016). According to Brian Marick (2016c), Agile and Scrum are a *“brand rather than a mindset”*, and the craft of software development has been lost.

4.2.9 New agile trends arise

Since 2001, the original ideas of the manifesto have become more and more commercialized and different trends have arisen. New trends like Scrum of Scrums (SoS) or the Scaled Agile Framework (SAFe) are common talk. Ron Jeffries mentions that *“the agile ideas are [...] often adopt[ed] as roots”* (Lockard et al. 2017), which forces the development into frameworks like SAFe. According to Ron Jeffries, *“it is depressive”*

(Lockard et al. 2017), because programmers *“have to do [the same] stuff every week”* (Lockard et al. 2017). He further calls this the *“dark scrum”* (Lockard et al. 2017). Arie van Bennekum points out that *“scaling Agile is [...] a non-issue”* (Lockard and Cleff 2016). He elucidates that the success of the agile scaling frameworks resulted from the misbelief that *“there was no full delivery method in the time when the agile manifesto came out”* (Lockard and Cleff 2016). He mentions that *“there was DSDM, including full delivery of needed business value and the required business change”* (Lockard and Cleff 2016). As he defines, *“Agile is about delivering better business value earlier with a better internal quality”* (Hohl 2017). Sticking to the old paradigms and processes will not help. He points out that *“being Agile makes you [and your company] future-proof”* (Hohl 2017). Mike Beedle confirms this, saying that *“the rate of change in the world has so increased”* (Lockard and Gifford 2016a). He furthermore points out that *“this is a world of rapid change”* which needs *“new way[s] of management techniques”* (Lockard and Gifford 2016a).

4.2.10 Take aways for research question 2:

Martin Fowler mentions that it is important to remember the *“long list of people, who were pushing agile in that time”* (Lockard 2017).

Further important takeaways for current view on the agile manifesto:

- The success of the manifesto for agile software development took off faster than all authors of the manifesto anticipated.
- The majority of the authors would not change the wording of the manifesto.
- It is a difference between *“doing agile”* and *“being agile”*.
- Agile methods and practices are often poorly implemented.
- A lot of people claim to be agile, because it is fashionable to *“be agile”*.
- It is important to preserve the origin of the agile manifesto.
- Agile certifications and agile coaches, both a blessing and a curse.
- A world of rapid change needs new ways of management techniques.

4.3 Research question 3: views on future of the agile manifesto

What do the original authors think about the manifesto’s future?

As we figured out, there are two different views on the future of the manifesto. One supports the adaption of the manifesto to different domains in and outside the area of software engineering. The other one takes a critical stance on it. This latter viewpoint is based on the ongoing commercialization of selling *“agile”* and a loss of the sharpness of the core ideas. In addition, one participant emphasizes the first sentence of the manifesto where the very first line says *“We are uncovering better ways of developing software by doing it and helping others do it”* (Beck et al. 2001). What Steve Mellor likes to see in future directions of the manifesto is the awareness on *“how they could make their process, their abstraction in turning more towards reality”* (Lockard and Gifford 2017).

One participant mentions that the manifesto sets a standard of values and principles most people ignore. He does not see any sense in updating it. Arie van Bennekum refers to the fact that *“too many people use the word without understanding”* (Hohl 2017). Furthermore he mentions that *“too many of so called agile experts did not read the manifesto or investigate other methods than Scrum”* (Hohl 2017). For the future of agile software

development, Jeff Sutherland states that he does not “*know any better ways to do software than scrum*” (Lockard and Gifford 2017a), but he is not focusing on the future trends right now.

However, there have been some thoughts about a rework or version 2 of the manifesto. Bob Martin (2016) does not believe that any of the original authors are interested in doing that. In 2011, though, Kent Beck published his “beyond agile manifesto” (Denning 2011):

In 2011 Kent Beck proposed that even a great manifesto needs to evolve and that this might be the next step:

Team vision and discipline over individuals and interactions
(over processes and tools)
Validated learning over working software (over comprehensive documentation)
Customer discovery over customer collaboration (over contract negotiation)
Initiating change over responding to change (over following a plan)

The importance of team vision and discipline is also acknowledged by Arie van Bennekum. He points out that “*the success in agile comes with the quality and discipline you apply using the agile rituals and concept*” (Hohl 2017). Ron Jeffries (2017) mentions the loss of the sharpness of the core ideas by spreading agile and selling it to large enterprises.

