Wagenaar et al. Journal of Software Engineering Research and Development (2018) 6:7
https://doi.org/10.1186/s40411-018-0051-7

Journal of Software Engineering Research and Development

RESEARCH Open Access

Working software over comprehensive documentation – Rationales of agile teams for artefacts usage

Gerard Wagenaar[1*], Sietse Overbeek[2], Garm Lucassen[2], Sjaak Brinkkemper[2] and Kurt Schneider[3]

* Correspondence: g.wagenaar@avans.nl
[1]Avans University of Applied Sciences, Breda, the Netherlands
Full list of author information is available at the end of the article

Abstract

Agile software development (ASD) promotes working software over comprehensive documentation. Still, recent research has shown agile teams to use quite a number of artefacts. Whereas some artefacts may be adopted because they are inherently included in an ASD method, an agile team decides itself on the usage of additional artefacts. However, explicit rationales for using them remain unclear. We start off to explore those rationales, and state our primary research question as: What are rationales for agile teams to use artefacts? Our research method was a multiple case study. In 19 agile teams we identified 55 artefacts and concluded that they in general confirm existing research results. We introduce five rationales underlying the usage of artefacts in ASD: (1) Adoption of ASD leads to agile artefacts, (2) team-internal communication leads to functional and technical design artefacts, (3) quality assurance leads to test-related artefacts, (4) agile teams impose governance on their own activities, and (5) external influences impose user-related material. With our contribution we substantiate the theoretical basis of the Agile Manifesto in general and contribute to the current research with regard to the usage of artefacts in ASD in particular. Agile teams themselves may from this research extract guidelines to use more or less comprehensive documentation.

Keywords: Agile manifesto, Agile software development, Agile teams, Artefacts, Rationales for documentation

1 Background

Fifteen years have passed since the Agile Manifesto (Beck et al. 2001) was published. After half of this period, some 7½ years after its inception, the research community had lavished attention on issues related to agile software development (Dybå & Dingsøyr, 2008). Yet at the same time "*exciting research areas that can further our understanding of the effectiveness of agile methods and practices*" (*p.1219*) were still identified, among which not the least one: the 'core' of agile. Research thus continued, but five years later a research gap with regard to the implications of agile information system development on the coordination, collaboration and communication mechanisms within agile teams was still noted (Hummel, 2014).

The Agile manifesto itself values "*working software over comprehensive documentation*" and emphasizes "*The most efficient and effective method of conveying information to and within a development team is face-to-face conversation*". However, Stettina and Heijstek (2011) concluded: "*Agile practitioners do not seem to agree with this agile principle*" (*p. 159*). Instead

developers find documentation important but at the same time observe that too little of it is available in their projects.

Despite the fact that documentation usage in agile software development (ASD) has a somewhat 'old-fashioned' connotation, recent research has shown that traditional software development approaches and ASD are being blended in a hybrid approach; among the different combinations, Scrum, the classic Waterfall model, and V-shaped processes account for the majority (Kuhrmann et al. 2016; Theocharis et al. 2015). This result is further supported by a survey outcome, where approximately 75% of the participants answered that they (intentionally) combine different development approaches (Kuhrmann et al. 2017). Models have emerged to describe combinations of traditional and agile methodologies. Bustamante and Rincón (2017) developed the WYDIWYN (What You Define, Is What You Need) model to define agile and traditional methodologies oriented to what a company needs, simplifying the identification, correlation and selection of phases, roles, tasks and work products among different frameworks; Gill et al. (2016) describe a reference model for hybrid traditional-agile software development methodologies.

More in particular not only traditional and agile software development processes are blended, but also elements from them, especially artefacts. We define an artefact, in line with previous research, as a tangible deliverable produced during software development, including materials in both physical and electronic format (Wagenaar et al. 2015). This definition certainly includes physical artefacts, such as story cards and the Wall (Sharp et al. 2009), but allows in fact for a much broader spectre of artefacts. We join (Sharp et al. 2009) in their description of the connection between documentation and artefacts in ASD as "*Documentation is kept to a minimum, in favour of close collaboration, and simple artefacts*". This definition equates (minimum) documentation and simple artefacts, opposing the both to the comprehensive documentation of the Agile Manifesto. Following this observation we prefer using the term 'artefacts' in the remainder of this article; this also allows us to overcome the perceived ravine between agile and old-fashioned documentation.

Recent studies on artefacts usage in ASD shows that agile developers use quite a number of artefacts (Bass 2016; Gröber 2013; Liskin 2015; Wagenaar et al. 2015; Wagenaar et al. 2017). Some of them are inherent to an ASD method, for instance, user story or backlog, but others are not, recalling bygone memories from a Waterfall era, that advocated program design first and documenting it thoroughly, using, for instance, a design and a test document (Royce 1970). This by no means implies that the latter, for instance test documents, should not be part of a sound software development process, in fact they should be, but merely that they are not explicitly included as element of an ASD method, for instance, Scrum.

While there is value in comprehensive documentation the authors of the Agile Manifesto value working software more. Documentation functions as a mean of communication in software development in general (Curtis et al. 1988), but its importance is recognized in ASD as well and the agile approach to artefacts may be compared with 'travelling light': "*Create just enough models and documentation to get by*" (Ambler 2002, p.29). Turk et al. (2005) also discussed this tension when they formulated a documentation proposition, "*The de-emphasis of documentation as a communication aid is based on an assumption that tacit knowledge is to be valued over externalized knowledge*" (*p.11*) and mention that this proposition has both proponents and critics. They also observe that "*the agile process community claims that more is gained through*

informal personal communications than through communication based on formal documentation" (*p.10*).

These observations again illustrate the turbid relation between ASD and artefacts. One way to shine light on this relationship is to explicitly investigate in agile teams what artefacts they use and, even more important, why they use them. The need for research in this area was recently confirmed again: "*In-depth insight into agile software engineers' needs for information that can be covered by documentation is still lacking*" (Voigt et al. 2016, p. 4.2). With our research we add another brick in the wall in the research on communication in ASD, more in particular in the extent to which formal communication in the form of artefacts is used.

We formulate our research question as:

What are rationales for agile teams to use artefacts?

With this research question we investigated the rationales behind decisions an agile team take with regard to its usage of artefacts. The answer to this question should reveal how members of an agile team, with the Agile Manifesto in mind, apply one of its statements. Since ASD methods are commonly used in delivering state-of-the-art software (Bustard et al. 2013; de O Melo et al. 2013; Rodríguez et al. 2012; VersionOne 2017), it becomes even more important to annotate or elaborate the agile manifesto with regard to its use of artefacts.

To answer our research question we performed case studies within 19 organizations to ask representatives of agile teams our above-mentioned question. All organizations were software producing organizations (SPOs), involved in software product management (Fricker 2012). The organizations varied (1) in size, from rather small meaning less than 50 to large with one as large as several thousands and others as several hundreds of employees, (2) in business domains, with finance, customer relationship management, health care, and others, (3) in team size, where in general the 7 ± 2 advice (Schwaber and Beedle 2002) was followed, but also 10+ teams, and (4) in agile experience, from (over) ten years to recent, with a year order of magnitude.

Our conclusions from the case study articulate that agile teams' rationale for the usage of artefacts has five major elements: (1) artefacts may simply come with ASD as 'prescribed', (2) they provide useful governance for the team, (3) they are useful and/or necessary for internal communication, which is then not face-to-face at all, (4) they are useful and/or necessary for quality reasons, or (5) external parties need it.

With our contribution we substantiate the theoretical basis of the Agile Manifesto. We also add another piece in the puzzle for the hybrid agile / waterfall research arena, when we show that agile teams in practice use all kinds of artefacts, inherent to an ASD method or not. Finally, agile teams themselves may extract guidelines to enlarge or diminish their usage of artefacts.

The remainder of this paper is structured as follows. In Section 2 we review related work where we focus on research with regard to the usage of artefacts in ASD in general and its implications for the derivation of a rationale for their usage in particular. In Section 3 we present our research method, a multiple case study, including data collection and analysis. Section 4 presents our findings. Finally, section 5 presents our conclusions, discusses validity of our research and indicates directions for further research.

