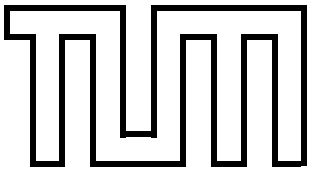


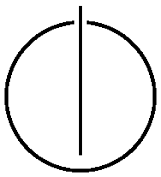
FAKULTÄT FÜR INFORMATIK

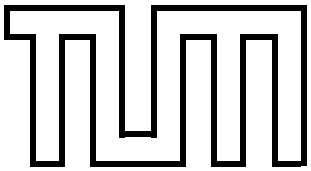
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Computer Science

Investigation of the Usage of Artifacts in Agile Methods

Matthias Gröber





FAKULTÄT FÜR INFORMATIK

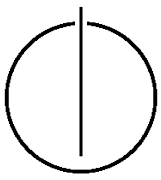
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Computer Science

Analyse der Verwendung von Artefakten in Agilen Methoden

Investigation of the Usage of Artifacts in Agile Methods

Author:	Matthias Gröber
Supervisor:	Prof. Dr. Dr. h.c. Manfred Broy
Advisor:	Dr. Marco Kuhrmann
Submission date:	January 15, 2013



I assure the single handed composition of this master's thesis only supported by declared resources.

Munich, January 15, 2013 Matthias Gröber

Acknowledgements

I like to express my honest gratitude to all the people who supported and encouraged me during the realization of this thesis.

First of all, I would like to express my deepest gratitude to my advisor Dr. Marco Kuhrmann who welcomed me well at the Chair for Software & Systems Engineering and provided me a good introduction to the topic. I am very thankful for his confidence, optimism, advice and guidance I got whenever we met. All the consultations we had were very inspiring and motivating for me.

I also would like to thank my co-advisor Dr. Daniel Méndez Fernández for taking the time to equip me with the necessary expertise and knowledge about study methods. Further I am very grateful for the insightful talks about artifact orientation and modeling. I would like to thank him for all the support he provided when we met or by studying his publications.

My very sincere thanks go to my family, my parents and my sister, for their constant support and encouragement. They were, and are, always supporting me as good as possible. My special thanks go to my father for proofreading and his helpful remarks on the thesis.

Finally I want to express my gratefulness to my girlfriend for motivating me every time when it was necessary. Moreover I thank her for help me to elaborate and stick to my schedule.

Also I am grateful to my flat mates for not complaining about missing check marks on the cleaning roster in the last weeks before submitting this thesis.

Without the effort of all of you, the completion of the thesis would not have been possible.

Abstract

BACKGROUND: There is a lack in formalism regarding artifact-orientation in agile development. Even though artifacts support the values stated in the Agile Manifesto, they attract little attention. Artifacts are poorly defined for agile methods. This thesis starts to close this gap.

AIMS: The target is to investigate the state of the art in artifact-orientation in agile software development. Find and analyze artifacts commonly used in agile development and the relationships between them. Generalize artifacts and relations over different agile methods. The aim of this study is to create an artifact model supporting various agile development methods.

METHODS: A systematic review technique combining a Systematic Literature Review and a Systematic Mapping Study is used to extract data about artifacts from literature. The data was extracted from literature in digital libraries with the help of an automated approach following a strict research protocol.

RESULTS: Artifact-orientation is not very common in agile development but the number of publications is increasing in the last years and the research is maturing. 76 of 489 analyzed publications qualified to contribute to the study. 19 commonly used artifacts and their relationships could be extracted from a well-chosen set of agile methods.

CONCLUSION: The resulting artifacts and relations were composed to an artifact model for agile development that is independent of any specific agile method. It is completely new to the field of agile development to have an empirically generated artifact model. The artifact model could be refined, extended and validated as further work.

Zusammenfassung

HINTERGRUND: Bezüglich der Artefakt-orientierung in der agilen Software Entwicklung herrscht ein Mangel an Formalismus. Obwohl Artefakte mit den Werten des Agilen Manifests harmonieren, erfahren sie wenig Aufmerksamkeit. Artefakte sind für agile Methoden mangelhaft definiert, diese Arbeit beginnt damit diesen Mangel zu beheben.

ZIELE: Das Ziel ist es den Stand der Technik in der Artefakt-orientierung in der agilen Software Entwicklung zu untersuchen. Gängige Artefakte und ihre Beziehungen zu finden, zu analysieren und für unterschiedliche agile Methoden zu generalisieren. Das Ziel dieser Arbeit ist es ein Artefakt Model für verschiedene agile Methoden zu erstellen.

METHODIK: Eine systematische Studie kombiniert aus Systematischer Literaturrecherche und systematischer Kartierung wird verwendet um Informationen über Artefakte zu gewinnen. Die Daten werden mittels Forschungsprotokoll automatisiert aus Einträgen digitaler Bibliotheken extrahiert.

ERGEBNISSE: Artefakt-orientierung ist nicht sehr gängig in agiler Software Entwicklung aber die Zahl der Publikationen und die Forschungsreife nehmen in den letzten Jahren zu. Aus 76 von 489 analysierten Veröffentlichungen konnten 19 Artefakte und ihre Beziehungen extrahiert werden.

ZUSAMMENFASSUNG: Die resultierenden Artefakte und Beziehungen wurden zu einem methodenübergreifenden agilen Artefakt Model vereint. Ein empirisch erzeugtes Artefakt Modell ist es völlig neu für die agile Software Entwicklung Das Modell könnte in der Zukunft verfeinert, erweitert und validiert werden.

Contents

List of Figures

List of Tables

1 Introduction

Agile software development is more and more common in industry. According to the survey [43, p. 8] already 25% of software projects are utilizing agile processes like Scrum or Extreme Programming (XP). And the rate of adopters for agile processes is still growing. These processes and other agile techniques are even playing an important role on time, cost or security critical projects ranging from small application to complex business applications. Agile is also used by big players like Microsoft[1], IBM[2] or Google[3]. Also the academia has an interest in research in this area which can be seen by the plenty of conferences related to this topic. But what is agile all about?

Agile software development (agile development) is defining a whole bunch of methods, principles, practices and processes talking about the values defined in the Agile Manifesto. The agile approach is focusing on how to create software, necessary activities and not on intermediate results on the way to the final product. There is no clear answer related to which intermediate results lead to high quality software. This is where the investigation of this study starts. The intermediate results represent the artifacts created by agile development. Unfortunately there is no unique formal definition and therefore leaving room for interpretation related to structure, relationships and usage.

One common application for artifacts is the domain of controlling where they are used to automatically create metrics. This approach is also recommended for agile development and also applied in practice hence there are artifacts.

But what artifacts are common in agile development and how can they be extracted? The study done investigated in artifacts in agile development and the thesis provides an overview of the common artifacts.

1.1 Motivation

> Manifesto for Agile Software Development
>
> 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 [10].

Agile development is built on the Agile Manifesto which states the common values of all what is called "agile" today. The Agile Manifesto values "working software" which

1 www.microsoft.com
2 www.ibm.com
3 www.google.com

can be seen as collection of artifacts forming a final software product. But it would be interesting to understand witch artifacts contribute to working software. Another value of the Agile Manifesto is "responding to change". Which artifacts are affected by changes and how do they simplify response to change. To answer these questions there is a need for an artifact model which supports agile development.

Since 2002 there has been a large amount of research publications around agile development. A significant increase in publications is noticeable since 2006 according to the study [23] as shown in Figure 3.1. The number of published studies is still growing and the quality of studies in the field is maturing. This makes it more and more interesting for secondary studies like the study done in the context of this thesis.

1.2 Problem Statement

Artifacts are often mentioned in literature and used a lot in practice but are not very often correctly described. Processes and practices create and use artifacts but seldom define them. Artifacts may be shared between processes and practices and may be connected to other artifacts. Not all of these relationships are necessarily documented. The task is to extract existing artifacts and connections and develop an artifact model for agile development.

1.3 Research objective

The aim of this study is to investigate the state of the art in artifact-orientation (see Section 1.6.2) in agile development. Therefore all artifacts that are commonly used in agile development shall be found. It is necessary to investigate the relations between those artifacts. In case they are similar over different processes and practices it is the task of this study to find commonalities. The study will generalize artifacts of different processes and practices to develop a common artifact model for different processes or practices. As a final goal an artifact model for agile development shall be created.

1.4 Contribution

This is a systematic case study which combines methods of a Systematic Literature Review (SLR) and a Systematic Mapping Study (SMS). This approach is proposed by Petersen et al. which see no reason for not use several analysis methods in the same study [76]. Based on this study towards artifacts in agile development the following contribution is made.

- Systematic overview of found publications
- Systematic map of found artifacts
- Investigation on relations and classification of artifacts
- Artifact model for agile development

1.5 Related work

In the field of artifact oriented agile development there has not been too much research. Hence all studies about agile or artifact-orientation can be used.

A Systematic review on agile development. Dybå and Dingsøyr identify empirical studies on agile software development until 2005. They investigate in the four thematic groups introduction and adoption, human and social factors, perceptions of agile methods and comparative studies. They found, that almost all studies were investigating XP and no other processes. They conclude criticizing the low number and a lack of quality and strength of evidence in the found studies on agile software development and propose a common research agenda to follow [27].

A decade of agile methodologies. Dingsøyr et al. undertook a systematic review which provides a good overview of the publications in last ten years since the Agile Manifesto was published. In the review the number and distribution over years and countries is detailed. He emphasizes that the number of studies published has increased significantly since 2005. Also the quality seems to increase according to this study. The interest in other topics like Scrum, Flow based and Lean approaches, is increasing as well. They also provide directions for future research. Another important conclusion of this study is the need for more methodology independent theory underpinning the agile development [23].

Agile Software Development Model (ASDM). Janus tries to create in his study a common agile software development model by formalizing the Agile Manifesto and a set of processes to derive a combined model. Then he applies rules to derive a formal model in form of sets of characteristics for agile development. A part of his work describes how to derive artifacts from the Agile Manifesto, Scrum and XP [50].

Discussion. It can be concluded that most related work concentrates on agile development without a special attention to artifacts. Also a first tendency towards searching for generalization instead of treating agile development no longer as a set of individual methodologies can be seen. This study concentrates on the lack of artifact-orientation and follows the trend of generalization of agile methodologies.

1.6 Fundamental and Terminology

1.6.1 Fundamentals

Agile is only an umbrella term created during the conference where the Agile Manifesto was born. This is why it is necessary to look what's under the hood. Certain processes and practices are considered to be agile so let's see which ones are commonly used. Most representative processes and practices for agile development can be found in the studies of VersionOne[4] [94], Peter Haberl et al. [43] and Abrantes and Travassos [3]. Details can be found in the papers.

Top five common processes

- Scrum
- XP
- Kanban
- Feature Driven Development (FDD)
- Agile Unified Process (AUP)

4 www.versionone.com

Top ten common practices

- Stand-Up meetings
- Iteration planning
- Test Driven Development (TDD) and Unit tests
- Burn-down and measuring
- Retrospectives
- Continuous integration
- Velocity / Sustainable pace
- Coding standards
- Refactoring
- Collective ownership of code

1.6.2 Terminology

This section will explain the terms used in this thesis.

Artifact: Artifacts have different meaning in different areas of computer science. In image processing artifacts are commonly used in the sense of an inaccurate observation or an error in an image. It's a part of an image which was not recorded or coded as intended.

In software engineering artifacts have a complementary meaning and are no longer associated with a failure. "An artefact is any form of representation of significant (in the sense of required by a process) intermediate or final work product (result) of the development process" [65]. The difference is, artifacts are used and manipulated to develop software and are created by intention.

Artifact-orientation: According to Méndez Fernández et al. "artefact-orientation gives a more detailed view onto the corresponding results structure with the purpose of enabling seamless modelling of consistent results without having to take into account the variability of volatile processes and the compatibility of methods." [66]

Agile development: The IEEE Computer Society defines in Standard 26515 agile development as "software development approach based on iterative development, frequent inspection and adaptation, and incremental deliveries, in which requirements and solutions evolve through collaboration in cross-functional teams and through continuous stakeholder feedback" [49].

1.7 Outline

The next chapter describes the design of the study conducted to achieve the research objectives. Chapter 3 presents the results of this study starting with the study population of papers conducted for the study and followed by answers to the individual research questions. In the following chapter a resulting artifact model is presented. The last chapter provides a summary of the findings and an outlook about what research could be done related to this topic in future.