4.3.1 Take aways for research question 3:

- Two different views on the future of the manifesto exist.
- Some authors stop focusing on the future trends of the agile manifest.
- Some authors support the distribution of the agile mindset into new fields of application.

4.4 Research question 4: the future directions of agile software development

What do the original authors think about the future of agile software development?

4.4.1 Scrum is not the end of the road for being agile

Steve Mellor had “*been expecting agile to reach the top of the hype curve, for a good 5 years now. And there are two hype curves, the marketing hype curve, and a very similar curve which has to do with the underlying technology*” (Lockard and Gifford 2017). Andy Hunt states that “*Scrum is not the end of the road for being agile, neither XP*” (Lockard and Gifford 2016b). Mike Beedle (2016a) identifies the agile ideas as a good starting point for further improvements. In a changing world, the evolution of agile software development is still going on. Brian Marick mentions that “*the agile train keeps moving*” (Lockard and Gifford 2016c).

4.4.2 Stay flexible and deliver high quality outcomes

Ron Jeffries (2017) hopes that the future of agile comprises continuous innovation and pushing the boundaries. As Arie van Bennekum emphasizes, “*one of the characteristic of Agile is continuous improvement and learning. This means agile will not, cannot, stay as it is*” (Hohl 2017). Organizations want to become more agile, especially from the business view. Increasing business agility will go on for many years,

combined with flexible leadership. Nevertheless, all ideas such as short cycles or a closer collaboration with the customer are based on agile (Lockard and Gifford 2017c). According to one participant, the used agile method itself is not important [Anonymous]. One should focus on selecting the right principles. What helps most, should be used. In addition, Ron Jeffries (2017) would like to return to some values, such as retaining some of the elements from XP that are mostly forgotten in the average installation.

The rapid change in agile adoptions is seen as a problem by another participant. If organizations spontaneously restructure the development process towards an agile development, appoint a Scrum Master and develop differently, this will not work (Lockard and Gifford 2016b). Ron Jeffries (2017) claims that it is the first priority to focus on automatic and reliable software releases. Andy Hunt adds that, *“if you cannot get software automatically and reliably out of your pipeline for continuous delivery, if you cannot do this mechanically, then all the other stuff like talking to users and so on does not matter”* (Lockard and Gifford 2016b). In contrast, Jon Kern states: *“I don’t have to suffer doing stupid stuff and focus on delivering value more quickly. Respecting each other, the client, sustainable pace, I think that will make a better environment. I hope we made it not suck as much as it might.”* (Lockard 2016a)

One participant claims that the level of abstraction slowly increases [Anonymous]. As a consequence, software should become less “precious”. Many of the ills come from the mistaken notion that because the software code was hard to implement, it should be protected against changes [Anonymous]. Instead, it is important to realize that software is ephemeral. He recommends focusing on writing software that is easy to replace.

4.4.3 Be professional

As one participant denotes there is a need for professionalism and software “craftsmanship”. People should be proud of the job they do, tackling it with more personal responsibility and self-respect. *“We don’t have to be hackers, we don’t have to be horrible coders by disappointing people every time we release software. We can be professionals. And that is the attitude I want to see, pushed out from snowbird, from the agile community, into the industry at large, I think we have a really big problem there”* (Lockard and Gifford 2016). Furthermore, one participant [Anonymous] hopes that over time more people will discover the principles of agile and benefit from them. He predicts that agile ideas will permeate business, but he is aware of the fact that this is a slow and uncertain process. Furthermore, Brian Marick (2016c) adds his opinion on the tendency in agile projects to select methods. The ensuing homogeneity is detrimental for the working process. He argues that there is probably untapped potential for making an explicit goal, instead of figuring out, how to make everyone the same. Different people work more productively together [Anonymous].

4.4.4 Take aways for research question 4:

- The agile train keeps moving, Scrum is not the end of the road.
- Maintain flexibility, stop doing stupid stuff and focus on delivering value more quickly.
- Be professional.