2 Related work on ASD artefacts

In the search for hybrid agile/waterfall models the usage of artefacts has received little attention. Adversely, usage of artefacts in ASD alone has recently attracted research attention.

Gröber (2013) constructed an agile artefact class diagram with 19 artefacts as result of a systematic literature study on artefacts usage in ASD. The diagram was subsequently extended to a more generic model to abstract it from local, project-specific processes as well as refined on the basis of experiences from practice (Femmer and Kuhrmann 2014; Kuhrmann et al. 2013). Categories in the diagram are: (1) Project management, (2) Requirements specification, (3) Production process, and (4) Final artefacts (Gröber 2013). Final artefacts are equated to working software and include source code and Commercial-Of-The-Shelf (COTS) software. Requirement specification artefacts need to be implemented in order to build final artefacts: Feature, requirement, user story and use case. To transform requirements to a final product, production process artefacts are suggested: Backlogs and their backlog items, including the wall artefact which is used to enact the process as it can be seen as a simplified physical representation of a backlog containing items in the form of index cards. Also parts of the test cases are considered as they drive the production of code. Finally project management artefacts are listed: A release plan, a burn-down chart, and a coding standard.

Classification criteria were used to cluster artefacts in groups. In building the classification a tag cloud was generated on the basis of one of the inclusion criteria of the literature study, namely a 'Classification' column in the criterion "*Study is classifying a number of artifacts with an arbitrary characteristic. (Possible characteristics could be e. g. textual artifacts, test artifacts)*" (Gröber 2013, p. 9). The cloud revealed classification criteria in which artefacts can be grouped with the most used criteria being: code, test, design, requirements, physical, documentation, management, production, intermediate, final, internal and digital. These were the basis for the final classification.

This research lists ASD artefacts on the basis of a classification and thus provides an overview of which artefacts are used in ASD. Although some hints may be derived on the basis of the classification, neither its derivation procedure nor its contents addresses the rationale for usage of the artefacts.

Several studies investigated the usage of agile artefacts in specific domains, such as requirements communication or user centred design. Liskin (2015) lists three artefact categories: Container, Individual element, and Solution model. Containers are characterized by their value to hold everything together in one place, for instance a virtual project environment. Individual elements are either user-oriented or technical. Examples are a use case (user-oriented) and a system requirement (technical element). Finally a solution model illustrates aspects of the future solution in a concrete model, such as a GUI mock-up or an abstract model, such as a data model. Garcia et al. (2017) used a systematic mapping study to identify which artefacts are used in order to facilitate the communication in an agile user-centred design approach, but did not classify the resulting artefacts. Artefacts in both studies bear similarities to those in previously mentioned models, but are restricted to artefacts with regard to requirements and user-centred design respectively.

In a study on large-scale offshore software development programmes Bass (2016) identified 25 artefacts on five levels of abstraction: Programme governance, Product, Release, Sprint, and Feature. Program governance artefacts are created to coordinate cooperating agile development teams and mitigate risk of development programme

failure by providing a layer of oversight and governance; they include risk assessment and several architectural artefacts. The other four levels consist of ever finer-grained artefacts used within a product itself, its releases, its constituting sprints and the features dealt with within a sprint. The research focused on artefacts across the development life cycle but only investigated large-scale agile development programmes. It is argued that the artefacts used in large-scale agile development programmes are a superset of those used in smaller projects.

This study was designed to focus on artefacts to subsequently shed light on the tailoring of agile methods in a large-scale software development programme context. The study's interest was in practitioner interactions with artefacts and not in the artefacts per se, where its classification mechanism was mainly derived from previous research (Femmer and Kuhrmann 2014; Kuhrmann et al. 2013).

Teams using agile development shall, among others, identify documents to be produced by the process or project (ISO 2011). Although ISO-standard 26,515 primarily focusses on user documentation processes, it at the same time identifies documentation items that should be produced by agile projects. Such life cycle documentation should be produced in projects using agile development to communicate processes, requirements, and deliverables required of the teams working on the project. These documents may contain less detail than their counterparts in other software development methods. Examples include user stories, burn down charts, and test plans.

In summary, previous research enumerated artefacts in several models or lists by using different classifications. The classifications contain elements which could relate to rationales for using them, although none of the models was built with this purpose explicitly in mind. Implicit rationales for the usage of artefacts could be derived, but the question 'Why are you using this artefact?' was never answered by the ultimate source: the agile team itself.

3 Method

In this section we describe our research method, starting with our study protocol, and the subsequent data collection, as well as the analysis procedure. To investigate our research question we used a holistic multiple-case study with as unit of analysis the agile team and its artefacts; this approach is an accustomed way to investigate phenomena in a context where events cannot be controlled and where the focus is on contemporary events (Yin 2013). Since research results are scarce with regard to our research question, our aim is theory building rather than theory testing; the former is useful in early stages of research on a topic or when a fresh perspective is needed, while the latter is useful in later stages of knowledge (Eisenhardt 1989).

3.1 Case study protocol

Although a case study allows flexible research, this does not mean that planning is unnecessary (Wohlin et al. 2012). A case study protocol (CSP) is the container for the design decisions on the case study as well as for field procedures for carrying through the study. Among a CSP's purposes are guiding data collection and reporting. As an element in guiding data collection, interview questions were drawn up and a reporting structure was also included in our CSP. A reference to the CSP is provided in the 'Declarations' section at the end of this

paper, but elements from the CSP will also be used to accompany our description of the data collection process (section 3.2).

For our research we formed a research group in which two senior researchers, being two of the authors of this manuscript, coordinated and supervised the activities of the group. Other members of the group were graduate students in preparation for their Master's thesis. If "*no reason exists to preclude most elementary, secondary and university students from becoming critical researchers*" (Kincheloe and Steinberg 1998, p. 2), then graduate students certainly qualify as members of a research group, provided that they are properly equipped. In our case they were supported by a CSP, i.e., interview guidelines and reporting structure.

3.2 Data collection on artefact usage in agile teams

Data collection took place through the use of various single-site case studies following widely accepted guidelines for case studies (Yin 2013). Due to the exploratory nature of the research, semi-structured interviews were used to allow interviewees to speak freely and to be able to ask follow-up questions (Kajornboon 2005). Data collection took place by members of the research group, under supervision of the senior researchers.

Approximately 80 SPOs were approached by our group, of which 19 organizations with 19 software development teams were willing to participate in our research (Table 1). In case an organization had a portfolio of several products, the research focused on one agile team working on one of the products only. In order to qualify for our research, organizations were further required to have their own in-house software development unit, although part of its activities could be outsourced. Within the unit, at least one team had to apply an ASD method.

Organizations/teams are letter coded with 'Id' for reasons of confidentiality. The size of an organization is considered to be small if it has less than 50 employees, medium between 50 and 250 employees, and large more than 250. For an agile team small indicates less than 7 members, medium between 8 and 11, and large 12 or more.

The organizations and their corresponding agile teams thus vary in business domains, in organization and team size, and in country, although the Netherlands predominates. This is not a surprise, since the vast majority of the organizations we approached was based in the Netherlands in the first place. Our interviewees have in general 'leading' positions, such as product owner, Scrum master, or lead developer. This is mainly caused by our design, in which we favoured interviewing at least one representative with a broad view on the team and its activities.

The data collection procedure according to the CSP was the same for all 19 teams. In each organization interviews were held. Most of the time this was one interview, often with a product manager or owner. In some cases two interviewees were involved and/or two interviews were conducted. Interviews lasted, on average, one hour and were transcribed and/or summarized thereafter.