2 Case Study Design

The reason for this systematic case study is to find commonly used artifacts in agile software development. As they are not properly defined nor collected at a centralized place they cannot be found easily. The term agile development is a generic term as defined by the IEEE Standard (see Section 1.6.2) hence it is essential to define the scope of it for the study. The study will concentrate on the main agile processes such as Scrum or XP for example and the main practices like TDD or Continuous Integration (CI) to find artifacts that contribute to a general agile artifact model.

Kitchenham and Charters proposed to use a SMS to evaluate if there is enough evidence on a topic and if a SLR makes sense [55]. Because artifact-orientation is not very common in agile development as visible from the related work section there might be a lack of evidence within the primary studies. This leads to the approach to combine a SMS with a SLR for this study. The goal is to provide a wide overview with the methods of a SMS but extract data about artifacts on the level of detail used in a SLR. For the design of the study the structure proposed by Kitchenham and Charters [55] for SLRs is followed. But some of the methods described by Petersen et al. [76] are injected to the single protocol steps of the SLR. Also the detailed report of reviewed publications will be omitted due to the lack of significance of each single publication.

The research protocol that was followed is described in this chapter divided into the following sections. First the research questions are defined. Then Section 2.2 describes the search process which is followed. The next section describes the criteria and the procedure to include promising studies from the search results as qualified publications. Afterwards the procedure to extract relevant data from the studies is introduced and the protocol step for the data analysis procedure is described. Deviating from Kitchenham and Charters' protocol in the last section research limitations are considered.

Overview

2.1 Research Questions

The overall target of this study is to investigate the state of the art in agile artifact-orientation. Therefore agile artifact-orientation is split into several subfields dealing with the agile processes and practices, the degree of maturity, common artifacts and their relations. The study tries to answer the following research questions related to artifacts and the subfields:

RQ1: Which agile software engineering processes and practices consider artifacts to which extent?
This addresses common artifacts in agile development and their relation to practices and processes or parts of them.

RQ2: Which degree of maturity have the agile development processes and practices and their artifact oriented concepts with respect to their research type facet?
This addresses mainly the research type facets and their behavior since 2001. The research type facet can shed light on the maturity of scientific studies in a domain.

RQ3: Which artifacts are proposed for agile development and could contribute to a cross-process and cross-practice artifact model?
Artifacts being produced and artifacts that are utilized by agile development are contributing in the same way.
RQ3a: Which of these proposed artifacts are commonly used?
RQ3b: Which of these artifacts are used in the domain of controlling?

RQ4: How are the common artifacts related to each other and what model does result from these relationships?
This question aims at giving an answer if the common artifacts have enough relations to build a complete artifact model upon the results.

The first two RQs are developed for quantitative investigation and the second two for qualitative investigation. Thus this study goes beyond a classical SMS and complements with a SLR.

2.2 Search Process

To find the artifacts used in agile development, literature research is conducted in a hybrid form of a SMS and SLR as described by Petersen et al. [76] and Kitchenham and Charters [55]. The primary studies are found based on an automatic search within digital libraries. This section describes how the search is performed starting from data sources continuing with the search terms, sub-queries and the query itself.

2.2.1 Data Sources

Because most relevant literature about computer science is digitally available, digital libraries are used as sources for the study. To not miss any important result different libraries are included. The selected libraries have been chosen as they are the major digital resources in computer science subscribed by the University. The sources for the study are following libraries:
ACM Digital Library, IEEE Explore, SpringerLink and ScienceDirect.

2.2.2 Search Terms

Titles, Abstracts and Keyword fields of all sources are scanned by queries. The overall search result is the conjunction of the results of all performed sub-queries (SQ) in all digital libraries. Sub-queries need to be adjusted to the input masks of the digital libraries before performing the search. Each sub-query is composed of the conjunction of the name of an agile process, practice or a generic agile term with a synonym for artifact. The synonyms for artifacts used are listed below:

- artifact, artefact
- work item, work product, work result
- deliverable
- manufacture

Agile processes and practices. According to several studies [43, 94] the most commonly used agile processes in industry are the following five. This is why the queries are limited on these terms. The searched process models are:

- Scrum (most common)
- XP
- Kanban
- Feature Driven development (FDD)
- Agile Unified Process (AUP) (least common)

In the same way the most used practices are identified according to a study of Abrantes and Travassos [3] and the yearly surveys of VersionOne [94]. All of the selected practices have been found in both publications. The search is limited to the ten most common agile practices to find existing artifacts. The selected practices are:

- Stand-Up meetings
- Iteration planning
- TDD and unit tests
- Burn-down and measuring
- Retrospectives
- Continuous integration
- Velocity / sustainable pace
- Coding standards
- Refactoring
- Collective ownership of code

Terms for Agile. To not miss artifacts not correlated with specific processes or practices but common to agile software engineering another query is created by a synonym of artifact and an agile term combined with software. Agile terms used are the following:

- agile
- agility
- lean
- light weight

Agile and Controlling. In addition to address the third research question the sources are scanned for a synonym of artifact and a common term for controlling combined with a synonym for Software Engineering (SE). The used terms are:

- controlling
- measuring

- reporting

And for SE the following terms are used:

- software development
- software engineering

Query construction. The following table shows the individual steps in the process of query creation from the individual terms identified above till the complete query.

Query name	Search string
ARTIFACT	(artifact **OR** artefact **OR** work item **OR** work product **OR** work result **OR** deliverable **OR** manufacture)
PROCESS	(scrum **OR** extreme programm **OR** kanban **OR** feature driven develop **OR** agile unified process)
PRACTICE	(Stand up **OR** (Iteration **OR** Sprint) plan **OR** test driven **OR** unit test **OR** burn down **OR** retrospective **OR** continuous integration **OR** (velocity **OR** sustainable pace) **OR** coding standard **OR** refactoring **OR** collective ownership)
AGILESE	(agile **OR** agility **OR** lean **OR** light weight) **AND** (software)
SE	(software **AND** (development **OR** engineering))
SQ1	PROCESS **AND** ARTIFACT
SQ2	PRACTICE **AND** ARTIFACT
SQ3	AGILESE **AND** ARTIFACT
SQ4	(controlling **OR** measuring **OR** reporting) **AND** SE **AND** ARTIFACT
FINAL	SQ1 **OR** SQ2 **OR** SQ3 **OR** SQ4

Table 2.1: Sub-queries and the final search string

2.3 Inclusion and Exclusion

This section defines a way how to classify the results of the beforehand defined query and select them to qualify for the study. First some classification criteria are defined. Afterwards the procedure is described how to use the classification criteria to select publications from the results of the query.

2.3.1 Inclusion and Exclusion Criteria

In general only literature published between Feb 2001 and Sept 2012 (present) is subject to the study as before the term agile wasn't emerged. In case of studies published in multiple papers, the most complete version is used.

For further filtering of the results the selection criteria listed below are applied. The inclusion and exclusion criteria are determining the studies which are relevant to answer the research questions.

The criteria are chosen in a way that all publications can be either included or excluded but not without selection criterion. All criteria were tested and adjusted in trial runs of the study and were tagged with an abbreviation for better traceability of results.

Inclusion Criteria. The inclusion criteria are listed in Table 2.2.

Abbr.	Description
(xN)	Study is mentioning at least one artifact by name and a process or practice related to the artifact if available.
(xD)	Study is describing at least one artifact in detail and mentions a process model or practice related to the artifact.
(xC)	Study is mentioning at least one artifact in context of controlling and agile development
(xG)	Study is classifying a number of artifacts with an arbitrary characteristic. (Possible characteristics could be e. g. textual artifacts, test artifacts)

Table 2.2: Inclusion criteria and abbreviations

Exclusion Criteria. The exclusion criteria are listed in Table 2.3.

Abbr.	Description
(NA)	Study is mentioning artifact(s) not in the context of agile development.
(ND)	Study is mentioning artifact(s) only in general without name, description or classifying characteristic. But in the Context of agile development
(OD)	Study is outside the discipline of software engineering, process engineering, software development or controlling.
(NF)	Study is not available as digital full text or cannot be used due to copyright issues.
(OL)	Study is not written in English.

Table 2.3: Exclusion criteria and abbreviations

2.3.2 Inclusion and Exclusion Procedures

Now follows a detailed description how to apply the criteria on the results to select cases in the results. All studies found are checked against the inclusion and exclusion criteria by a single researcher in two stages. In the first stage publications are scanned to achieve exclusion according to the criteria **OD**, **NF** and **OL**. This is done with help of the metadata, abstract and a random view into the paper. If the study is definitely associated with another discipline or there is no full text in English available it is added to the exclusion list. This list contains a column for the document identifier, title, corresponding subquery and the exclusion criterion.

After this first check all studies so far identified as candidates will be tested against the other criteria by queries searching within the whole text. In this stage also studies are excluded that do not deal with agile development **NA** or do not provide detailed information about artifacts **ND**. There are also studies excluded according to the three above mentioned exclusion criteria. In this step the excluded studies are noted in the exclusion list without the originating search number as there could be multiple entries. The papers in the exclusion list are manually reviewed to not accidentally miss an important one for further research.

The selected studies are classified according to the inclusion criteria. One study can be classified by multiple criteria. Meta information and full text of the selected studies are imported into a document management system (citavi) for further processing. For each study a quality assessment is performed and a data collection step as described in Section 2.5 is executed.

2.4 Quality Assessment

The quality assessment of publications is not following theory described by Kitchenham in [55]. As most of the studies are not dealing with artifact-orientation Kitchenhams quality assessment approach defining quality scores does not seem to be a good fit. Instead the research type facet which is an existing classification of research approaches by Wieringa et al. [95] is used. Petersen et al. introduced the applicability of facets to proof the quality of results in a SMS [76]. Wieringa describes the facets listed in Table 2.4.

Category	Description
Validation Research	Techniques investigated are novel and have not yet been implemented in practice. Techniques used are for example experiments, i.e., work done in the lab.
Evaluation Research	Techniques are implemented in practice and an evaluation of the technique is conducted. That means, it is shown how the technique is implemented in practice (solution implementation) and what are the consequences of the implementation in terms of benefits and drawbacks (implementation evaluation). This also includes to identify problems in industry.
Solution Proposal	A solution for a problem is proposed, the solution can be either novel or a significant extension of an existing technique. The potential benefits and the applicability of the solution is shown by a small example or a good line of argumentation.
Philosophical Papers	These papers sketch a new way of looking at existing things by structuring the field in form of a taxonomy or conceptual framework.
Opinion Papers	These papers express the personal opinion of somebody whether a certain technique is good or bad, or how things should been done. They do not rely on related work and research methodologies.
Experience Papers	Experience papers explain on what and how something has been done in practice. It has to be the personal experience of the author.

Table 2.4: Research Type Facets of Wieringa summarized by Petersen [76, 95]

This study interprets philosophical papers in a slightly different way than described by Wieringa. Here an emphasis is put on the new interpretation of existing facts by using a different view point. This is done to compensate for a lack in scientific proven taxonomies and conceptual frameworks in agile development.

Utilizing the classification in research type facets it is possible to ensure that found results do not only origin from an individual research approach but are reconfirmed and supported by different approaches. If for example a finding is supported by a validation research and an experience paper we can assume it was successfully applied and thus delivers a quality to build on. If a result of the study is only relying on opinion- or philosophical papers or solution proposals there might a be a risk that the finding is not proven sufficiently.

2.5 Data Collection

All studies that passed the selection procedure are considered one by one for data collection. After scanning the content of a selected study the information is extracted into a table called the inclusion list with entries of the format showed below in Table 2.5.

Each Study is identified by its DOI or ISBN (see DOI column in Table 2.5). In case there is no identifier the Title and Author is used to identify the document. A progressive number is introduced to index the qualified results. There are four options for publication types: Conference paper, Journal article, Book / Chapter or Other. Additionally the Research Type Facet according to Petersen and Wieringa (see Table 2.4) is collected from the papers. Each entry has a field for the number of resulting artifacts. The names are listed in the next column. The practice and process field can contain one of the above mentioned practices or process models to keep track where artifacts origin from. The describing field (Yes / No) is identifying if there is a detailed description or a formal model for an artifact available. The Classification section is only filled for papers suggesting a characteristic or category for a group of artifacts.