5 Discussion

This publication aims at preserving the initial thoughts of the original contributors of the manifesto, to elicit their viewpoint of the future of the manifesto and agile development in general. The results provide deep insights into the history and current status of the manifesto and on agile software development based on the thoughts of the original authors of the manifesto. This publication is separated into four logical parts, the influences and intentions which led to the manifesto, the original authors' thoughts about the manifesto today, the future of the agile manifesto and the future of agile development in general.

The first part provides an overview on agile practices and methods that had an influence on the manifesto. In 2001, various different development processes, methods and practices were in use. The original contributors recognized the need and the potential of lightweight processes. Therefore, a meeting was initiated with interested developers and researchers to discuss lightweight processes. In the meeting, they identified common areas of their used methodologies. It results in a higher-level guideline, summarized in the agile manifesto. Arie van Bennekum pointed out, *"they all share two common treats: the aim to diminish bureaucracy and a focus on valuable deliverables"*. Some authors mentioned that it was the right time and the right people to set up the manifesto [Anonymous]. However, one could ask the hypothetical question, if the manifesto would be existing or in a different shape, if the participants were different. Some invited participants could not take an active part in the meeting. From current viewpoint, the success of the agile movement confirms that this is the right approach. Martin Fowler mentions that it is important to remember the *"long list of people, who were pushing agile in that time"* (Lockard 2017).

The second part of the publication elaborated what the original authors think about the manifesto today. It helps to clarify whether the interpretations of the manifesto are still valid and represent the original ideas. Since 2001, the success of the manifesto for agile software development took off faster than all authors of the manifesto anticipated. However, agile methods and practices are often poorly implemented. In 2002, Abrahamsson et al. (2002) analyzed ten methods such as Extreme Programming and Scrum which are characterized as "being agile". The original authors also emphasize the difference between "doing agile" and "being agile". They further mention that a lot of people claim to be agile, because it is fashionable. They emphasize the importance to preserve the origin of the agile manifesto. Therefore, the majority of the authors would not change the wording of the manifesto.

The third part focuses on thoughts about the manifesto's future and deals with the necessity of new concepts and adjustments of the manifesto. We identified two different views on the future of the manifesto. On the one hand, some authors stop focusing on the future trends of the manifesto. They mention that they do not follow the latest trends in Agile. They state that the agile manifesto shall remain as it is and are good with it. On the other hand, some contributors support the distribution of the agile mindset into new fields of application. Bob Martin (2016) does not believe that any of the original authors are interested in evolving the manifesto. However, in 2011, though, Kent Beck published his "beyond agile manifesto" (Denning 2011). We identified this trend of setting up manifestos, such as the *manifesto for software craftsmanship*⁴ or the *manifesto for async software development*⁵, trying to jump on the bandwagon of success of the original one.

The last part considers the future of agile software development in general. Starting in the area of software development, agile is nowadays spreading more and more into business. Scrum is the most popular agile method (Weber 2015; VersionOne Inc. 2017) and the dominant way of developing software. A lot of “agile on-boarders” identify Scrum as the only method. Other techniques and agile principles are often not considered any more or are simply unknown.

Our study aims at giving an overview of the basics behind the agile manifesto. However, we limit the results to the opinions of the originators. In particular, the statements do not necessarily reflect our personal opinion.

The study revealed, how the agile mindset evolved since 2001. Influence factors, such as commercialization of agile certifications directed the agile mindset towards, what is today. Nowadays, agile has become a buzzword for “having fun at work, while simultaneously performing better than before”. With this publication, we want to increase the awareness for the differences between the original ideas and what it is meant to be today.

The study results emphasize that it is in our hands to create and maintain the future of agile. Therefore, the publication shall help the readers to build up their own picture of agile trends. We encourage the readers to critically rethink the current agile movement.

6 Conclusion

This publication presents results from a survey and an interview study in collaboration with the original contributors of the manifesto for agile software development. Furthermore, it comprises the results from an workshop with one of the original authors. This publication focuses on the origins of the manifesto, the contributors’ views from today’s perspective, and their outlook on future directions. We evaluated 11 responses from the survey and 14 interviews to understand the viewpoint of the contributors.