To allow for the results to be comparable over the single cases we established a common vocabulary in the description of artefacts by using a FLOW model (Stapel et al. 2009; Stapel and Schneider 2012, 2014). It is a modelling technique containing several elements:

- Documented information is called solid information if it is (1) long term accessible, (2) repeatedly readable, and (3) comprehensive for third parties. In contrast,

Table 1 Characteristics of teams/organizations

Id	Domain	Size		Country	Interviewee(s)
		Organization	Team		
A	Software	Large	Medium	NL	Product owner
B	CRM	Small	Small	NL	Scrum master (lead develop architect)
C	CRM	Small	Small / Outsourced	NL / TR	Product owner
D	Industry	Large	Large	NL	Scrum master
E	Software	Small	Large	NL	Product owner (CTO)
F	HRM	Medium	Small	NL	Product owner (lead developer) Product owner
G	E-learning	Small	Small	UK	Product owner
H	Hospitality	Small	Small / Outsourced	NL / MK	Scrum master (project manager)
I	Finance	Large	Medium	NL	Scrum master (tester)
J	Health care	Medium	Small	NL	Lead software developer Senior functional developer
K	ERP	Medium	Small	NL	Developer Product owner Chief executive officer
L	Purchase	Medium	Small / Outsourced	NL / BG	Product owner Chief technical officer
M	Insurance	Large	Small	NL	Development director
N	HRM	Small	Medium	NL	Scrum master (lead developer)
O	Finance	Large	Small	CN	Product owner
P	Finance	Medium	Medium	NL	Scrum master (agile coach)
Q	Health care	Large	Medium	NL	Scrum master (business line manager)
R	Finance	Large	Medium	NL	Scrum master
S	Finance	Small	Small / Outsourced	NL / SP	Senior technical lead developer

undocumented or fluid information is information that violates any one of the above criteria.

- Transfer of documented/undocumented information originates from solid and fluid flows respectively.
- A role describes an actor being the producer or consumer of any form of information. A role can also be an individual person.

The FLOW notation was designed to explicitly visualize the concepts of fluid and solid information and flows (Schneider et al. 2008). The notation is intended for a variety of users and thus reuses well-known concepts from existing notations. Although differences between FLOW and related notations may seem subtle at first, fluid flows are obviously more difficult to represent in other notations which were not tailor-made for them. The emphasis on information and its distinction between solid and fluid information makes a FLOW model suitable for the representation of artefacts.

As an example the model for organization R is provided (Fig. 1). The model shows documents and people involved. Direct flow of information (fluid) is represented by faces and dashed arrows originating from the people they represent, for instance 'Customers' towards 'Requirements gathering'. Solid arrows originate from documents (solid information, as in, for instance, 'Maintenance backlog'). Rectangles stand for activities and they are treated as

black boxes. In the diagram persons participating in an activity are shown. The internal flow of information within an activity, however, is hidden or unknown and not shown here.

Members of the research group transcribed and/or summarized the interviews before visualizing them in a FLOW model. To this extent the members collectively acquainted themselves with relevant FLOW literature and drawing up FLOW models, although for fictitious cases at first. Reporting instructions concerning the FLOW models were also part of the CSP (see 'Declarations' section). Besides a FLOW model itself, instructions in the CSP also required a description of solid and fluid information and the rationale for usage of the information. In first instance members of the research group combined results in a draft report. The senior researchers then provided interim feedback, which would sometimes lead to additional data collection. At approximately the same time the draft report was put at the disposal of interviewees to allow for comments with regard to correctness of transforming the interviews to FLOW models and the accompanying descriptions. Both sources of feedback were introduced to assure quality of the final report. An average (final) report comprised 24 pages.

3.3 Data analysis

The 19 FLOW models together with their description of solid and fluid information and the rationale for the usage of the information formed our base data. We started data analysis by extracting solid information as artefacts from the models, and this resulted in 360 artefacts. The list was reduced in a two-step process: (1) a lexical analysis and (2) a semantic analysis (Jurafsky and Martin 2008).

In the lexical analysis we removed distinctions in singular and plural forms, for instance 'User story' (listed 3 times) and 'User stories' (11 appearances). We removed adjectives, for instance mapped both '(Conceptual) user story' and 'User story' on 'User story'. This reduced the number of 360 to 200 artefacts. In our lexical analysis we primarily used the names of the artefacts and only occasionally and superficially used their descriptions.

In a further semantic analysis we did use the description of artefacts from the FLOW model to identify similarities and differences in artefacts. In this analysis we applied

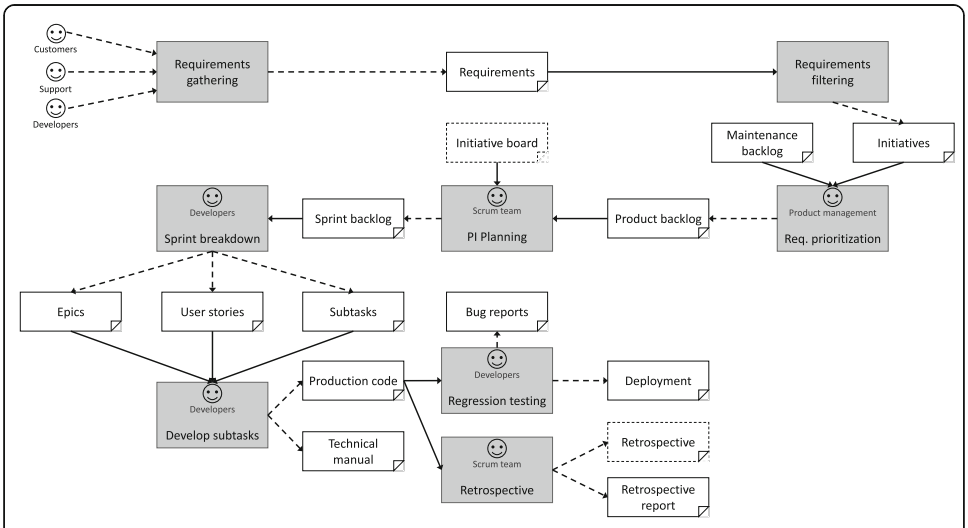

Fig. 1 Example of a FLOW model (team R)

constant comparison (Glaser and Strauss 1967). This comparison was done manually, proceeding from one artefact to another and working backwards whenever appropriate. For instance, we already identified an artefact 'Bug report' with description *'A bug report is a document that is used to report a bug, an unwanted and/or unintended malfunction in the source code*'. When we found 'Maintenance backlog' described as *'A maintenance backlog records data about the software during runtime, used to identify issues and points of improvement*', we mapped this backlog to the artefact 'Bug report'. As another example we mapped 'Specification' ("*…an extensive description of the requirement*") to 'Functional design' ("*… describes techniques or prescribed methods that need to be implemented in the user stories, as well as properties of the required input and output*"). In the process we deleted artefacts when they could be equated to another artefact mentioned by the same team or we mapped artefacts when they could be equated to artefacts also mentioned by another team. In both cases we chose one of the applicable descriptions of the artefact or combined them to form a new blended description. This process also applies to the rationales provided for the artefacts.

4 Results

4.1 Artefacts

Lexical and semantic analysis resulted in first instance in a list of 55 different artefacts (Table 2). An 'x' stands for usage of the artefact (row) by the agile team (column).

The artefacts that are mentioned most (top 5) are: Sprint backlog (18), Source code (17), Product backlog (16), User story (15), and Release note (12). This shows a predominance of artefacts directly related to an ASD method.

4.2 Influence of agile team on artefact usage

All artefacts from Table 2 are used by at least one agile team in our case study. However, not all of them are used as a choice of an agile team itself. To distinguish between usage in general on the one hand and usage as result of an agile team's decision on the other hand, we use IEEE standard 1074–2006, which provides a process for creating a software project life cycle process (IEEE 2006). In particular, the standard distinguishes the following phases in the software development-oriented processes:

- Pre-development process.
- Development process, to be further divided into requirements process, design process, and implementation process.
- Post-development process.