2.6 Data Analysis

The tabulated inclusion list contains basic data about each study. The data can be categorized by practice or process. For further analysis an extra table is created to list all artifacts of each study. Therefore all artifact-names are noted together with a reference to its origin and potentially a synonym which is used in other documents for the same artifact. The synonym column in this table is also used for word stemming to homogenize the found artifacts. This table containing all artifacts can be used to answer research questions about the occurrences of artifacts in agile development. Further analysis is done by creating tag clouds for different categories and attributes of artifacts. The found artifacts are analyzed regarding the frequency of occurrence and the research type facets dealing with. The artifacts with a high frequency of occurrences are defined as commonly used artifacts. These are further analyzed regarding relationships and association with processes and practices.

The Table is reviewed to answer the Research questions as follows:

RQ1: Which agile software engineering processes and practices consider artifacts to which extent?
All qualified papers will be evaluated based on their classification by process or practice. The artifacts found will be related to the processes and practices to see if certain artifacts are only used in certain processes or practices.

RQ2: Which degree of maturity have the agile development processes and practices and their artifact oriented concepts with respect to their research type facet?
The degree of maturity will be measured with the help of the research type facets described in Section 2.4 and the constellation of different facets over time. By building a chronological ordering of facets the research maturity can be determined. Some facets are independent of the chronological ordering but nevertheless shed light on the maturity.

RQ3: Which artifacts are proposed for agile development and could contribute to a cross-process and cross-practice artifact model?
All artifacts named in the included papers are extracted. Word stemming and synonyms are used to find commonality. By counting the instances of all different artifacts it is possible to find common artifacts. Common artifacts need to be mentioned

Column name	Values	Description
Inclusion criteria	xN, xD, xC, xG	Lists the reason why a paper is part of this list.
Number	Integer	Indexing for easy referencing.
Reference	DOI, ISBN or ISSN/EISSN	A reference to find the paper.
Title	String	Self explanatory.
Authors	String	Self explanatory.
Year	Integer	Self explanatory.
Keywords	String	Keywords provided by the author. If not available Keywords used in the digital library.
Publication type	Conference, Journal article, Book / Chapter, Other	Information about which type of publication was used
Research type facet	Research type facets as defined by Wieringa. See Table 2.4 for details.	Facet of the research that will be used for quality assessment.
Number of artifacts	Integer	Number of different artifacts that were found in the paper.
Name of artifacts	String	List of names of all artifacts that were found in the paper.
Agile process or practice	Processes and practices as defined in Section 1.6.1	The process or practice that is discussed in the paper.
Describing	Yes / No	This field indicates if the found artifacts or classification categories are only named or described with more details.
Classification	String	Possibilities to categorize or classify artifacts.

Table 2.5: Format of inclusion list table

in at least three different papers classified with certain facets. Valid combinations of facets are explained in Section 3.4. All homogenized artifacts resulting from papers marked with the inclusion criterion **xC** help to answer RQ3b related to controlling.

RQ4: How are the common artifacts related to each other and what model does result from these relationships?
The relation of different artifacts can either be answered based on the papers describing relations or with respect to the naming of artifacts in context of the same process or practice. Based on the papers a classification is created that also helps to find relations between artifacts. In this RQ it will also be investigated if there are different types of relationships between artifacts. The relations shall be consulted to construct an artifact model.

2.7 Research Limitations

Due to the automatic approach to search publications, papers that are not fully digitalized or indexed might not be found. In case of changing processes or practices commonly used in agile development the artifacts also could change and the findings might become invalid. As the research team consists only of one single person, this might lead to an increased risk of internal validity.

3 Study Results

The results of the applied research protocol will be presented in this chapter. The first 3 sections provide a quantitative view of the results. The last 2 sections are a qualitative analysis of the collected data. The key findings are based on the 19 different most common artifacts that result from 76 qualified publications. Scrum and XP are the best investigated processes in the qualified papers. The analysis of the publications reveals a clear tendency towards a maturing area of research. The research is done mostly in a steady cooperation with industry which can be seen in the amount of evaluation studies and experience reports. The found artifact set shows a high cohesion which seems to be a good baseline to develop an artifact model.

Overview

3.1 Study Population

A total number of 540 papers were returned by the query in all the defined digital libraries. After removing the duplicates 489 were left. About 10% of the found papers were duplicates but only a few were identical papers published in different libraries. The largest portion of the duplicates was created because the search was performed by subquery due to the complex search string. This was necessary because the libraries allowed only a limited number of logically linked search terms. The distribution of qualified publications over the sources can be seen in Table 3.1. About half of the papers are from IEEE Explore followed by ACM DL and Springerlink with $\tilde{1}00$ results. In the first stage all

Library	Results	Removed in stage one
ACM DL	109	36
IEEE Explore	263	72
Science Direct	65	39
Springerlink	103	19
Total	540	166

Table 3.1: Search results by source

papers are excluded that are not available as full text in English and which are related to other disciplines. Even after the first exclusion IEEE Explore remains the library with the most contributions with about 190 results.

Inclusion and Exclusion. In stage two the Exclusion criteria **NA** and **ND** are checked. From the second stage on the results are no longer listed and evaluated by database as some papers result from multiple sources. The large number of papers excluded for the lack of details **ND** is an indicator for the usage of the word artifact in a common sense without worrying about a correct definition or detailed examples. In total 413 publications have been excluded (see Table 3.2). After the exclusion of all irrelevant papers a total

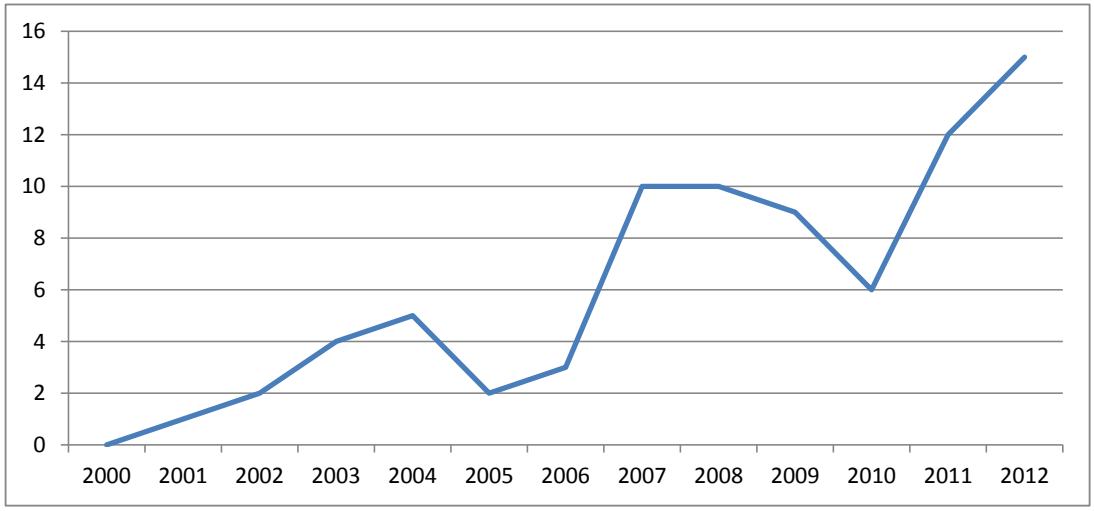
Exclusion criteria	Stage one	Stage two	Total	Inclusion criteria	#
NA	0	88	88	xN	71
ND	0	131	131	xD	8
OD	163	5	168	xC	5
NF	1	21	22	xG	32
OL	2	2	4		
Total	166	247	413		116

Table 3.2: Resulting exclusions and inclusions by criteria

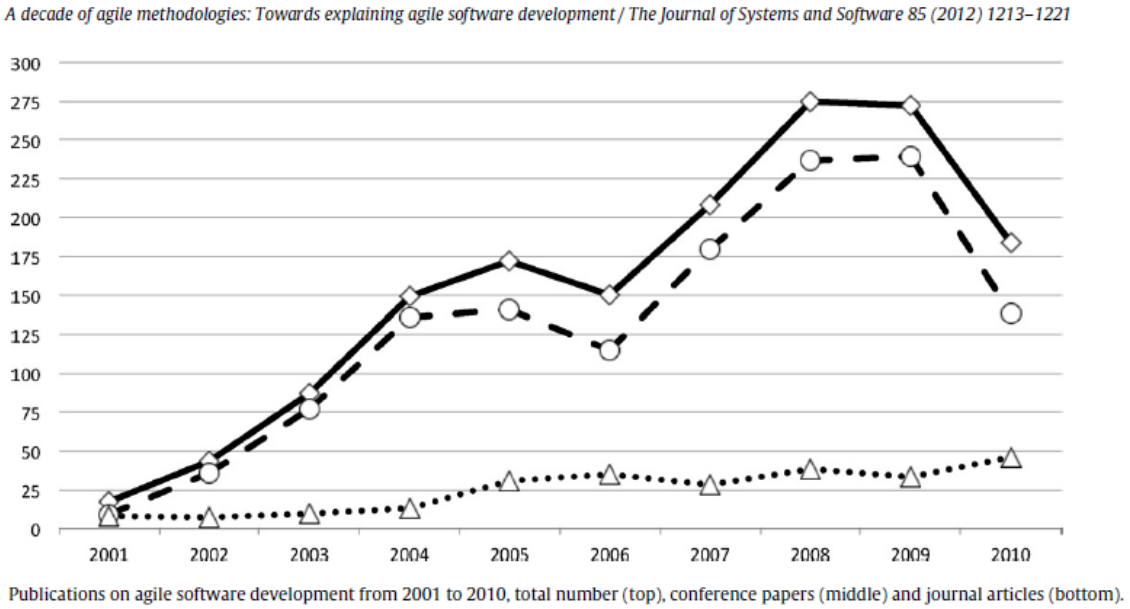
set of 76 papers is left for data collection and analysis. The included papers are classified according to the inclusion criteria where one paper can be mapped to multiple inclusion criteria. The distribution of the inclusion criteria is shown in Table 3.2 The high number of the exclusion criteria **xN** is a hint for a lack of artifact-orientation and formalism in agile development.

Publications over Time. By plotting the number of included publications over time there is an immense similarity to the graph of Dingsøyr et al. [23] published in his SLR. Both graphs show a steady increase in papers but a local decline on studies for the years

2006 and 2010. The comparison of these two graphs can be seen in Figure 3.1. As the results of this study range from 2001 until 2012 and the one from Dingsøyr only until 2010 the upper image continues after the decline in 2010. This comparison shows that papers related to agile artifact-orientation represent a similar percentage of the overall agile papers and probably don't follow other trends.



(a) Papers resulting from this study



(b) Papers resulting from Dingsøyr et al. [23]

Figure 3.1: Comparison of the number of resulting artifact oriented papers and all agile papers over time.

Publication Type. The type of publications was also evaluated as demonstrated in Figure 3.2 About two-thirds of the 76 qualified publications result from conference proceedings. And only 15 results were published in journals. A similar amount of 10 publications are books. There was only one paper that could not be classified in the before mentioned categories. This distribution shows that most of the research is presented at conferences.

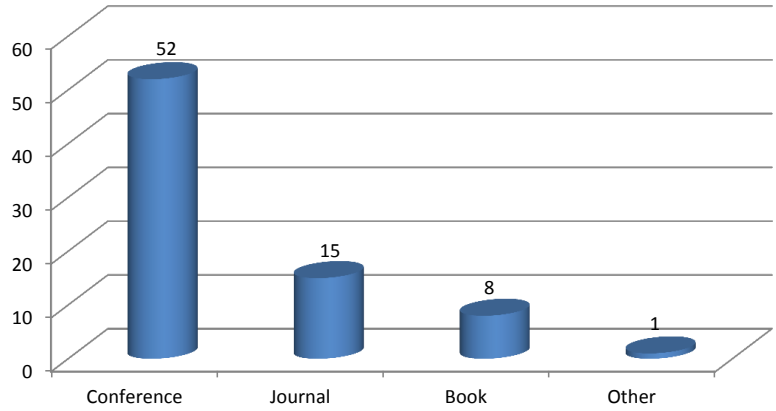


Figure 3.2: Distribution of publication types for qualified papers

Artifact vs. Work Item. As stated in the study design, there are a couple of synonyms in use describing the concept of artifact. Thus the included papers were analyzed for the word for or spelling of artifact they used. The distribution found can be seen in Figure 3.3. Because the word "artifact" was used in most cases in this way it is used through the entire paper.