The original contributors emphasize that agile methods need to be carefully selected and agile should not be seen as a silver bullet. They underline the importance of considering the variety of different practices and methods that had an influence on the development of the manifesto. Furthermore, they mention that people should question their current understanding of “agile” and recommend reconsidering the core ideas of the manifesto.

This publication provides impressions of the manifesto. It reveals the discomfort of some of the contributors with current trends such as certification or commercialization. Furthermore, interesting directions for further evolution of agile development and beyond are identified.

With this work, we try to retain this knowledge for people who want to learn more about the original contributor’s view on the manifesto for agile software development. Understanding and a continuous learning is an important step in mastering the basics of agile.

The publication reveals many avenues for further research. Potential directions could be the influence of certification on the agile mindset. It is necessary to investigate, if employees are doing the agile certifications for gaining knowledge in agile techniques or are they just forced by the agile movement to pass the exam without knowing the basics.

We could expand the presented survey to well-recognized thought leaders in the agile community. They have contributed to agile becoming the main stream. Their opinions should be studied as well, especially their view on the future.

Endnotes

¹ <https://helenastudy.wordpress.com/>

² Both studies were started without knowing of each other. We decided to combine them into one paper.

³ <https://www.linkedin.com/>

⁴ <http://manifesto.softwarecraftsmanship.org/>

⁵ <http://asyncmanifesto.org/>

Appendix A: Interview guide

(1) The first part aims at understanding what the contributors have been doing before the meeting in Salt Lake City. The five main topics are covered by the following questions:

- What were you doing professionally prior to being asked to participate in the manifesto?
- What were your thoughts and feeling about being asked to be part of the manifesto?
- How did doing it at a ski resort affect the overall process? If you were to do it again would you choose a similar location? Why or why not?
- How were you evangelizing your discipline before the meeting?
- In addition, ask about the previous work and education and how it led them to the meeting.

(2) The second part discusses the two-day-meeting in detail:

- Describe from your recollection the setting in the room and how the conversation began and evolved.
- Can you recall how all the different disciplines brought by each individual were received by the others in the room?
- Do you recall any good debates or disputes that led to valuable contributions to the final version of the manifesto?
- Were there any values and/or principles that you proposed the were not selected for the final version? If so what were they and why were they important to you?

(3) The third part reflects the perspective on the state of agile software development after the meeting:

- What are your thoughts on the state of agile 15 years later?
- The goal was to help others in your profession to think about software development, methodologies, and organizations, in new – more agile – ways. Would you say the manifesto has achieved those goals?
- If you were to write the same manifesto today, what would you change (if anything)?
- What has surprised you most in the past 15 years when it comes to the reception and implementation of agile on a global scale?
- Looking back now is there anything you would have done differently at the time of putting together the original manifesto?

- With regards to restoring balance and credibility to the word “methodology” how do you think you have fared thus far?
- What are a few patterns that agile practitioners should focus on to improve the future of agile?
- For those who are not in IT or “agile inclined” how can they implement agile methodology into their regular lives?
- How would you like to see the younger generation of “agilistas” continue your legacy?

Abbreviations

ASD: Adaptive software development; D3: Design driven development; DSDM: Dynamic systems development method; FDD: Feature driven development; IID: Incremental software development; RAD: Rapid application development; RIPP: Rapid iterative production prototyping; SAFe: Scaled agile framework; SoS: Scrum of scrums; TDD: Test driven development; XP: Extreme programming

Acknowledgments

The authors would like to thank the whole network behind the agile uprising podcast for preserving the agile mindset by remaining agnostic of certifying bodies and focusing purely on the advancement of the agile craft. The authors thank all the interviewers for conducting the interviews with the contributors, especially Andy Cleff, Jason Cusack and Troy Lightfoot.

Furthermore, we thank the contributors of the time they invested in answering the questionnaire and in participating in the interview studies.

Funding

No funding sources.

Availability of data and materials

The records of the interviews are available online as part of the Agile Uprising Podcast (Lockard 2016). We cannot provide the surveys' answers and the results from the workshop due to data privacy protection.