We present our list of artefacts according to this distinction (Table 3). As a consequence of this categorization we exclude pre-development artefacts from our consideration. This is in line with our research question which is related to the rationales of an agile team for artefacts usage. Artefacts that are not under an agile team's influence are therefore not contributing to an answer to our research question. The exclusion is neither a negation of the existence of such artefacts nor of their influence on some aspects of ASD by an agile team, but pre-development artefacts usage is decided upon outside its realm. Nevertheless, particularly a product owner (Schwaber and Sutherland 2016)

Table 2 Overview of artefacts

Artefact	Agile team ID																		# occurrences
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	
Acceptance criteria														x		x				2
Architecture standard	x						x						x							3
Bug report	x	x	x	x			x	x		x	x	x	x		x			x		12
Burndown chart	x	x	x	x		x						x			x		x			8
Business case					x									x			x			3
Business plan			x			x				x			x							4
Coding standard																			x	1
Contact																			x	1
Contract	x												x				x			3
Definition of done								x									x			2
Definition of ready								x												1
Deployment script												x								1
Epic	x				x	x						x					x	x		6
Functional design	x		x			x		x		x								x	x	7
Legislation/Regulation										x			x							2
Impact analysis			x					x												2
Implementation guide											x									1
ISO standards												x								1
Market analysis										x					x					2
Market requirement					x	x		x	x	x	x	x	x	x	x	x				11
Mock-up							x					x					x			3
Product backlog	x	x	x			x	x	x	x		x	x	x	x	x	x	x	x	x	16
Product portfolio													x							1
Product requirement						x			x	x		x	x	x	x	x				8
Prototype																	x			1
Quotation					x															1
Release				x		x	x	x			x			x				x		7
Release checklist	x																x			2
Release log																			x	1
Release note	x	x		x		x	x			x	x	x		x		x	x		x	12
Release plan	x			x				x					x							4
Request for information						x														1
Requirement	x	x	x	x	x	x	x	x	x	x		x	x	x		x	x	x		2
Retrospective report									x									x		2
Risk assessment			x			x							x			x	x			5
Roadmap	x		x				x	x	x	x	x		x				x		x	10
Source code	x	x	x	x	x	x	x	x	x		x		x	x	x	x	x	x	x	17
Sprint backlog	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		18
Status board		x	x			x	x	x											x	6
Task								x	x									x		3
Technical design			x											x						2
Technical documentation	x	x				x						x						x		5
Technical requirement	x							x				x								3

Table 2 Overview of artefacts *(Continued)*

Artefact	Agile team ID																		# occurrences
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	
Technical roadmap										x										1
Template functional description																			x	1
Test case			x								x	x			x					4
Test criteria		x																		1
Test plan	x					x								x		x				4
Test report		x		x	x		x	x						x			x			7
Unit test report		x				x											x			3
Use case						x	x										x			3
User acceptance test	x	x				x											x			4
User manual										x	x				x				x	4
User story	x	x	x	x	x	x		x	x	x	x	x	x			x	x	x		15
Vision/Strategy			x					x					x			x				4

or product manager (Ebert and Brinkkemper 2014) may, as an agile team member, be partially involved in construction of pre-development artefacts. Yet these artefacts cannot be attributed to an effort of an agile team as they do in fact hardly directly contribute to the agile team's activities.

Our limitation is also reflected in rationales which are given for usage of the pre-development artefacts. As a first example we present some rationales for the usage of pre-development artefacts:

— "*Organization L needs to comply to ISO standards in order to have Service Level Agreements (SLAs) with customers and suppliers.*"
— "*The market analysis [...] is required by internal stakeholders to have an overview of the product.*" (organization O)
— "*Without the contract, it is not possible to develop a software product for the customer at all.*" (organization Q)

Throughout this and the next section citations may have been slightly edited, as compared to the original statements, to improve readability. Source material is available (see 'Declarations' section).

As a second example we present some rationales for the use of 'Roadmap':

— "*Product roadmaps stimulates thinking over the immediate and future evolutions of the product.*" (organization G)
— "*The roadmap provides the team with a clear direction in which they should develop the product, in accordance with management.*" (organization I)
— "*The roadmap provides important information to the stakeholders about future plans.*" (organization S)

Especially the demarcation line between pre-development and development is a delicate one, because some artefacts, especially requirement artefacts, seem to fit on

Table 3 Artefacts in software development-oriented processes

Pre-development	Development	Post-development
Business case	Acceptance criteria	Deployment script
Business plan	Architecture standard	Implementation guide
Contact	Bug report	Release
Contract	Burndown chart	Release checklist
Legislation/Regulation	Coding standard	Release log
ISO standards	Definition of done	Release note
Market analysis	Definition of ready	User manual
Market requirement	Epic	
Product portfolio	Functional design	
Product requirement	Impact analysis	
Quotation	Mock-up	
Release plan	Product backlog	
Requirement	Retrospective report	
Request for information	Source code	
Risk assessment	Sprint backlog	
Roadmap	Status board	
Vision/Strategy	Task	
	Technical design	
	Technical documentation	
	Technical requirement	
	Technical roadmap	
	Template functional description	
	Test case	
	Test criteria	
	Test plan	
	Test report	
	Unit test report	
	Use case	
	User acceptance test	
	User story	

either side of the line. Development includes a requirements process. Yet we classified some forms of requirements as belonging to the pre-development process, which are market requirement, product requirement and requirement in general. The organizations in our research were all SPOs and in this context market requirements are requirements as they are formulated by external stakeholders. Product requirements are market requirements which are adopted for inclusion in a future release of the software product. The term 'requirement' in general is in some cases used for market requirement, sometimes for product requirement or for both. We decided to categorize them as part of pre-development, since the involvement of an agile team in their formulation is limited, apart from some contribution by, mostly, a product owner. Also, requirements do still appear in (the requirement process within) development in the shape of epics and user stories.

The involvement of an agile team focusses on the development activities, with participation in post-development activities, where the latter depends heavily on the level of implementation of DevOps, Development-Operations (Dyck et al. 2015).

4.3 Rationales for artefacts usage

Having identified development and post-development artefacts as artefacts for which an agile team is involved in their usage, we now turn an exploration of their rationales. We will introduce five groups of rationales: Agility, Governance, Quality Assurance, Internal communication and External follow-up. Our choice of just those categories is inspired by the work of Grant (1996) and Strode and Huff (2014). Grant points to four mechanisms for integrating specialized knowledge:

- Rules and directives are (impersonal) approaches to coordination where rules may be viewed as standards which regulate the interactions between individuals.
- Sequencing is a means by which individuals can integrate their specialist knowledge by organizing activities in a time-patterned sequence.
- Routines may be simple sequences, their interesting feature is their ability to support complex patterns of interactions between individuals in the absence of rules, directives, or even significant verbal communication.
- Group problem solving and decision making supplements all the above mechanisms by recognizing that some tasks may require more personal and communication-intensive forms of integration.

Strode and Huff (2014) distinguished categories of artefacts: Synchronization and boundary spanning. Synchronisation artefacts are produced during synchronisation activities that contain information used by all team members in accomplishing their work. These artefacts include, among others, working software, a product backlog, and stories. Boundary spanning artefacts are physical things produced to support boundary-spanning activities and enable coordination beyond the team and project boundaries. These could be a project management plan for a project management office or a Request for Change form for an IT support unit when additional servers were required. Some similarities with the product/process dimension of the previous research may be discovered. Although this research explicitly concerns reasons for artefacts usage, i.e. for synchronisation or boundary spanning, it at the same time lacks artefacts involved, other than in a context of examples. This research builds on a long tradition of the role of artefacts in communication, for instance in Computer Supported Cooperative Work, CSCW (Dix 1994) or ASD (Sharp and Robinson 2010).

We will now describe the rationales in more detail. In addition to the description we also indicate a group's relation to the base material and/or to the literature mentioned above.