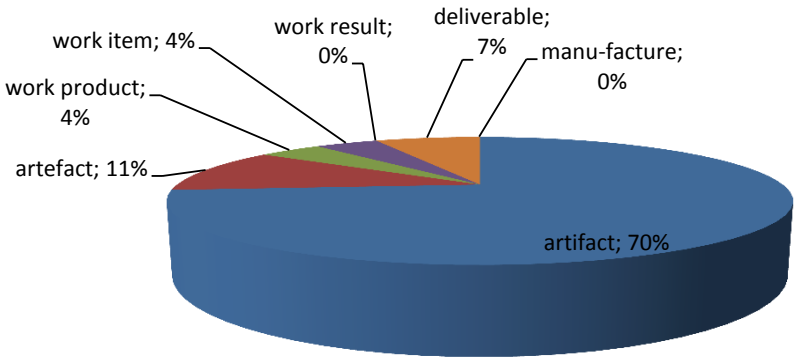


Figure 3.3: Distribution of terms used to describe artifacts

Keywords. Another interesting aspect is the used keywords within the qualified papers. The used keywords lead to the research field that deals with artifact-orientation in agile development. To visualize the keywords a tag cloud shown in Figure 3.4 was created. In the tag cloud the font size represents the commonality of a tag or word in the underlying data. Words that are more common in the data thus result in bigger tags in the cloud. Prominent keywords are of course "agile", "software engineering" and "software development" as well as "management" and "programming". Against all expectations the individual processes are rather under represented as keywords.

Research Type Facets. Analyzing the research type facet of the included 76 papers reveals that four of six facets are almost equally present. Only five results are classified as opinion papers and there is no philosophical paper included. For details see Figure 3.5. An even distribution over many facets is important, as they are used for the quality assessment of the results.

Figure 3.4: Key words of all included papers

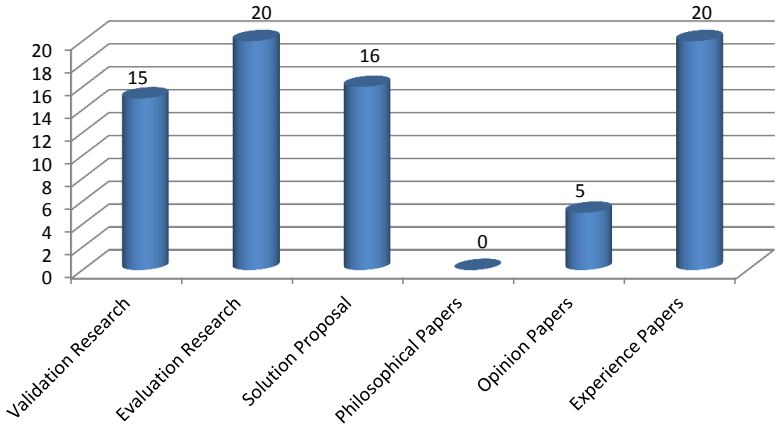


Figure 3.5: Distribution of research type facets within results.

3.2 Artifact-orientation in processes and practices - RQ1

This section provides an insight which processes and practices contribute to artifact-orientation and which are independent of artifacts. The five selected processes and ten selected practices are not evenly represented within the results of this study. Therefore the papers have been attributed with processes and practices where each paper can be linked to none or multiple processes and practices. Resources dealing with agility in general without reference to any process or practice are not considered for this evaluation and aren't attributed. Table 3.3 shows the relationship between papers and process or practice. Because some papers are attributed to multiple or none process or practice the total number deviates from 76. Against the expectation that AUP will provide lots of information because of more than 50 official defined deliverables, Scrum and XP are the most common processes to deal with artifacts in the resulting papers. The most common practices are TDD and Refactoring as expected.

But what was unexpected is that no paper explicitly deals with artifacts related to measuring and burn-down, velocity and collective ownership of code.

This basic statistics also can be found in the map showing the most common artifacts with the processes and practices they result from. In Figure 3.6 it is visible, that almost all artifacts are present in Scrum and XP where Kanban, AUP and FDD only provide almost no additional artifacts. These two processes seem to rely on artifacts even if not specified exhaustive in their definitions. The amount of distinct artifacts used in both Scrum and XP seem to give an answer why the two processes are often combined (14%

Process	#	Practice	#
XP	27	TDD	9
Scrum	21	Refactoring	5
Kanban	2	pair programming	4
AUP	1	continuous integration	4
FDD	1	iteration planning	4
		stand-up	2
		retrospective	1
		Burn-down and measuring	0
		Velocity	0
		Collective ownership of code	0

Table 3.3: Number of papers by process and practice

of agile adopters [94]).

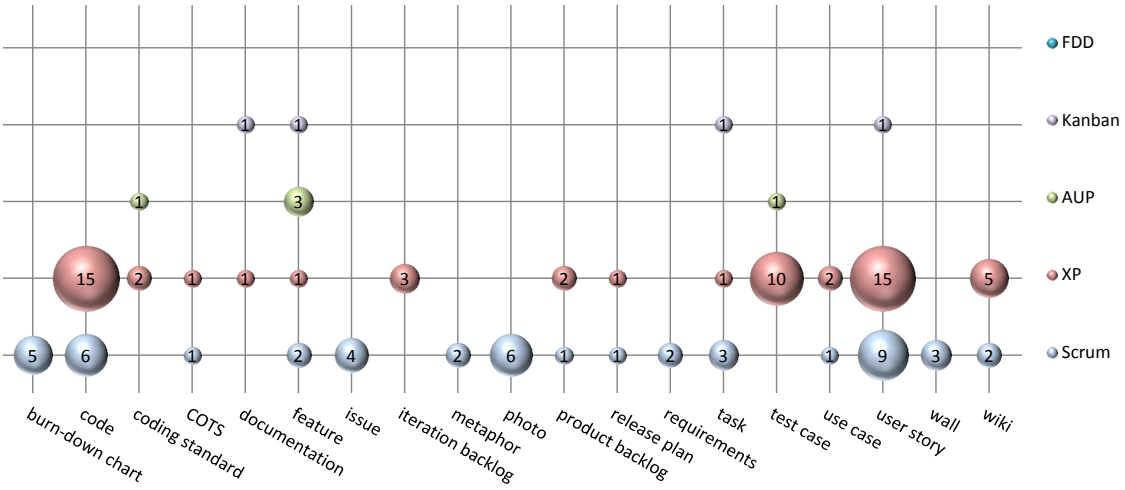


Figure 3.6: Map of processes and artifact names

The map for practices and artifact names (Figure 3.7) shows almost no contributions to artifacts in comparison to Figure 3.6. Only eight artifacts are part of more than one practice where twelve artifacts are part of more than one process. The different practices can interact and work on a common target via these intersecting artifacts.

3.3 Maturity of research on agile processes and practices - RQ2

This section tries to answer how mature the research results for agile development are. Therefore the research type facet is considered. The six facets (see Table 2.4) give a hint related to the maturity of research as most of them can be ordered in a chronological way. Usually in an immature field there are lots of solution proposals which may be validated after a period of time. While maturing experience papers are published describing the implementation in practice. They are more a hint for industry adoption than for research maturity. Opinion papers also occur while maturing because authors give their personal impression before having scientific evidence about the correctness of their opinion. In a mature research field you can find lots of evaluation researches that evaluate well imple-

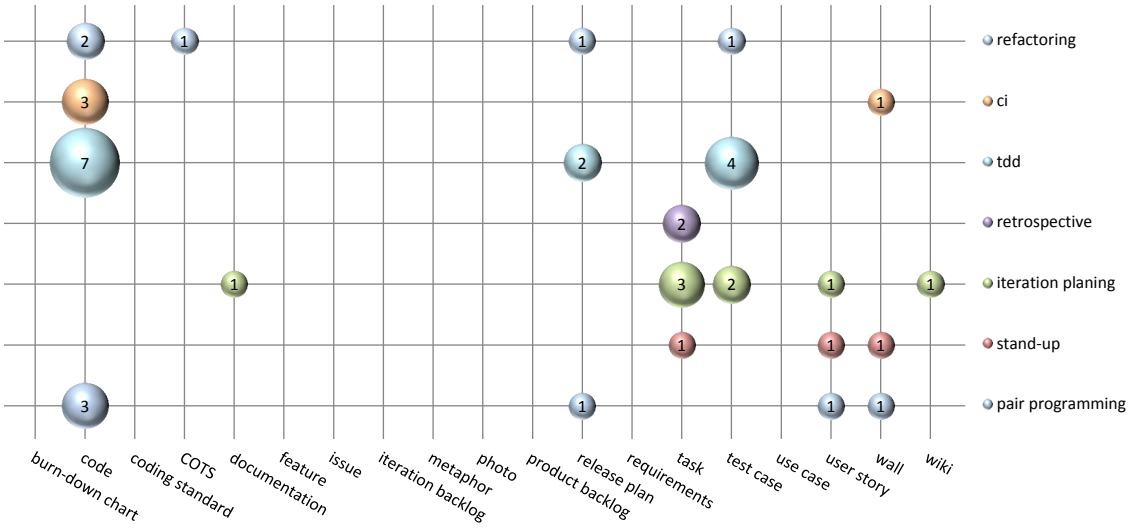


Figure 3.7: Map of practices and artifact names

mented concepts. Philosophical papers do not provide any information about maturity as they could give another view on a topic at any given degree of maturity.

Relying on this concept the presence of evaluation research is an indication for a mature research field. Solution proposals and validation research without support of evaluation give a hint for an immature research field. The presence of solution proposals and validation research until now shows that there is still a lot of innovation ongoing. Experience reports show the industry adoption.

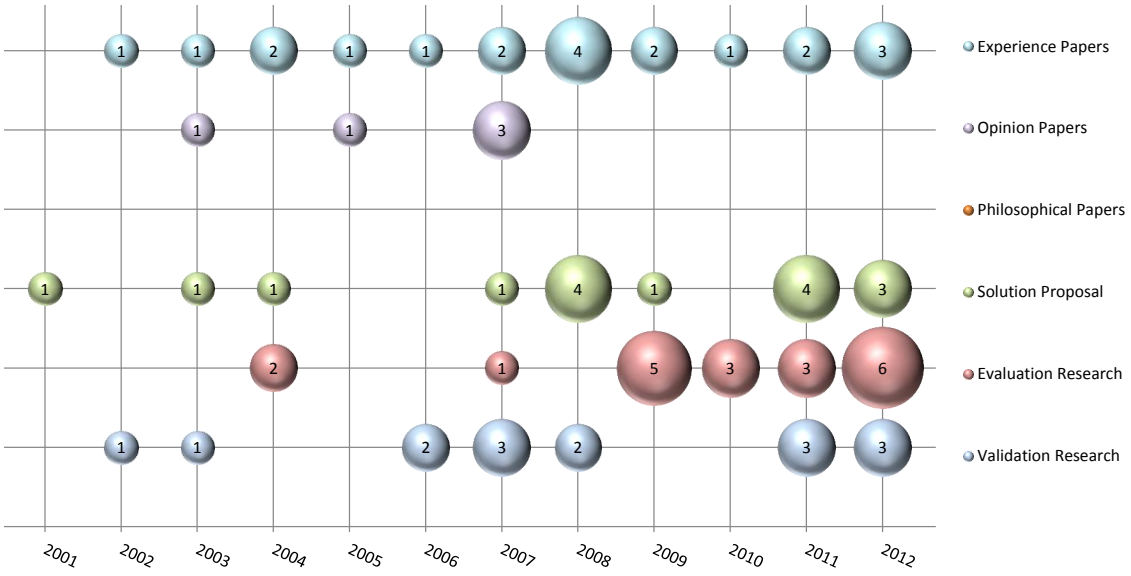


Figure 3.8: Map of publications per research type facet per year

The described approach is used to analyses the map of publications per research type facet per year as illustrated in Figure 3.8. Within the 76 qualified papers there is steady number of experience reports over the years which represents the wide adoption of agile practices in the industry and their roots in best practices used. There had been at least one experience report dealing with artifacts per year since 2002. The amount of solution proposals and validation research in the last two years shows that there is still a lot of ongoing research around agile development and artifacts creating new insights. The

enormous increase in evaluation research since 2009 (three to six papers per year) definitely shows a maturing area of research proving more and more findings. Considering only these evaluation studies they provide a tremendous evidence base for artifacts in agile SE with 90 appearances of artifact names.