Authors' contributions

PH designed the survey, collected and analyzed the data, coordinated the research activities, drafted and reviewed the manuscript. JK participated in the data collection, data analysis and in all quality assessments and revisions of the paper. AvB participated in the survey, and supported the writing process with his expertise in the agile manifesto and agile transformations. He furthermore reviewed the manuscript. RL and JG conducted the interview studies and reviewed the manuscript. JM, MS and KS provided guidance for the study design and reviewed the manuscript. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Author details

¹Daimler AG, Research and Development, Wilhelm-Runge-Str. 11, 89081 Ulm, Germany. ²Leibniz Universität Hannover, Welfengarten 1, 30167 Hanover, Germany. ³wemacity, Weesperstraat 61, 1018 Amsterdam, Nederland. ⁴Contino, 404 N 5th Ave, 10018 New York, NY, United States. ⁵Lean Agile Intelligence, 109 Hunt Club Ln, 19073 Newtown Square, PA, United States. ⁶Reutlingen University, Danziger Str. 6, 71034 Böblingen, Germany.

Received: 6 November 2017 Accepted: 21 October 2018

Published online: 09 November 2018

References

- Abbas N, Gravell AM, Wills GB (2008) Historical roots of agile methods: Where did “agile thinking” come from? In: Abrahamsson P, Baskerville R, Conboy K, Fitzgerald B, Morgan L, Wang X (eds). *Agile Processes in Software Engineering and Extreme Programming*. Lecture notes in business information processing, edn. Springer, Berlin and Heidelberg, pp 94–103
- Abidin FAZ, Jawawi DNA, Ghani I (2017) Agile transition model based on human factors. *Int J Innov Comput* 7:23–32
- Abrahamsson P, Salo O, Ronkainen J, Warsta J (2002) Agile software development methods: Review and analysis. *VTT Publ* 478:107
- Beck K (2000) *Extreme Programming Explained: Embrace Change*. Addison-Wesley, Reading Mass. u.a.
- Beck K (2003) *Test Driven Development: By Example*. Addison-Wesley, Boston Mass
- Beck K, Beedle M, van Bennekum A, Cockburn A, Fowler M, Grenning J, Highsmith J, Hunt A, Jeffries R, Kern J, Marick B, Martin R, Mellor S, Schwaber K, Sutherland J, Thomas D (2001) *Manifesto for Agile Software Development*. <http://agilemanifesto.org/>. Accessed 10 Feb 2017
- Boehm B (1986) A spiral model of software development and enhancement. *ACM SIGSOFT Softw Eng Notes* 11:22–42