4.3.1 Agility

When inspecting rationales for the development artefacts a first group arises for which we can identify an 'agility rationale'. We use this term to describe rationales which explain the usage of artefacts as a result of the adoption of an ASD in the first place. Sometimes the rationale for such an artefact was worded in a straightforward manner:

— "*User stories are needed to describe features when using SCRUM.*" (team E)

Others were more covert, but upon inspection they correspond well with other sources. For instance, an 'official' description of a product backlog taken from Schwaber & Sutherland (2013, p. 12–13) is: "*The Product Backlog is an ordered list of everything that might be needed in the product and is the single source of requirements for any changes to be made to the product*". This coincides well with rationales which are given for its use:

— "*The backlog is used to store the product requirements.*" (team A)
— "*The product backlog serves as a way of keeping track what has been changed and implemented during a sprint; it is important, because it records everything that was implemented, including reasons and dates and is thus available for reference.*" (team I)
— "*The product backlog keeps track of all the user stories associated with the product.*" (team B)

Several artefacts share this mutual rationale. Burndown chart, Definition of done, Definition of ready, Epic, Product backlog, Retrospective (report), Source code (Increment), Sprint backlog, Status board, Task, and User story: They all come with the adoption of ASD. This is also the reason for referring to them as agile artefacts, as opposed to non-agile artefacts which are then in fact all other artefacts from Table 3, pre-development artefacts excluded.

The rationale for these artefacts arose in a natural way from our base material. In terms of Grant the rationale reflects a kind of rule/directive, where Strode would consider them as artefacts used for synchronisation.

4.3.2 Governance

Some artefacts are motivated by governance an agile team adopts itself. Of course some standards are imposed externally or organizationally, where the latter refers to standards within the organization, but outside a team's influence. Examples of this are legislation/regulation or ISO standards, but they were considered to be pre-development artefacts and thus outside the influence of an agile team. Governance here applies only to rationales for which the team itself decides on their use. They include:

— "*Architecture standards describe requirements for the platform for which a feature is being developed, as well as provide guidelines for code standards and style. This is important since all code within the project needs to be consistent and the architecture of the projects needs to be standardized to facilitate future development.*" (team A)
— "*A definition of ready describes what a user story must adhere to, before development can start. It contains a checklist for a requirement and relates to what makes a good user story.*" (team H)
— "*A coding standard is used to write good code and to check if code is written properly.*" (team S)
— "*The template (functional description) helps the product manager in writing a functional description.*" (team S)

The governance rationale clusters acceptance criteria, architecture standard, coding standard, definitions of done & ready, and template functional description. From this enumeration it shows that some rationales may apply to multiple artefacts. For example, the definition of done was also motivated by the rationale that it comes with the adoption of an ASD.

This rationale was inspired by the notion of Grant's routines, although in its elaboration it is in fact a mixture between rules and routines. There are no predefined rules or directives, but some teams more or less decided to still formalize their routines in a rule-based way. With this viewpoint the mentioned artefacts were unified under the governance rationale. For Strode they would again be synchronization artefacts.

4.3.3 Quality assurance

Testing is for a long time considered to be an important quality aspect of software quality assurance: "*The responsibilities of the quality assurance activity generally include: … Test Surveillance - Reporting of software problems, analysis of error causes and assurance of corrective action*" (Boehm et al. 1976, p. 601). This is still recognized by agile teams today as they motivate their use of test-related artefacts:

- "*These (unit) tests are performed to ensure the quality of the programmed code and eventually the products.*" (team B).
- "*The test plan ensures that all delivered software is of high quality and captures the intended functional behaviour.*" (team F)
- "*Test results ensure the correct- and completeness of a new piece of code. If a new piece is not correct and/or complete the result indicates what is wrong and the problem is easy to find for the developer.*" (team N)
- "*For documentation and proof purpose, bug reports are created to give an insight on how maintenance on the software is performed.*" (team L)

Bug report, test case, test criteria, test plan, test report, unit test report, and user acceptance test all share a rationale in software quality assurance.

This rationale finds its origin mainly in the base material. It does not have a direct counterpart in neither Grant nor Strode.

4.3.4 Internal communication

As for testing, design is also for a long time considered to contribute to especially the efficiency of the software development process. Some (initial) steps to transform a risky development process into one that will provide the desired product: Program design comes first (step 1) and document the design thereafter (step 2) (Royce 1970). This statement has not lost its value in agile times: "*Fundamentals of XP include starting a project with a simple design …*" (Beck 1999) or explicit Functional Model and Design (& Build) iterations in DSDM (Dynamic System Development Method) (Stapleton 1997). Of course, this applies especially to evolutionary design, as opposed to big design up front. However in both cases: "*First keep in mind what you're drawing the diagrams for. The primary value is communication*" (Fowler 2004). Agile teams value this type of communication:

— "*The design flow allows the developer to understand how a newly developed function is going to be tailored inside the product.*" (team A)
— "*For the user experience, visualizing the features (screens) of the software product is better than just a textual description.*" (team Q)
— "*The functional design improves communication and understanding of how new requirements fit into the overall software product.*" (team F)
— "*At organization H, a technical requirement describes how software should be designed from a technical standpoint; it is usually submitted by colleagues of the development team and they are gathered to improve the product specifically on technical aspects.*"

Functional design, impact analysis, mock-up, technical design, technical documentation, technical requirement, technical roadmap, and use case are several types of design that are used for communication purposes.

This rationale directly links to Grant's group problem solving as well as to Strode's boundary spanning, where in the latter case it is not crossing the boundary of the team as well expanding the range of an individual or a subgroup.

4.3.5 External follow-up

The rationale for the usage of post-development artefacts is somewhat different from the other categories. In fact their usage is not so much a decision of an agile team itself. These artefacts are especially appreciated by external parties, outside the agile team. This may be a customer, external to the organization as a whole, or another part of the organization, for instance Operations. However, the team certainly is involved with them, especially as their producer. This is reflected in the found rationales:

— "*Release notes are continuously added by the team and are a low effort approach to documenting each increment and release, because by adding small release notes to every commit, Visual Studio Team Services can generate an overview per increment or per release.*" (team K)
— "*Every product increment must be documented with a release note.*" (team L)
— "*User manual is required by external and internal stakeholders to know the new features of the product.*" (team O)

All post-development artefacts, Deployment script, Implementation guide, Release, Release checklist, Release log, Release note, and User manual are headed under this rationale.

This rationale followed from our base material, although a modest link with Strode's boundary spanning might be argued in the sense that these artefacts cross the border between the team and a part of the outside world.

5 Conclusions & Discussion

We first summarize our findings with regard to rationales for artefacts usage in agile teams (Fig. 2). Figure 2 shows the usage of artefacts (boxes) categorized by the rationale for their usage (ellipses).

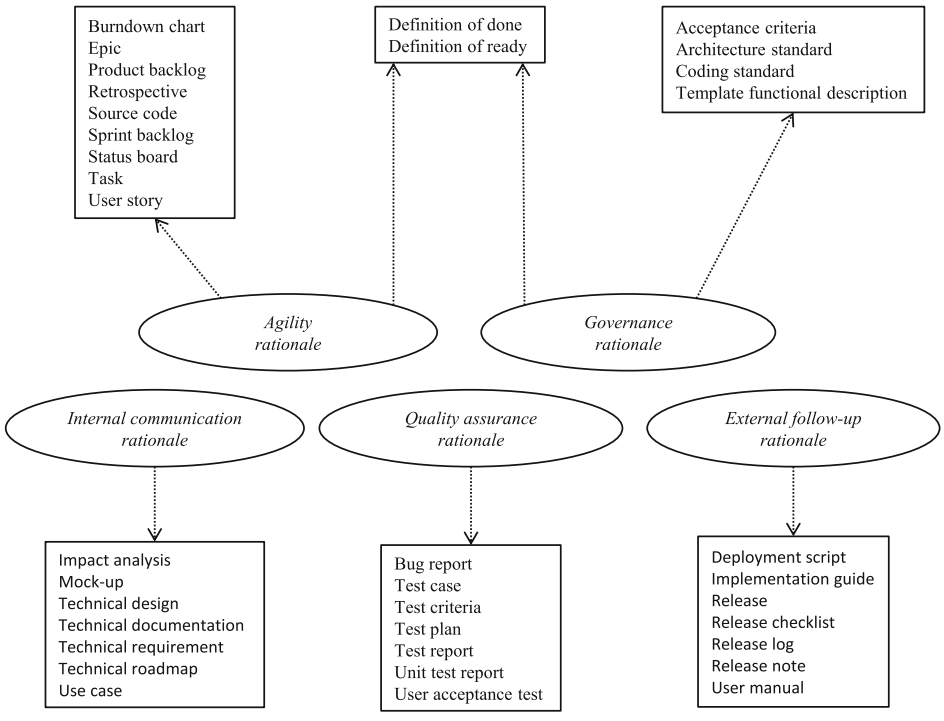

Fig. 2 Categorization of rationales for artefacts usage

We started our research from the observation that recent research reports that organizations blend waterfall and agile system development methods in all kinds of hybrid development varieties. At the same time, we noticed that recent research devotes attention to the usage artefacts in ASD. With the Agile Manifesto's valuation of working software over comprehensive documentation we phrased our research question as: 'What are rationales for agile teams to use artefacts?'