This shows huge potential for the future as research is maturing but there is still innovation potential paired with interest from industry for funding research and applying results.

3.4 Artifacts resulting from the literature search - RQ3

Figure 3.9: Tag cloud for resulting artifacts

This section describes the results of the data analysis with respect to the found artifacts. Within the 76 papers only three do not contribute to a list of artifact names and from the remaining 73 publications results a total of 314 named artifacts. This list contains each artifact once if named multiple times per paper but multiple times if named in multiple papers. In a first step these artifacts have been visualized in the tag cloud in Figure 3.9 to get a first impression on common artifacts and their distribution. Each word in an artifacts list entry is its own tag in the cloud. Thus 'user story' and 'story card' result in three different tags not two. The cloud shows that code, test, backlog and story are the main artifacts, related to agile development, that were found. A description of the found artifacts follows later in Table 4.1.

Within the 314 named artifacts there are many that implement the same concept but are called different in different papers. To avoid redundant results word stemming and renaming of artifacts with synonyms is implemented. After stemming and introducing

synonyms for same artifact names a total of 162 different artifacts can be distinguished. Synonyms introduced after stemming can be found in Table 3.4.

Synonym	Replacements	Description
code	class, method, program source, markup file, code packages, system code, production code, program code, java code, source code, gui code, unit code	Accumulates all types, parts and concepts of source code.
test case	test code, unit test, test requirement, test case code, test, acceptance test, automated test case, stress test, automation test, functional test, automated acceptance test case, integration test, test case	Accumulates tests and test descriptions of all different types.
user story	story, story card, shared story, user story card, index card user story, user stories with acceptance criteria, user story with usability issues	Accumulates all variants of user stories.
issue	defect list, bug, issue tracking, change request list, bug tracker items	Issue is a synonym for all kinds of problems that need action on the resulting system.
wall	scrum wall, scrum board, information radiator, kanban board	Accumulation for all types of display, for information on process and progress tracking and interacting.
coding standard	programming guidelines, coding style, coding standards, coding guidelines	Consistent name for all approaches to standardize the layout of source code.
iteration backlog	sprint backlog	Introduced to generalize the backlog used in scrum.

Table 3.4: Synonyms introduced for homogenization

Research question RQ3a asks for common artifacts. This will be addressed in the following paragraph. Lots of the artifacts are only named once in a single paper in a specific context or even only in the context of a solution proposal. These are definitely not classified as common artifacts. The set of common artifacts is for sure a subset of the 162 artifact that is handier. To find common artifacts first of all a definition for common artifacts is needed. This study defines an artifact as common by the number of papers mentioning a single artifact and the research type facets of these papers. Artifacts that are mentioned in at least three studies are considered as candidates for common to agile development in this case. To validate these candidates and assure a certain quality the finding the research type facet is used. A valid common artifact needs to be present in at least three studies with research type facets including multiple validation researches or an experience paper or an evaluation research. Figure 3.10 shows 19 artifacts out of the 162 meeting the above criterion. The main artifacts identified by the cloud in Figure 3.9

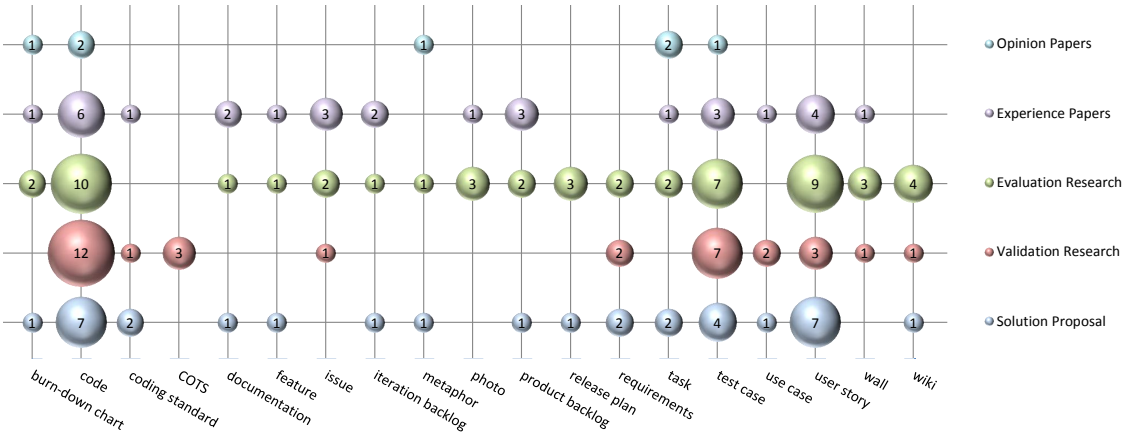


Figure 3.10: Valid common agile artifacts and research type facets of originating papers

can also be found in the map in Figure 3.10. Code, user story and test case are the most common artifacts with 32, 20 and 19 papers referencing them. Backlog was the fourth candidate. But it is split into two artifacts the iteration backlog and the production backlog and therefore is no more that prominent as thought in the beginning by evaluating only the cloud. Each other artifact is referenced by 3 to 7 papers naming it. The resulting artifacts will be described in section 4.1.

It was impossible to find answers to the research question related to controlling (RQ3b) since only five papers were classified with the inclusion criterion **xC** (see Table 3.2). These five papers did not reveal any knowledge apart from the fact that artifacts are used for controlling of agile development.

3.5 Relations between common artifacts - RQ4

This section tries to uncover the relationships between common artifacts and how they can be classified. These relationships and classifications shall later help to create an artifact model. Artifacts can relate to each other in different ways.

One way is the affiliation to a certain process or practice forming an artifact model. As the resulting artifacts are associated with different processes or practices there is no guarantee that they can form any connected artifact model. Also there is no proof that the set of resulting artifacts is complete.

Another relation between artifacts may be a dependency where an artifact is created as result by processing one or more other artifacts as explained by Gonzalez-Perez and Henderson-Sellers in their work product pool approach [39].

A third relationship is a containment relationship. This relationship is quite easy to explain with an example. As photo is one of the identified artifacts it is easy to imagine to take a picture of any other physical artifact or a collection of artifacts [87]. Thus a photo is a container for any other artifact. In the same way a backlog contains a set of user stories, tasks or requirements as described by Sutherland et al. [86, 52] or Beck [9].

Refinement and generalization is one more relationship between artifacts. For example the iteration backlog refines the entries of the product backlog. Also test cases are having the same relation where one test *T1* on a general level can be refined by several tests *t1 - tn* checking for parts of *T1*.

Interaction also can be seen as a relation between artifacts. Usually interaction is described in activities that follow a certain process or practice. For example the implemen-

tation of a specific requirement may result in an extension of the source code and the reconfirmation that no test case was broken by the extension. Thus requirement, code and test case are interacting within the activity of implementation.

By applying the previous types of relations to the nineteen common artifacts a matrix showing all the relations between the artifacts can be created. Utilizing this matrix it can be checked whether all found artifacts can contribute to one common artifact model. All

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19
A1	X				✓	✓	✓	✓			✓		✓	✓		✓	✓	✓	
A2		X	✓	✓	✓	✓	✓						✓	✓	✓	✓	✓		
A3		✓	X		✓														
A4		✓		X	✓														
A5	✓	✓	✓	✓	X	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
A6	✓	✓			✓	X	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
A7	✓	✓			✓	✓	X	✓		✓	✓		✓	✓	✓	✓	✓	✓	
A8	✓				✓	✓	✓	X			✓	✓	✓	✓	✓	✓	✓		
A9					✓	✓			X				✓			✓	✓		
A10					✓	✓	✓			X			✓	✓		✓	✓	✓	
A11	✓				✓	✓	✓	✓			X	✓	✓	✓		✓	✓	✓	
A12					✓	✓		✓			✓	X	✓	✓		✓	✓	✓	
A13	✓	✓			✓	✓	✓	✓	✓	✓	✓	✓	X	✓	✓	✓	✓	✓	✓
A14	✓	✓			✓	✓	✓	✓		✓	✓	✓	✓	X	✓	✓	✓	✓	
A15		✓			✓	✓	✓						✓	✓	X	✓	✓		✓
A16	✓	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	X	✓	✓	✓
A17	✓	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	X	✓	✓
A18	✓				✓	✓	✓	✓		✓	✓	✓	✓	✓		✓	✓	X	
A19					✓	✓							✓		✓	✓	✓		X

Table 3.5: Relationships of artifacts

artifacts within the matrix have a relation to others. A graph created by the artifacts as nodes and relations as edges is in the sense of graph theory connected. Thus it is possible to create a syntactically correct model of the artifacts including all nineteen findings. If there is a semantically correct model and how to create it will be discussed in the next chapter.

3.5.1 Classification of artifacts

During the data collection all classification criteria for artifacts were collected and noted in the inclusion list introduced in Table 2.5. 32 studies were marked with the inclusion criteria **xG** that displays contribution to the classification of artifacts. These classification criteria shall now be used to describe artifacts in a more precise way and to cluster them in groups. To aggregate all criteria and find those which are common again a tag cloud has been generated. The cloud was created with the help of the "Classification" column of the inclusion list and can be seen in Figure 3.11. The cloud revealed several classifi-

Figure 3.11: Tag cloud for categories and attributes

cation criteria in which artifacts can be grouped. The most used criteria are: code, test, design, requirements, physical, documentation, management, production, intermediate, final, internal and digital. The next paragraphs will concentrate on these classification criteria.

Source code artifacts are all artifacts that contain written source code and some other artifacts. This category includes production code as well as the code for tests or libraries. Also models that can be executed to generate code are classified as code artifacts. Further all forms of markup code e. g. HTML or XML contribute to the code artifacts. Resources for properties or language files also can be considered as they are also translated into the code.

A classification as **test** artifacts describes the ones related to tests. They can be test code, test case descriptions or produced and expected test results. But also test systems like xUnit in form of COTS should be classified as test artifact.

Design artifacts are not generally defined. There is a huge tag "design" in the cloud but it is created from different interpretations of the classification as design artifact. A part of the contributions to it is resulting from the artifacts of the AUP that are associated with the analysis and design discipline. This also holds for the Analysis tag in the cloud (see Making Rup Agile [46]). Another contribution to the design tag are papers concerned about UI design which see UI designs as design artifacts [74]. This is contrasted by Neill who states that “code is the primary design artifact, not a series of class and sequence diagrams, or dataflow models.” [72] and thus classifies code as design artifact. According to Cohn, Sim and Lee summarizing Bertelsen design artifacts are boundary objects that “move between construction and representation” [22]. They explain with the help of a

user story being between the envisioned software system to construct and the code as a representation [22]. Dulipovici and Robillard refer for design artifacts in XP to mental models which are shared among developers. These shared models form the final design model [26]. Owing to these manifold definitions and interpretations design seems not to be the best way to classify artifacts, as it is real ambiguous. Hence it is not used further in this thesis for a classification of artifacts.

Requirements specification is a classification for all artifacts that contain information about what system to develop and how it should behave. Beck names functional tests as the primary requirements specification artifact [9]. They are also supported by Ghanam [35]. Also stories are requirements specifications [20, 35]. Use cases and features are also described as requirements specification by Hirsch [46].

Another classification described in the included papers is **physical vs. mental** artifacts. Abdullah, Sharp and Honiden describe physical artifacts as those which "focus on capturing and displaying progress information rather than requirements issues such as problems, goals or functionality"[1]. They name story cards and the wall as examples which are confirmed in [69]. Abdullah, Sharp and Honiden contrast physical to mental artifacts e. g. requirements of the product which "reside in the social context and are sustained through communication and collaboration activities". The terms physical and mental are also defined by Dulipovici and Robillard [26] in the following way:

> We defined the term "physical artifact" - a piece of information produced by completing an active, creative activity, as Write, Draw, Code, Code & Test, Test, Integrate & Test, Technical Administration. A physical artifact can be a text document, a model - mainly represented by diagrams, or a piece of code.
>
> We also defined the term "mental artifact" - a piece of information produced by completing a reflexive or an interactive activity, as Read, Think, Discuss, Browse/Search, Training, Inspect/Review. A mental artifact is a model that "runs" in the human mind.