- Boehm B, Turner R (2004) Balancing agility and discipline: evaluating and integrating agile and plan-driven methods. In: Proc. ICSE'04. IEEE Computer Society, Washington, DC
- Broy M (2006) Challenges in automotive software engineering. In: Proceedings of the 28th International Conference on Software Engineering. ACM, New York. pp 33–42
- Buragga KA, Zaman N (2013) Software Development Techniques for Constructive Information Systems Design. Information Science Reference, Hershey PA
- Coad P, Lefebvre E, DeLuca J (1999) Java Modeling in Color with UML®: Enterprise Components and Process. Prentice Hall, Upper Saddle River NJ
- Cockburn A (2005) Crystal Clear: A Human-powered Methodology for Small Teams. Addison-Wesley, Boston Mass
- Copeland L (2001) Extreme Programming: HOW-TO. <http://www.computerworld.com/article/2585634/app-development/extreme-programming.html>. Accessed 10 Feb 2017
- Corbin J, Strauss A (1990) Grounded Theory Research: Procedures, Canons and Evaluative Criteria
- Denning S (2011) Innovation: Applying “Inspect & Adapt” To The Agile Manifesto. <http://www.forbes.com/sites/stevedenning/2011/05/04/innovation-applying-inspect-adapt-to-the-agile-manifesto/#150960361804>. Accessed 10 Sept 2017
- Dingsøyr T, Nerur S, Balijepally V, Moe NB (2012) A decade of agile methodologies: Towards explaining agile software development. *J Syst Softw* 85:1213–1221
- Dresch A, Lacerda DP, Antunes Jr JAV (2015) Design Science Research. Springer International Publishing, Cham
- DSDM Consortium (2018) What is DSDM? <https://www.agilebusiness.org/what-is-dsdm>. Accessed 10 Feb 2017
- Easterbrook S, Singer J, Storey M-A, Damian D (2008) Selecting empirical methods for software engineering research. In: Shull F, Singer J, Sjøberg DIK (eds). Guide to Advanced Empirical Software Engineering. Springer-Verlag London Limited, London. pp 285–311
- Forza C (2002) Survey research in operations management: A process-based perspective. *Int J Oper Prod Manag* 22:152–194
- Fowler M (1999) Refactoring: Improving the Design of Existing Code. Addison-Wesley, Reading Mass. u.a.
- Highsmith J (2000) Adaptive Software Development: A Collaborative Approach to Managing Complex Systems. Dorset House, New York
- Highsmith J (2002) Agile Software Development Ecosystems. Addison-Wesley, Boston Mass
- Hohl P (2017) Agile Manifesto - Review and Future Directions (personal communications)/Workshop. Participants: Arie van Bennekum, Philipp Hohl, and Jil Klünder, Rotterdam
- Hunt A, Thomas D (2000) The Pragmatic Programmer: From Journeyman to Master. Addison-Wesley, Boston Mass
- Klünder J, Schmitt A, Hohl P, Schneider K (2017) Fake news: Simply agile. In: Volland A, Engstler M, Fazal-Bagaie M, Hanser E, Linssen O, Mikusz M (eds). (Hrsg.), Projektmanagement und Vorgehensmodelle 2017 - Die Spannung zwischen dem Prozess und den Menschen im Projekt. Gesellschaft für Informatik, Bonn. (S. 187–192)
- Kuhrmann M, Hanser E, Prause CR, Diebold P, Münch J, Tell P, Garousi V, Felderer M, Trektere K, McCaffery F, Linssen O (2017) Hybrid software and system development in practice: Waterfall, scrum, and beyond. In: Bendraou R, Raffo D, LiGuo H, Maggi FM (eds). Proceedings of the 2017 International Conference on Software and System Process - ICSSP 2017. ACM Press, New York. pp 30–39
- Larman C (2004) Agile and Iterative Development: A Manager's Guide. Addison-Wesley, Boston
- Larman C, Basili VR (2003) Iterative and incremental developments. a brief history. *Computer* 36:47–56
- Lockard R (2016) Agile Uprising Agile Manifesto Review. <http://www.agileuprising.com/agile-uprising-agile-manifesto-review/>. Accessed 10 Feb 2017
- Lockard R (2017) Agile Uprising Agile Manifesto Review. <http://agileuprising.libsyn.com/manifesto-co-author-interview-martin-fowler>. Accessed 12 Feb 2017
- Lockard R (2016) Agile Uprising Agile Manifesto Review. <http://agileuprising.libsyn.com/manifesto-co-author-interview-jon-kern>. Accessed 06 Oct 2016
- Lockard R (2016) Agile Uprising Agile Manifesto Review. <http://agileuprising.libsyn.com/manifesto-co-author-interview-alistair-cockburn>. Accessed 06 Oct 2016
- Lockard R, Cleff A (2016) Agile Uprising Agile Manifesto Review. <http://agileuprising.libsyn.com/manifesto-co-author-interview-arie-van-bennekum>. Accessed 07 Nov 2016
- Lockard R, Cusack J, Lightfoot T (2017) Agile Uprising Agile Manifesto Review. <http://agileuprising.libsyn.com/manifesto-co-author-interview-ron-jeffries>. Accessed 09 Jan 2017
- Lockard R, Gifford J (2017) Agile Uprising Agile Manifesto Review. <http://agileuprising.libsyn.com/manifesto-co-author-interview-jeff-sutherland>. Accessed 03 Jan 2017
- Lockard R, Gifford J (2017) Agile Uprising Agile Manifesto Review. <https://coalition.agileuprising.com/t/podcast-released-interview-with-james-grenning/613?u=ryan>. Accessed 22 Jan 2017
- Lockard R, Gifford J (2017) Agile Uprising Agile Manifesto Review. <http://agileuprising.libsyn.com/manifesto-co-author-interview-jim-highsmith>. Accessed 30 Jan 2017
- Lockard R, Gifford J (2016) Agile Uprising Agile Manifesto Review. <http://agileuprising.libsyn.com/manifesto-co-author-interview-bob-martin>. Accessed 24 Oct 2016
- Lockard R, Gifford J (2017) Agile Uprising Agile Manifesto Review. <http://podcast.agileuprising.com/manifesto-co-author-interview-stephen-mellor/>. Accessed 26 Oct 2017
- Lockard R, Gifford J (2016) Agile Uprising Agile Manifesto Review. <http://agileuprising.libsyn.com/manifesto-co-author-interview-mike-beedle>. Accessed 07 Nov 2016
- Lockard R, Gifford J (2016) Agile Uprising Agile Manifesto Review. <http://agileuprising.libsyn.com/podcast/manifesto-co-author-interview-andy-hunt>. Accessed 22 Nov 2016
- Lockard R, Gifford J (2016) Agile Uprising Agile Manifesto Review. <http://agileuprising.libsyn.com/podcast/manifesto-co-author-interview-brian-marick>. Accessed 28 Nov 2016
- Margolis N (1988) Du pont case unit hits the road: Chemical giant melds technology, psychology to create customer software. *Computerworld* 7:105
- Mellor S, Balcer MJ (2002) Executable UML: A Foundation for Model-driven Architecture. Addison-Wesley, Boston and San Francisco and New York