5.1 Conclusions & discussion

In our findings we consider our overview of 55 artefacts to be near to complete. When we compare the artefacts in Fig. 2 with previous research on artefact models (Bass 2016; Gröber 2013; Wagenaar et al. 2015), we encounter many similarities. Differences occur, but seem to be more a result of choices with regard to granularity rather than fundamental differences.

ISO standard 26,515 also addresses documentation items should be produced by projects using agile development to assist both the production of software and user documentation: project and sprint plan, requirements documents (user stories), test plan, risk statement, use case, descriptions of persona, burndown chart, task list, Scrum report, and lessons learned report. Comparing them to our findings only two of them are not directly recognizable from Fig. 2: project plan and description of persona. With the project plan including the sprints to be developed and any milestones such as dates to provide the software and user documentation, it bears resembles to our release plan and/or the product backlog. The description of personas, as a form of user experience design, is not explicitly visible in our findings, but may be thought to be included in a functional and/or technical design. Probably not being consciously aware of the

standard, because never mentioned explicitly, the teams nevertheless cover almost the entire list of documentation items form the ISO standard.

Proceeding from the artefacts in our case study we also retrieved rationales for agile teams' usage of artefacts and draw our conclusions on this basis.

5.1.1 Adoption of ASD leads to agile artefacts

The main rationale for using agile artefacts, for example, epics, a sprint backlog or a definition of done, is not surprising. This usage is directly motivated by the adoption an ASD method. An overwhelming majority of our teams uses artefacts for mainly this reason. Artefact usage as a result of this rationale dominates the list of all artefacts (Table 2); over one third of all artefacts (94 out of 254 occurrences) are used because of this rationale.

5.1.2 Team-internal communication leads to functional and technical design artefacts

Under the rationale of internal communication we found that many of our agile teams, on their own initiative, decided to use various additional, other than being inherent to an ASD method, artefacts with examples like functional designs, mock-ups, and technical designs. Teams introduce them to allow communication between members of a team, for instance from a user story on the product or sprint backlog (product owner) through a functional design (designer) to source code (developer). We conclude that agile teams, despite the Agile Manifesto's preference for face-to-face communication, have rationales for this usage of artefacts. This rationale is reflected in the use of boundary spanning artefacts (Strode and Huff 2014), but not for coordination beyond the project team and its boundaries as for coordination between different disciplines, for instance requirement engineering and programming, within the team.

5.1.3 Quality assurance leads to test-related artefacts

Under the rationale of quality assurance we found agile teams to use quite some test-related artefacts, including, for instance, bug reports, test plans, and (unit) test reports. In fact, only two out of our nineteen teams did not use any of the artefacts motivated by this rationale (Table 2). Furthermore, the use of test-related artefacts is in fact easily combined with ASD, as in, for instance, Test-Driven Development (TDD); "*Test-driven development is a set of techniques that any software engineer can follow, which encourages simple designs and test suites that inspire confidence*" (Beck 2003, p. xix). However, another widely used method, Scrum, does not explicitly describe test artefacts. This rationale re-emphasizes the importance of basic principles which provide keys to a successful software effort (Barry W. Boehm 1983).

5.1.4 Agile teams impose governance on their own activities

Agile teams see the benefit of applying governance to their own activities. Architecture and coding standards are for instance constituted to ensure a team's conformity to team wide agreements. This rationale would also particularly be recognized by agile teams using Scrum as ASD: "*Self-organizing teams choose how best to accomplish their work*" (Schwaber and Sutherland 2016, p. 5). Both this governance rationale as well as the internal communication rationale confirm items on the contemporary research

agenda for large-scale agile software development: Architecture and Inter-team coordination (Dingsøyr and Moe 2014).

5.1.5 External influences impose user-related material

It may be put up for dispute if user-related material really is constituted as a result of a decision of an agile team itself. However, an agile team certainly is involved in its production. And the more an agile team operates according to DevOps, the more it benefits from its own artefacts, as, for instance, a deployment script or a release checklist. And a release note, while primarily meant for customers, could also serve as a team's memory.

Usage of artefacts may or may not contradict the Agile Manifesto. Travelling light has often been taken as a motto to describe this dilemma. It could be argued that each artefact de-agilizes ASD. But whatever position is taken in this dispute, knowing rationales for the usage of artefacts contributes to clarification on the issue.

5.2 Limitations

We explicitly consider our multiple case study as exploratory. This has two main reasons. Although we collected data for 19 teams we did not collect data from every individual team member. Rationales provided by our interviewees did not necessarily reflect the opinion of an entire team. Furthermore, as a second reason, deriving groups of rationales from individual rationales is not a process in which every single step is made on the basis of a one-by-one correspondence. We first had to combine descriptions and rationales for artefacts as artefacts (source artefacts) were mapped to others (destination artefacts), but still their descriptions and rationales had to be combined in one description c.q. rationale. Secondly, we also had to integrate descriptions and rationales for artefacts which were mentioned by several teams. Although we structured and documented this process as much as possible, its chain cannot be considered to be flawless.

Our exclusion of pre-development artefacts, especially the ones which are associated with requirements in one or another format, could be argued. In defence of our choice we note that teams that mentioned the artefacts market requirement, product requirement, and/or requirement also mention at least one of the artefacts epic, user story, product backlog, or sprint backlog. In this way rationales for agile requirement artefacts are still accounted for those teams.

5.3 Case study validity

Validity of a (multiple) case study in general depends on four criteria: construct validity, internal and external validity, and reliability (Yin 2013).

5.3.1 Construct validity

Construct validity identifies operational measures for the concepts under study. To enhance construct validity (1) key informants should review draft case study reports, (2) multiple sources of evidence should be used, and (3) a chain of evidence should be established. We addressed all three: (1) For each interview summaries were drafted, including a FLOW model, and interviewee(s) were able to comment on them, (2) within some teams, but certainly not all, interviews were held with more than one interviewee

so in those cases viewpoints could be complemented, and (3) we followed a strict procedure in proceeding from an interview protocol, via a FLOW model and its artefact descriptions, to the list of artefacts. Nevertheless, teams were not visited by the same members of the research group, so interpretation may have influenced especially the construction of FLOW models. Also, when only one interviewee was involved, personal opinions may have influenced our results.

5.3.2 Internal validity

Internal validity is considered mainly a concern for explanatory case studies. We do not claim our case study to be explanatory, it is rather exploratory. Still we considered internal validity in translating oral interviews via the FLOW models to an artefacts list. Clustering rationales, as described in Section 4, was, although structured, to a great extent exploratory.

5.3.3 External validity

External validity defines the domain to which a case study's findings can be generalized. The use of replication logic is listed as the main guarantee for this. Using various single case studies on the basis of a common procedure, as in our research, in general already contributes to external validity, and thus to generalizability of results. However, our organizations being SPOs may have influenced the usage of artefacts, especially in the post-development process. Generalizability to non-SPOs is therefore limited. Even so, although our research included teams with different characteristics (domain, country, and size) we cannot claim them to be representative for every agile team. We also recognize a geographical bias with dominance of the Netherlands.