This definition is quite close to the use of Abdullah and Sharp. Thus we can distinguish between an artifact that has a physical representation and those who are only a product of the social context of the software engineering process.

Documentation is a way to classify artifacts which is often mixed up with the artifact documentation itself thus in this study it is treated as the same concept. A common meaning is, source code is the only documentation of the system [80]. But there are different interpretations as well. Cohn, Sim and Lee found, that in agile development the role of documentation must be seen differently than in traditional software processes. Their study revealed that documentation does not follow formal document specifications but "it is whatever you want" [22]. This enables the use of any artifact in a process as documentation. This study follows this approach and classifies all artifacts as documentation.

Project **management** artifacts are those referring the managerial aspect of the process. Dulipovici and Robillard describe them for XP by "[...] short releases, which are planed based on the estimation and the priority of the user stories." [26] Based on this the artifacts: release plan, story estimates which reflect the specification of a user story and the backlog as a root for the priority of stories. Hirsch lists Software Development Plan, Iteration Plans and Iteration Assessments [46] as artifacts of the project management discipline. This discipline contains the project planning and monitoring thus all related artifacts even if not listed by Hirsch will be classified as management artifacts. The Software Development Plan implements the same concept as a release plan and the Iteration Plan provides the same functionality as the iteration backlog in this way the two sources support each other.

Production process can be used to classify artifacts whether if they are inside or outside

the software process. Following Cohn and her colleagues, artifacts that are necessary to enact a software process are classified as inside production process artifacts. Other artifacts that exist in the context of a process but are not used for enactments are outside, e. g. additional documentation of stories or tests [22]. Hence artifacts can move from within to outside of the production process or vice versa through an enactment, the classification is dynamic.

The classification as **intermediate** or **final** artifact is described by [39, 67]. Gonzalez-Perez and Henderson-Sellers define intermediate artifacts as such artifacts that need to be created to produce the final artifact but are not final by themselves. They further classify internal artifacts as those intermediate artifacts being provided from within the process. Final artifacts are defined as the artifacts that are delivered to the customer, in the simplest case the software system. Mikio et al. does not follow this definition of intermediate and final in the exact same manner. According to Mikio et al. final artifacts are those validated and accepted. [67] Intermediate products thus can be the same ones as final ones just before validation. Ceravolo et al. who use the words internal product and deliverable in their paper about the XP ontology also distinguish weather an artifact is only needed during the production of software or if it is part of the final delivery [19].

One more classification for artifacts is **digital** or **analog**. This is simply the representation in the process where a user story can be on an index card or on a virtual index card in a software system. The concept of digital artifacts is addressed by Kowark et al. [58] and Brown, Lindgaard and Biddle [15]. They named emails, wiki pages, bug tracker items, source code and digital images as digital artifacts but there are definitely more.

This classification also helps to find relations between the artifacts as all artifacts classified by the same criteria are somehow related and often interact with each other.

4 Resulting Artifact Model

This chapter shows the common artifacts that result from this thesis in detail and how to combine them to an artifact model for agile development. The first section describes the common artifacts found in this study and lists the sources where they result from. This is followed by a section describing the notation used to create the resulting artifact model. The next section describes the elements that can be found in the model. Concluding the last sections highlights the applicability and how the artifact model relates to the Agile Manifesto.

Overview

4.1 Overview of Artifacts

This section describes the artifacts found in Section 3.4. All nineteen findings are listed in alphabetical order in Table 4.1. The first column of this table contains the artifact number. It is followed by the name and description of the artifact. The last column contains references to the paper where the artifact was found within the qualified papers. As there is no generally accepted definition for all artifacts only a description is provided. But in some cases where available, an official definition is quoted. The source for the description is very often from complementary literature and could not be found in the qualified studies of this study. This is because most of the papers only name artifacts but do not describe them in details as visible in Table 3.2 where only eight of the included papers are classified with the inclusion criteria xD.

Abbr.	Aartifact	Description	Sources
A1	burn-down chart	"document that records project status, usually showing tasks completed against total number of tasks." ISO/IEC/IEEE 26515:2012 [49]	[98, 62, 41, 69, 11]
A2	Code	Source code is an artifact containing computer interpretable instructions implementing requirements of software to produce. The source code artifact can also be a model from which code can be generated. The compiled code forms the final software system	[37, 53, 28, 26, 71, 7, 72, 12, 30, 99, 38, 8, 5, 82, 92, 89, 68, 14, 77, 80, 4, 96, 60, 35, 51, 6, 40, 93, 24, 85, 84, 34]
A3	Coding standard	The coding standard is a set of rules to be followed during source code creation. These rules mainly regulate the appearance of source code (e. g. indentation, spaces or tabs) but can also treat the structuring of source code (e. g. ordering of methods and member within a class). Applying these rules is supporting collaboration and reuse.	[19, 99, 46, 77]
A4	COTS	A "Commercial off the Shelf" (COTS) software component is a piece of software that is commercially available to the general public without necessary customization [91]. Usually COTS software is reused in multiple identical copies by different institutions. It is usually a part of a software system to integrate requirements in a cheap manner.	[39, 85, 84]
A5	Documentation	"Any written or pictorial information describing, defining, specifying, reporting, or certifying activities, requirements, procedures, or results." IEEE 829-2008 [48]	[80, 87, 86, 78]

To be continued . . .

Abbr.	Aartifact	Description	Sources
A6	Feature	"functional or non-functional distinguishing characteristic of a system, usually an enhancement to an existing system." ISO/IEC/IEEE 26515:2012 [49]	[88, 73, 35]
A7	Issue	Issues are special tasks or requirements that usually deal with the removing of an error. Or an issue is a kind of refactoring or modification to improve the system.	[83, 17, 58, 46]
A8	Iteration backlog	"A list of tasks that defines a Team's work for a Sprint. The list emerges during the Sprint. Each task identifies those responsible for doing the work and the estimated amount of work remaining on the task on any given day during the Sprint." Sutherland [52]	[62, 59, 69, 11]
A9	Metaphor	"A story that everyone - customers, programmers and managers - can tell about how the system works" Beck [9]	[26, 57, 72]
A10	Photo	A picture of any other artifact taken mostly for documentation reasons.	[42, 69, 87]
A11	Product backlog	"The Backlog is a prioritized list. [...] Backlog is the work to be performed on a product. Completion of the work will transform the product from its current form into its vision." see Sutherland [52, Apendix I]	[98, 62, 59, 90, 69, 11]
A12	Release plan	"A release plan is a high-level plan that covers a period longer than an iteration." Cohn [21]	[26, 19, 69, 87]
A13	Requirement	"A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents." IEEE 610.12-1990 [47]	[28, 77, 39, 87, 60, 93]
A14	Task	"The smallest unit of work subject to management accountability. A task is a well-defined work assignment for one or more project members." IEEE 829-2008 [48]	[98, 19, 41, 88, 70, 73, 63]
A15	Test case	"A set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement." IEEE 829-2008 [48]	[37, 53, 26, 7, 57, 19, 12, 99, 8, 92, 31, 46, 77, 87, 60, 35, 32, 100, 78]
A16	Use case	"description of the behavioural requirements of a system and its interaction with a user." ISO/IEC/IEEE 26515:2012 [49]	[29, 31, 68, 39]

To be continued ...

Abbr.	Aartifact	Description	Sources
A17	User story	"simple narrative illustrating the user goals that a software function will satisfy." ISO/IEC/IEEE 26515:2012 [49]	[37, 98, 29, 53, 28, 26, 90, 57, 19, 97, 69, 1, 15, 82, 31, 22, 73, 6, 32, 33]
A18	Wall	"The Scrum Board[1] has emerged as a best practice for a team to manage their own tasks. Teams meet in front of the Board which has multiple columns. The first column has User Stories from the Product Backlog [...]. At the start of the Sprint, [...] User Story are in the left column as small cards. Each day developers move tasks to an "In Progress" column, then to a "Validation" column, then to a "Done" column. [...] and the Burndown Chart can easily be calculated and posted to the board" Kniberg[56]	[98, 69, 11, 1, 2]
A19	Wiki	"A wiki that helps on quickly weaving different kinds of contents into a single heterogeneous document, whilst preserving its semantic consistency. The fundamental goal of a wiki is to reduce the development-documentation gap by making documentation more convenient and attractive to developers." Riehle [79]	[19, 58, 22, 35, 32]

Table 4.1: Resulting common artifacts

4.2 Notation and Application

To model the artifacts that result from this study a UML[2] notation is used. A class diagram serves to model the artifact model. Artifacts are represented as classes that are shown with their relations. Attributes of artifacts are given rarely because they vary a lot in different implementations. The relations provided by UML seem to represent to a big part the relations introduced in Section 3.5 and thus can easily be used.

Abstract classes in UML will be used to represent artifacts that form a generalization for other artifacts. Packages in the artifact model represent classification categories as described in Section 3.5.1. Another notation for categories is the coloring which groups the artifacts together that are in different packages.

Other not described cases follow the UML conventions.

4.3 Model Description

After combining and connecting the artifacts the resulting artifact model is shown in Figure 4.1.

1 The term "Scrum Board" in this definition is used as a synonym for the artifact "Wall". See Table 3.4.
2 Unified Modeling Language see www.uml.org

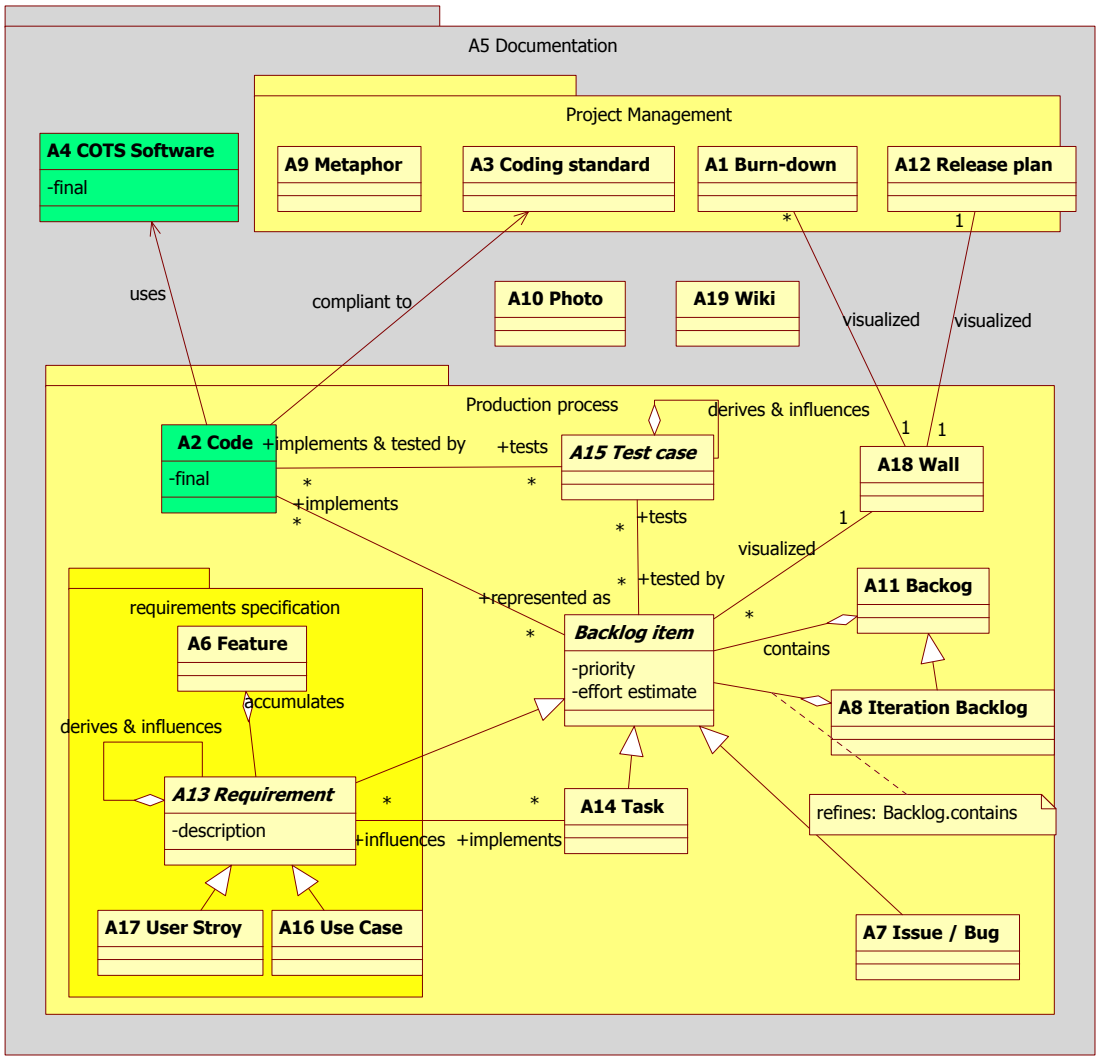


Figure 4.1: Artifact model constructed of resulting artifacts

Documentation. Within the model there is a huge package called documentation containing all other artifacts. This reflects the fact that in agile development every artifact can and should be used as documentation. This complies with the value of "Working software over comprehensive documentation" [10] in the Agile Manifesto where comprehensive documentation isn't valued that much. Thus there are no huge documentation documents. But all artifacts needed during software development can be used as documentation. This follows the classification described in Section 3.5.1.