- Pathak K, Saha A (2013) Review of agile software development methodologies. *Int J* 3:270–276
- Porta M, Greenland S, Hernán M, Silva IDS, Last JM (2014) *A Dictionary of Epidemiology*. Oxford University Press, Oxford
- Royce WW (1970) Managing the development of large software systems. *Proc. Westcon*, IEEE CS Press 26:328–339
- Runeson P, Höst M (2009) Guidelines for conducting and reporting case study research in software engineering. *Empir Softw Eng* 14:131–164
- Schwaber K (2004) *Agile Project Management with Scrum*. Microsoft Press, Redmond, Wash
- Scrum.org (2017) Ken Schwaber on the History of Scrum. <https://www.youtube.com/playlist?list=PLgDaZD8y4z0C-HxB71eRBRz-qTthrJhUi>. Accessed 08 Nov 2017
- Shlaer S, Mellor S (1988) *Object-oriented Systems Analysis: Modeling the World in Data*. Yourdon Pr, Englewood Cliffs, N.J.
- Stol K-J, Ralph P, Fitzgerald B (2016) Grounded theory in software engineering research. In: *Proceedings of the 38th International Conference on Software Engineering*. pp 120–131. <https://doi.org/10.1145/2884781.2884833>
- Swartout W, Balzer R (1982) On the inevitable intertwining of specification and implementation. *Commun ACM* 25:438–440
- Theocharis G, Kuhrmann M, Münch J, Diebold P (2015) Is water-scrum-fall reality? on the use of agile and traditional development practices. In: Abrahamsson P, Corral L, Oivo M, Russo B (eds). *Product-Focused Software Process Improvement*. Lecture notes in computer science, edn. Springer International Publishing, Cham Vol. 9459, pp 149–166
- Thomas D (2017) Agile is Dead. <https://www.youtube.com/watch?v=a-BOSpxYJ9M>. Accessed 10 Feb 2017
- VersionOne Inc. (2017) *State of Agile Survey: 11th annual Report*. <https://explore.versionone.com/state-of-agile/versionone-11th-annual-state-of-agile-report-2>. Accessed 05 Nov 2017
- Wallis P (1984) *Software engineering with ada*. Grady Booch, Benjamin/Cummings 1983:502
- Weber S (2015) *Agile in Automotive – State of Practice 2015*. www.kuglermaag.com/agile2015. Accessed 01 Dec 2015
- Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A (2012) *Experimentation in Software Engineering*. Springer Science and Business Media, New York
- Yin RK (2009) *Case Study Research: Design and Methods*. Sage, Los Angeles

Submit your manuscript to a SpringerOpen[®] journal and benefit from:

- Convenient online submission
- Rigorous peer review
- Open access: articles freely available online
- High visibility within the field
- Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com