5.3.4 Reliability

Reliability should demonstrate that the study can be repeated. Using an interview guideline, instructions for FLOW modelling, and formatting results as well as the use of a case study repository contributed to reliability.

5.4 Future research

Our research was exploratory and we found directions to group rationales with respect to the use of artefacts. However, to allow for more robust conclusions, viewpoints from several team members should be taken into account. This would require in-depth interviews with members from one team, thus focusing on depth, rather than breadth. This would also strengthen the evidence for the current group of rationales.

As we now concluded face-to-face communication is certainly supplemented with artefacts. Still, another interesting question is: What is being discussed face-to-face and why? This should then go beyond 'standard' meetings as sprint planning or daily stand-up from Scrum.

Finally, our current study did not explicitly investigate characteristics nor context of teams and organizations. As in other areas, for instance the choice for a software development methodology in general (Vijayasarathy and Butler 2016), context does of course matter in the rationales for artefacts usage. Incorporation of such a context would also allow for more concrete definitions, templates or examples of artefacts.

Abbreviations
ASD: Agile Software Development; CSP: Case Study Protocol; DevOps: Development-Operations; SPO: Software Producing Organization

Acknowledgements
The authors express their gratitude to the graduate students who participated in the research group as component of their course 'Software Product Management' as well as to the organizations involved in this research; the first author also expresses his gratitude to Avans University of Applied Sciences for facilitating and supporting this research.

Availability of data and materials
Interview guidelines are available. Due to confidentiality both interview minutes and FLOW models are not publicly available. An overview of artefacts from the FLOW models as wells as their description and their rationales is available. The data analysis, especially (the results of) lexical and semantic analysis are also available. All sources are available from https://osf.io/ub4qg/.

Authors' contributions
GW and GL, guided by SB, designed the study and supervised the collection of data. GW analysed the data and drafted the manuscript. SO helped draft and review the manuscript in several versions. SB and KS reviewed the manuscript and helped fine-tune the final draft. All authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Author details
[1]Avans University of Applied Sciences, Breda, the Netherlands. [2]Utrecht University, Utrecht, the Netherlands. [3]Leibniz Universität Hannover, Hannover, Germany.

Received: 8 November 2017 Accepted: 20 June 2018
Published online: 10 July 2018

References
Ambler, S. (2002). Agile modeling: effective practices for eXtreme programming and the unified process. Wiley, New York
Bass JM (2016) Artefacts and agile method tailoring in large-scale offshore software development programmes. Inf Softw Technol 75(C):1–16. https://doi.org/10.1016/j.infsof.2016.03.001
Beck K (1999) Extreme programming explained: embrace change. In: XP Series. https://doi.org/10.1136/adc.2005.076794
Beck K (2003) Test-driven development: by example. Addison-Wesley, Boston
Beck K. Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., … Thomas, D. (2001). Agile Manifesto. Retrieved September 24, 2012, from http://agilemanifesto.org/
Boehm BW (1983) Seven basic principles of software engineering. J Syst Softw 3(1):3–24. https://doi.org/10.1016/0164-1212(83)90003-1
Boehm B. W., Brown, J. R., & Lipow, M. (1976). Quantitative evaluation of software quality. In *Proceedings of the 2nd International Conference on* Softw Eng *(ICSE), San Francisco (CA), USA, 13–15 October, 1976* (pp. 592–605). IEEE Computer Society Press
Bustamante, A. F., & Rincón, R. D. (2017). WYDIWYN – What You Define, Is What You Need: Defining Agile/Traditional Mixed Methodologies. In J. Mejia, M. Muñoz, Á. Rocha, Y. Quiñonez, & J. Calvo-Manzano (Eds.), Trends and Applications in Softw Eng - Proceedings of the 6th International Conference on Software Process Improvement (CIMPS 2017), Zacatecas, Mexico, 18–20 October 2017 (pp. 35–44). Springer, Cham. https://doi.org/10.1007/978-3-319-69341-5_4
Bustard D, Wilkie G, Greer D (2013) The diffusion of agile software development: insights from a regional survey. In: Pooley R, Coady J, Schneider C, Linger H, Barry C, Lang M (eds) Information systems development: reflections, challenges and new directions. Springer New York, New York, NY, pp 219–230. https://doi.org/10.1007/978-1-4614-4951-5_18
Curtis B, Krasner H, Iscoe N (1988) A field study of the software design process for large systems. Commun ACM 31(11): 1268–1287. https://doi.org/10.1145/50087.50089
de O Melo C, Santos V, Katayama E, Corbucci H, Prikladnicki R, Goldman A, Kon F (2013) The evolution of agile software development in Brazil. J. Braz. Comput. Soc 19(4):523–552. https://doi.org/10.1007/s13173-013-0114-x
Dingsøyr, T., & Moe, N. B. (2014). Towards Principles of Large-Scale Agile Development - A Summary of Workshop at XP2014 and a Revised Research Agenda In T. Dingsoyr, N. B. Moe, R. Tonelli, S. Counsell, C. Gencel, & K. Petersen (Eds.), Revised Selected Papers of the XP2014) International Workshops, Rome, Italy, 26–30 May 2014 (pp. 1–8). Springer, Cham. https://doi.org/10.1007/978-3-319-14358-3_1
Dix A (1994) Computer supported cooperative work: a framework. In: Rosenberg D, Hutchison C (eds) Design issues in CSCW. Springer, London, pp 9–26. https://doi.org/10.1007/978-1-4471-2029-2_2
Dybå T, Dingsøyr T (2008) Empirical studies of agile software development: A systematic review. Information and Software Technology 50(9-10):833–859
Dyck, A., Penners, R., & Lichter, H. (2015). Towards definitions for release engineering and DevOps. In Proceedings of the 3rd International Workshop on Release Engineering (RELENG), Florence, Italy, 16–24 May, 2015 (pp. 3). IEEE Press
Ebert C, Brinkkemper S (2014) Software product management – an industry evaluation. J Syst Softw 95(0):10–18. https://doi.org/10.1016/j.jss.2013.12.042

Eisenhardt KM (1989) Building theories from case study research. Acad Manag Rev 14(4):532–550. https://doi.org/10.5465/AMR.1989.4308385

Femmer H, Kuhrmann M (2014) Experiences from the Design of an Artifact Model for distributed agile Project Management. https://doi.org/10.1109/ICGSE.2014.9

Fowler, M. (2004). Is Design Dead? Retrieved October 25, 2017, from https://www.martinfowler.com/articles/designDead.html#PlannedAndEvolutionaryDesign

Fricker SA (2012) Software Product Managament In A. Maedche, A. Botzenhardt, & L. Neer (Eds.), Software for People - Fundamentals, Trends and Best Practices (pp. 53–81). Springer, Berlin, Heidelberg

Garcia A, Silva da Silva T, Selbach Silveira M (2017) Artifacts for agile user-centered design: a systematic mapping. In: In Proceedings of the Hawaii International Conference on System Sciences (HICSS-50), Waikoloa Village, Hawaii, 4–7 January, p 2017

Gill AQ, Henderson-Sellers B, Niazi M (2016) Scaling for agility: a reference model for hybrid traditional-agile software development methodologies. Inf Syst Front:1–27. https://doi.org/10.1007/s10796-016-9672-8

Glaser BG, Strauss AL (1967) The discovery of grounded theory: strategies for qualitative research. Aldine, Chicago

Grant RM (1996) Toward a knowledge-based theory of the firm. Strateg Manag J 17(S2):109–122. https://doi.org/10.1002/smj.4250171110

Gröber, M. (2013). Investigation of the usage of artifacts in agile methods. Technischen Universität München

Hummel M (2014) State-of-the-Art: A Systematic Literature Review on Agile Information Systems Development. In Proceedings of the 47th Hawaii International Conference on System Sciences (HICSS), Waikoloa (HI,) USA, 6 -9 January, 2014 (pp. 4712–4721). IEEE. https://doi.org/10.1109/HICSS.2014.579