Final Artifacts. The other value in the above citation (working software) is represented in the model by artifacts classified as final. These final artifacts build the delivered software system. In this model it is only the source code and COTS software, which is a quite slim set of artifacts that need to be delivered. They are illustrated in green in the model.

Requirement Specification. In order to build final artifacts the requirement specification artifacts need to be implemented. The package is composed of Feature, Requirement, User story and Use case in the model. Functional tests which were found earlier as requirements specifications were moved out to be able to generalize all tests. The structure

and hierarchy of requirement specification artifacts was not derivable from the included papers because of ambiguity. Thus the "concept model of the requirements specification" provided in [64] was adapted.

Test Case. The test case is an abstract class in the model. Such an abstract class is necessary because a lot of different test case implementations were joined with the synonyms table (see Table 3.4). The Test case class provides the baseline for any test implementation. Usually the hierarchy created through requirements specification artifacts is mapped to the test artifacts which are all derived from the abstract class Test case.

Production Process. To transform the requirements to the final product the following in-production process artifacts are suggested. But as defined by Cohn et al. these artifacts can vary depending on the enactment of the process and thus the artifacts presented here are only a hint for what would be applicable [21]. All backlogs and their backlog items should be seen as in-process artifacts. As well the wall artifact is used to enact the process as it can be seen as a simplified physical representation of a backlog containing items in form of index cards. Also parts of the test cases shall be considered as they drive the production of code which is the last artifact in this classification.

Project Management. The last group of artifacts is the one containing project management artifacts. Release plan is definitely a way for the project management to plan for the future progress. Usually the release plan is using features to plan the content for further releases but also could contain issues. Also the Burn-down chart is a management instrument, which allows depicting the current results of the development activities and reacting on them. The metaphor is also placed in the management package as it reflects a vision of the product or the development process and should be used to guide the development into a certain direction. Coding standard is the last part of the project management artifacts as it is used to enable collaboration on code and thus is an instrument of the project management.

Backlog Item. There is an abstract class that cannot be found within the common artifacts. It is the "backlog item" which is used to abstract different types of content for a backlog. It was introduced to reduce the complexity of a model that would result without this abstraction step. As two different backlogs which contain items on a different degree of detail were identified it is necessary to provide an option to put the items into the backlogs. Usually the iteration backlog contains backlog items of the type task or issue but also can contain small requirements that need no further splitting. The product backlog usually contains requirements or features but also can contain issues that are not relevant for the current iteration or tasks that result from earlier not implemented planning.

4.4 Interpretation

The model is built from the 19 common artifacts that were found through RQ3 and the relationships found in RQ4. These artifacts and relations were found through an empirical method. Thus the model is empirically sound.

But as with all agile processes and practices the artifact model is not a silver bullet. It may be adapted according to the needs of every special case. In some cases the provided model is too complex. Then artifacts not classified in the production process package should be considered as candidates to drop. In cases where other artifacts are needed or

prescribed not available in the model they can be integrated and classified where ever they suit best.

The core of the model can be seen as the triangle of code, test cases and backlog items which support each other forming a solid structure. This triangular structure tries to reduce faults in the software because the requirement is on one hand implemented as code and on the other hand as test cases. Only if the code is doing what is described by the test cases it is considered as implemented correctly. This relates to the value of working software of the Manifesto for agile software development.

The value of responding to change is represented in the model by the fact that any customer wish can be fulfilled just by adding a backlog item with an according priority and the agile development will implement it as soon as possible. No matter if it was a new requirement, new feature, a bug fix or any other task changing anything in the software system.

The value of customer collaboration is addressed in the model by the fact that a customer can provide requirements in any form he wants. The absence of any artifact representing a contract also indicates that contracts are no priority in agile development and the artifact model.

The last mentioned but first value of the Agile Manifesto is: Individuals and interactions. Abdullah et al. generalize their findings that physical artifacts and individuals are fundamental stimuli to start, sustain and end interactions. Further they identified eight patterns that stimulate interactions between developers and six out of them were initialized by artifacts [1]. Thus every artifact in a model can foster the interaction between individuals.

By addressing all four values of the Agile Manifesto the model itself can be considered as agile.

5 Conclusion

After investigating in the artifact-orientation of agile development methods through a literature study this Chapter summarizes the outcomes and provides an idea for further research activities.

Overview

5.1 Summary of Outcomes

This section summarizes the outcomes reported in this paper. The systematic case study reveals that artifact-orientation is not very present in agile development. But enough research has been done to perform a secondary study. The results of this secondary study show that the used research methods in primary studies are maturing. Since the last four years research can be considered as mature due to the presence of numerous Evaluation Research publications. The number of Solution Proposals indicates that there is still a lot of innovation in research. Lots of the studies are performed in close cooperation with or by the industry. Scrum and XP are the agile processes in the focus of research on artifact-orientation. But it can be assumed that the trend to cover other processes found in general research on agile development will be followed soon by artifact-oriented research.

The analysis of 76 qualified publications lead to 19 artifacts that are commonly used in agile development across processes and practices. The found artifacts have a high cohesion which means there are lots of relationships between them. These relationships have been empirically extracted and were analyzed for their applicability for an artifact model. Finally an artifact model was created. This model is applicable to any agile development methodology. The model can easily be adjusted according to the needs. Categories help to identify the right spot for modification.

19 common artifacts and the relationships between them found utilizing empirical methods were used to create the artifact model. Thus the extracted artifact model is empirically sound.

5.2 Future work

The thesis at hand provides an artifact model based on a well-chosen subset of agile processes and practices. But it is only a first step towards a common artifact model for all agile development methods.

To provide an advanced model it can be investigated if the inclusion of further processes and practices leads to additional artifacts and relationships. Also the processes and practices so far underrepresented in the qualified publications (see Table 3.3) need further investigation to find artifacts based on other sources. For example not Feature artifact was found in the context of FDD. This could lead to a refinement and extension of the artifact model.

The collected data is based literature and therefore might not represent the point of view from an industry perspective. A further study collecting artifact data from other sources e. g. an industry survey could be used to verify the findings.

The results of this study are generated through an empirical approach based on a systematic evaluation of literature. An empirical evaluation of the applicability of the model in praxis is a recommended step towards a model for agile artifacts accepted by industry and academia.

Bibliography

[1] ABDULLAH, N. N. B., SHARP, H., AND HONIDEN, S. Communication in context: A stimulus-response account of agile team interactions. In *Agile Processes in Software Engineering and Extreme Programming*, W. Aalst, J. Mylopoulos, N. M. Sadeh, M. J. Shaw, C. Szyperski, A. Sillitti, A. Martin, X. Wang, and E. Whitworth, Eds., vol. 48 of *Lecture Notes in Business Information Processing*. Springer Berlin Heidelberg, 2010, pp. 166–171.

[2] ABLETT, R., SHARLIN, E., MAURER, F., DENZINGER, J., AND SCHOCK, C. Buildbot: Robotic monitoring of agile software development teams. In *RO-MAN 2007 - The 16th IEEE International Symposium on Robot and Human Interactive Communication* (2007), IEEE, pp. 931–936.

[3] ABRANTES, J., AND TRAVASSOS, G. Common agile practices in software processes. In *Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on* (sept. 2011), pp. 355 –358.

[4] AL-ZOABI, Z. Introducing discipline to xp: Applying prince2 on xp projects. In *2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications* (2008), IEEE, pp. 1–7.

[5] ALVES, T. L. Categories of source code in industrial systems. In *2011 International Symposium on Empirical Software Engineering and Measurement* (2011), IEEE, pp. 335–338.

[6] ALYAHYA, S., IVINS, W. K., AND GRAY, W. Co-ordination support for managing progress of distributed agile projects. In *2011 IEEE Sixth International Conference on Global Software Engineering Workshop* (2011), IEEE, pp. 31–34.

[7] ANDREA, J. Envisioning next-generation functional testing tools. *IEEE Software 24*, 3 (2007), 58–66.

[8] BAUMEISTER, H. Combining formal specifications with test driven development. In *Extreme Programming and Agile Methods - XP/Agile Universe 2004*, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, C. Zannier, H. Erdogmus, and L. Lindstrom, Eds., vol. 3134 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2004, pp. 1–12.

[9] BECK, K. *Extreme programming eXplained: Embrace change*. Addison-Wesley, Reading and MA, 2000.

[10] BECK, K., BEEDLE, M., VAN BENNEKUM, A., COCKBURN, A., CUNNINGHAM, W., FOWLER, M., GRENNING, J., HIGHSMITH, J., HUNT, A., JEFFRIES, R., KERN, J., MARICK, B., MARTIN, R. C., MELLOR, S., SCHWABER, K., SUTHERLAND, J., AND THOMAS, D. Manifesto for agile software development, 2001.

[11] BLANKENSHIP, J., BUSSA, M., AND MILLETT, S. Managing agile projects with scrum. In *Pro Agile .NET Development with Scrum*, J. Blankenship, M. Bussa, and S. Millett, Eds. Apress, 2011, pp. 13–27.

[12] BOGACKI, B., AND WALTER, B. Evaluation of test code quality with aspect-oriented mutations. In *Extreme Programming and Agile Processes in Software Engineering*, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos,

D. Tygar, M. Y. Vardi, G. Weikum, P. Abrahamsson, M. Marchesi, and G. Succi, Eds., vol. 4044 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2006, pp. 202–204.

[13] BOOCH, G. The economics of architecture-first. *IEEE Software 24*, 5 (2007), 18–20.

[14] BREWER, J., AND LORENZ, L. Using uml and agile development methodologies to teach object-oriented analysis & design tools and techniques. In *Proceeding of the 4th conference on Information technology education - CITC '03* (2003), ACM Press, p. 54.

[15] BROWN, J. M., LINDGAARD, G., AND BIDDLE, R. Collaborative events and shared artefacts: Agile interaction designers and developers working toward common aims. In *2011 AGILE Conference* (2011), IEEE, pp. 87–96.

[16] BUDGEN, D., KITCHENHAM, B. A., CHARTERS, S. M., TURNER, M., BRERETON, P., AND LINKMAN, S. G. Presenting software engineering results using structured abstracts: a randomised experiment. *Empirical Software Engineering 13*, 4 (2008), 435–468.

[17] CANFORA, G., CONCAS, G., MARCHESI, M., TEMPERO, E., ZHANG, H., MENEELY, A., CORCORAN, M., AND WILLIAMS, L. Improving developer activity metrics with issue tracking annotations. In *Proceedings of the 2010 ICSE Workshop on Emerging Trends in Software Metrics - WETSoM '10* (2010), ACM Press, pp. 75–80.

[18] CAPILUPPI, A., FERNANDEZ-RAMIL, J., HIGMAN, J., SHARP, H., AND SMITH, N. An empirical study of the evolution of an agile-developed software system. In *29th International Conference on Software Engineering (ICSE'07)* (2007), IEEE, pp. 511–518.

[19] CERAVOLO, P., DAMIANI, E., MARCHESI, M., PINNA, S., AND ZAVATARELLI, F. A ontology-based process modelling for xp. In *Tenth Asia-Pacific Software Engineering Conference, 2003* (2003), IEEE, pp. 236–242.