IEEE (2006) Standard for developing a software project life cycle process. IEEE Standard:1074–2006

ISO. (2011). ISO/IEC/IEEE 26515:2011 Systems and Softw Eng - Developing user documentation in an agile Environment

Jurafsky, D., & Martin, J. H. (2008). Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition (2nd editio). Prentice Hall

Kajornboon AB (2005) Using interviews as research instruments. E-Journal for Researching Teachers (EJRT) 2(1)

Kincheloe JL, Steinberg SR (1998) Students as researchers: critical visions, emancipatory insights - Shirley R. Steinberg, joe L. Kincheloe. In: Steinberg SR, Kincheloe JL (eds) Students as researchers: creating classrooms that matter. Farmer Press, London (UK)/Bristol (USA), pp 2–19

Kuhrmann, M., Diebold, P., MacDonell, S., & Münch, J. (2017). 2nd workshop on hybrid development approaches in software systems development. In M. Felderer, D. M. Fernández, B. Turhan, M. Kalinowski, F. Sarro, & D. Winkler (Eds.), Proceedings of the 18th International Conference on Product-Focused Software Process Improvement (PROFES 2017), Innsbrück, Austria, 2017 (pp. 397–403). Springer International Publishing

Kuhrmann, M., Méndez Fernández, D., & Gröber, M. (2013). Towards Artifact Models as Process Interfaces in Distributed Software Projects. In Proc IEEE *8th International Conference on Global* Softw Eng *(ICGSE) Bari, Italy, 26–29 August, 2013* (pp. 11–20). Bari (Italy): IEEE. https://doi.org/10.1109/ICGSE.2013.11

Kuhrmann M, Münch J, Diebold P, Linssen O, Prause CR (2016) On the use of hybrid development approaches in software and systems development: construction and test of the HELENA survey. In Proceedings of the conference "Projektmanagement und Vorgehensmodelle" (PVM 2016), Paderborn, Germany, 6–7 October, 2016, Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn, pp. 59–68

Liskin, O. (2015). How artifacts support and impede requirements communication. In S. A. Fricker & K. Schneider (Eds.), *Requirements Engineering: Foundation for Software Quality - Proceedings of the 21st International Working Conference (REFSQ 2015), Essen, Germany, 23–26 March, 2015* (pp. 132–147). Springer International Publishing. https://doi.org/10.1007/978-3-319-16101-3_9

Rodríguez, P., Markkula, J., Oivo, M., & Turula, K. (2012). Survey on Agile and Lean Usage in Finnish Software Industry. In *Proceedings of the ACM-*IEEE *12th International Symposium on* Empir Softw Eng *and* Measurement *(ESEM), Lund, Sweden, 17–22 September, 2012* (pp. 139–148). New York, NY, USA: ACM. https://doi.org/10.1145/2372251.2372275

Royce W (1970) Managing the development of large software systems. Proceedings of IEEE WESCON 26:1–9

Schneider, K., Stapel, K., & Knauss, E. (2008). Beyond Documents: Visualizing Informal Communication In *Proceedings of* Requir Eng *Visualization (REV '08), Barcelona, Spain, 8 September, 2008* (pp. 31–40). IEEE https://doi.org/10.1109/REV.2008.1

Schwaber K, Beedle M (2002) *Agile Software Development with Scrum* (1st ed.) Upper Saddle River. Prentice Hall, NJ, USA

Schwaber, K., & Sutherland, J. (2013). The Scrum guide – the definitive guide to Scrum: the rules of the game

Schwaber, K., & Sutherland, J. (2016). The Scrum guide - the definitive guide to Scrum: the rules of the game

Sharp, H., & Robinson, H. (2010). Three "C"s of agile practice: collaboration, co-ordination and communication. In T. Dingsøyr, T. Dybå, & N. B. Moe (Eds.), *Agile Software Development* (pp. 61–85). Springer. https://doi.org/10.1007/978-3-642-12575-1_4

Sharp H, Robinson H, Petre M (2009) The role of physical artefacts in agile software development: two complementary perspectives. Interact Comput 21(1–2):108–116. https://doi.org/10.1016/j.intcom.2008.10.006

Stapel, K., Knauss, E., & Schneider, K. (2009). Using FLOW to Improve Communication of Requirements in Globally Distributed Software Projects. In *Proceedings of* Workshop *on Collaboration and Intercultural Issues on Requirements:* Communication*, Understanding and Softskills (CIRCUS),* Atlanta *(GA), USA, 31 August 2009* (pp. 5–14). IEEE. https://doi.org/10.1109/CIRCUS.2009.6

Stapel, K., & Schneider, K. (2012). *FLOW-Methode - Methodenbeschreibung zur Anwendung von FLOW* (Software Engineering)

Stapel K, Schneider K (2014) Managing knowledge on communication and information flow in global software projects. Expert Syst 31(3):234–252. https://doi.org/10.1111/j.1468-0394.2012.00649.x

Stapleton, J. (1997). DSDM, dynamic systems development method: the method in practice. Addison Wesley Publishing Company, Boston

Stettina, C. J., & Heijstek, W. (2011). Necessary and neglected? An empirical study of internal documentation in agile software development teams. In Proceedings of the 29th ACM international conference on Design of Communication (SIGDOC 2011), Pisa, Italy 3–5 October, 2011 (pp. 159–166)

Strode DE, Huff SL (2014) A coordination perspective on agile software development. In: *Modern Techniques for Successful IT Project management*, pp 1–28

Theocharis, G., Kuhrmann, M., Münch, J., & Diebold, P. (2015). Is Water-Scrum-Fall Reality? On the Use of Agile and Traditional Development Practices. In *Proceedings of the 16th International Conference on Product-Focused Software Process Improvement (PROFES 2015), Bolzano, Italy, 2–4 December, 2015* (pp. 149–166). Springer-Verlag New York, Inc. https://doi.org/10.1007/978-3-319-26844-6_11

Turk D, Robert F, Rumpe B (2005) Assumptions underlying agile software-development processes. J Database Manag 16(4):62–87. https://doi.org/10.4018/jdm.2005100104

VersionOne. (2017). *11th Annual State of Agile Report*

Vijayasarathy LR, Butler CW (2016) Choice of software development methodologies: do organizational, project, and team characteristics matter? IEEE Softw 33(5):86–94. https://doi.org/10.1109/MS.2015.26

Voigt, S., Garrel, J. von, Müller, J., & Wirth, D. (2016). A study of documentation in agile software projects. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '16), Ciudad Real, Spain, 8–9 September, 2016* (p. Article no. 4.1-no. 4.6). New York, New York, USA: ACM Press. https://doi.org/10.1145/2961111.2962616

Wagenaar, G., Helms, R., Damian, D., & Brinkkemper, S. (2015). Artefacts in agile software development. In P. Abrahamsson, L. Corral, M. Oivo, & B. Russo (Eds.), *Product-Focused Software Process Improvement - Proceedings of the 16th International Conference on Product-Focused Software Process Improvement (PROFES 2015), Bolzano, Italy, December 2–4, 2015* (Vol. LNCS 9459, pp. 133–148). Springer International Publishing. https://doi.org/10.1007/978-3-319-26844-6_10

Wagenaar, G., Overbeek, S., Lucassen, G., Brinkkemper, S., & Schneider, K. (2017). Influence of Software Prod Manag Maturity on Usage of Artefacts in Agile Software Development In *Proceedings of the 18th International Conference on Product-Focused Software Process Improvement (PROFES 2017) Innsbrück, Austria, 30 November - 2 December, 2017* (pp. 19–27). Springer, Cham. https://doi.org/10.1007/978-3-319-69926-4_2

Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A (2012) Experimentation in software engineering. Springer Berlin Heidelberg, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-29044-2

Yin RK (2013) Case study research: design and methods (applied social research methods series - volume 5). Case study research design and methods (5th ed., Vol. 34). Sage Publications, Inc., Thousand oaks