[20] CHOUDHARI, J., AND SUMAN, U. Story points based effort estimation model for software maintenance. *Procedia Technology 4* (2012), 761–765.

[21] COHN, M. *Agile estimating and planning*. Prentice Hall Professional Technical Reference, Upper Saddle River and N.J, 2006.

[22] COHN, M. L., SIM, S. E., AND LEE, C. P. What counts as software process? negotiating the boundary of software work through artifacts and conversation. *Computer Supported Cooperative Work (CSCW) 18*, 5-6 (2009), 401–443.

[23] DINGSØYR, T., NERUR, S., BALIJEPALLY, V., AND MOE, N. B. A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software 85*, 6 (2012), 1213–1221.

[24] DOWNS, J., HOSKING, J., AND PLIMMER, B. Status communication in agile software teams: A case study. In *2010 Fifth International Conference on Software Engineering Advances* (2010), IEEE, pp. 82–87.

[25] DROBKA, J., NOFTZ, D., AND REKHA RAGHU. Piloting xp on four mission-critical projects. *IEEE Software 21*, 6 (2004), 70–75.

[26] DULIPOVICI, M., AND ROBILLARD, P. Cognitive aspects in a project-based course in software engineering. In *Information Technology Based Proceedings of the FIfth International Conference onHigher Education and Training, 2004. ITHET 2004* (2004), IEEE, pp. 353–359.

[27] DYBÅ, T., AND DINGSØYR, T. Empirical studies of agile software development: A systematic review. *Information and Software Technology 50*, 9-10 (2008), 833–859.

[28] FARID, W. M., AND MITROPOULOS, F. J. Normatic: A visual tool for modeling nonfunctional requirements in agile processes. In *2012 Proceedings of IEEE Southeastcon* (2012), IEEE, pp. 1–8.

[29] FARID, W. M., AND MITROPOULOS, F. J. Novel lightweight engineering artifacts for modeling non-functional requirements in agile processes. In *2012 Proceedings of IEEE Southeastcon* (2012), IEEE, pp. 1–7.

[30] FAVELA, J., NATSU, H., PÉREZ, C., ROBLES, O., MORÁN, A. L., ROMERO, R., MARTÍNEZ-ENRÍQUEZ, A. M., AND DECOUCHANT, D. Empirical evaluation of collaborative support for distributed pair programming. In *Groupware: Design, Implementation, and Use*, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, G.-J. Vreede, L. A. Guerrero, and G. Marín Raventós, Eds., vol. 3198 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2004, pp. 215–222.

[31] GALLARDO-VALENCIA, R. E., OLIVERA, V., AND SIM, S. E. Are use cases beneficial for developers using agile requirements? In *2007 Fifth International Workshop on Comparative Evaluation in Requirements Engineering* (2007), IEEE, pp. 11–22.

[32] GALLARDO-VALENCIA, R. E., AND SIM, S. E. Continuous and collaborative validation: A field study of requirements knowledge in agile. In *2009 Second International Workshop on Managing Requirements Knowledge* (2009), IEEE, pp. 65–74.

[33] GANIS, M., LEIP, D., GROSSMAN, F., AND BERGIN, J. Introducing agile development (xp) into a corporate webmaster environment - an experience report. In *Agile Development Conference (ADC'05)* (2005), IEEE Comput. Soc, pp. 145–152.

[34] GHANAM, Y., ANDREYCHUK, D., AND MAURER, F. Reactive variability management in agile software development. In *2010 Agile Conference* (2010), IEEE, pp. 27–34.

[35] GHANAM, Y., AND MAURER, F. Extreme product line engineering: Managing variability and traceability via executable specifications. In *2009 Agile Conference* (2009), IEEE, pp. 41–48.

[36] GHANAM, Y., MAURER, F., ABRAHAMSSON, P., AND COOPER, K. A report on the xp workshop on agile product line engineering. *ACM SIGSOFT Software Engineering Notes 34*, 5 (2009), 25.

[37] GINI, M., ISHIDA, T., CASTELFRANCHI, C., JOHNSON, W. L., AND KNUBLAUCH, H. Extreme programming of multi-agent systems. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems part 2 - AAMAS '02* (2002), ACM Press, p. 704.

[38] GOEDICKE, M., MENZIES, T., SAEKI, M., DÖSINGER, S., MORDINYI, R., AND BIFFL, S. Communicating continuous integration servers for increasing effectiveness of automated testing. In *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering - ASE 2012* (2012), ACM Press, p. 374.

[39] GONZALEZ-PEREZ, C., AND HENDERSON-SELLERS, B. A work product pool approach to methodology specification and enactment. *Journal of Systems and Software 81*, 8 (2008), 1288–1305.

[40] GOODMAN, D., AND ELBAZ, M. It's not the pants, it's the people in the pants learnings from the gap agile transformation what worked, how we did it, and what still puzzles us. In *Agile 2008 Conference* (2008), IEEE, pp. 112–115.

[41] GREEN, E. O. Empirical management of engineering projects: A common sense approach to agility. In *Agile Manufacturing, 2007. ICAM 2007. IET International Conference on,* (2007), pp. 74–77.

[42] GREGORIO, D. D. How the business analyst supports and encourages collaboration on agile projects. In *2012 IEEE International Systems Conference SysCon 2012* (2012), IEEE, pp. 1–4.

[43] HABERL, P., SPILLNER, A., VOSSEBERG, K., AND WINTER, M. Softwaretest in der praxis: Umfrage 2011.

[44] HADDAD, H. M., DOBRZAŃSKI, Ł., AND KUŹNIARZ, L. An approach to refactoring of executable uml models. In *Proceedings of the 2006 ACM symposium on Applied computing - SAC '06* (2006), ACM Press, p. 1273.

[45] HASHMI, S. I., AND BAIK, J. Quantitative process improvement in xp using six sigma tools. In *Seventh IEEE/ACIS International Conference on Computer and Information Science (icis 2008)* (2008), IEEE, pp. 519–524.

[46] HIRSCH, M. Making rup agile. In *OOPSLA 2002 Practitioners Reports on - OOPSLA '02* (2002), ACM Press, p. 1.

[47] IEEE COMPUTER SOCIETY. Ieee standard glossary of software engineering terminology. *IEEE Std 610.12-1990* (1990), 1–84.

[48] IEEE COMPUTER SOCIETY. Ieee standard for software and system test documentation. *IEEE Std 829-2008* (2008), 1–118.

[49] IEEE COMPUTER SOCIETY. Systems and software engineering – developing user documentation in an agile environment. *ISO/IEC/IEEE 26515 First edition 2011-12-01; Corrected version 2012-03-15* (2012), 1–36.

[50] JANUS, A. Towards a common agile software development model (asdm): (asdm). *ACM SIGSOFT Software Engineering Notes 37*, 4 (2012), 1–8.

[51] JANUS, A., DUMKE, R., SCHMIETENDORF, A., AND JAGER, J. The 3c approach for agile quality assurance. In *2012 3rd International Workshop on Emerging Trends in Software Metrics (WETSoM)* (2012), IEEE, pp. 9–13.

[52] JEFF SUTHERLAND PH. D, SCHWABER, K., SCRUM, C.-C. O., SUTHERL, C. J., AND D, P. *The Scrum Papers: Nuts, Bolts, and Origins of an Agile Process*, 1.1 ed. Scrum Inc., Cambridge, 2012.

[53] JI, F., AND SEDANO, T. Comparing extreme programming and waterfall project results. In *2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T)* (2011), IEEE, pp. 482–486.

[54] KITCHENHAM, B., PEARL BRERETON, O., OWEN, S., BUTCHER, J., AND JEFFERIES, C. Length and readability of structured software engineering abstracts. *IET Software 2*, 1 (2008), 37.

[55] KITCHENHAM, B. A., AND CHARTERS, S. Guidelines for performing systematic literature reviews in software engineering, 2007.

[56] KNIBERG, H. *Scrum and xp from the trenches: How we do scrum*. C4Media Inc., 2007.

[57] KOKKONIEMI, J. K. Gathering experience knowledge from iterative software development processes. In *Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)* (2008), IEEE, p. 333.

[58] KOWARK, T., MÜLLER, J., MÜLLER, S., AND ZEIER, A. An educational testbed for the computational analysis of collaboration in early stages of software development processes. In *2011 44th Hawaii International Conference on System Sciences* (2011), IEEE, pp. 1–10.

[59] LAANTI, M. Implementing program model with agile principles in a large software development organization. In *2008 32nd Annual IEEE International Computer Software and Applications Conference* (2008), IEEE, pp. 1383–1391.

[60] LI, N., PRAPHAMONTRIPONG, U., AND OFFUTT, J. An experimental comparison of four unit test criteria: Mutation, edge-pair, all-uses and prime path coverage. In *2009 International Conference on Software Testing, Verification, and Validation Workshops* (2009), IEEE, pp. 220–229.

[61] MAALEJ, W., AND HAPPEL, H.-J. A lightweight approach for knowledge sharing in distributed software teams. In *Practical Aspects of Knowledge Management*, T. Yamaguchi, Ed., vol. 5345 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2008, pp. 14–25.

[62] MARCHESI, M., MANNARO, K., URAS, S., AND LOCCI, M. Distributed scrum in research project management. In *Agile Processes in Software Engineering and Extreme Programming*, G. Concas, E. Damiani, M. Scotto, and G. Succi, Eds., vol. 4536 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2007, pp. 240–244.

[63] MCHUGH, O., CONBOY, K., AND LANG, M. Agile practices: The impact on trust in software project teams. *IEEE Software 29*, 3 (2012), 71–76.

[64] MÉNDEZ FERNÁNDEZ, D. *Requirements Engineering: Artefact-Based Customisation*. PhD thesis, Technische Universität München, München, 2011.

[65] MÉNDEZ FERNÁNDEZ, D., AND BROY, M. Artefact orientation: Concepts and terms, 2012.

[66] MÉNDEZ FERNÁNDEZ, D., PENZENSTADLER, B., KUHRMANN, M., AND BROY, M. A meta model for artefact-orientation: Fundamentals and lessons learned in requirements engineering. In *Model Driven Engineering Languages and Systems*, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, D. C. Petriu, N. Rouquette, and Ø. Haugen, Eds., vol. 6395 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2010, pp. 183–197.

[67] MIKIO, I., MASAYUKI, O., TAKAHIRO, T., MICHIKO, O., AND SANSHIRO, S. Using a validation model to measure the agility of software development in a large software development organization. In *2009 31st International Conference on Software Engineering - Companion Volume* (2009), IEEE, pp. 91–100.

[68] MITRA, S., AND BULLINGER, T. A. Using formal software development methodologies in a real-world student project: an experience report. *J. Comput. Sci. Coll. 22*, 6 (2007), 100–108.

[69] MOE, N. B., AURUM, A., AND DYBÅ, T. Challenges of shared decision-making: A multiple case study of agile software development. *Information and Software Technology 54*, 8 (2012), 853–865.

[70] MORIEN, R. Agile management and the toyota way for software project management. In *INDIN '05. 2005 3rd IEEE International Conference on Industrial Informatics, 2005* (2005), IEEE, pp. 516–522.

[71] NATSU, H., FAVELA, J., MORAN, A., DECOUCHANT, D., AND MARTINEZ-ENRIQUEZ, A. M. Distributed pair programming on the web. In *Proceedings of the Fourth Mexican International Conference on Computer Science, 2003. ENC 2003* (2003), IEEE Comput. Soc, pp. 81–88.

[72] NEILL, C. The extreme programming bandwagon: Revolution or just revolting? *IT Professional 5*, 5 (2003), 62–64.

[73] NORD, R. L., OZKAYA, I., AND SANGWAN, R. S. Making architecture visible to improve flow management in lean software development. *IEEE Software 29*, 5 (2012), 33–39.

[74] PATERNÒ, F., LUYTEN, K., MAURER, F., BOWEN, J., AND REEVES, S. Ui-driven test-first development of interactive systems. In *Proceedings of the 3rd ACM SIGCHI symposium on Engineering interactive computing systems - EICS '11* (2011), ACM Press, p. 165.

[75] PAVLOVSKI, C. J., AND ZOU, J. Non-functional requirements in business process modeling. In *Proceedings of the fifth Asia-Pacific conference on Conceptual Modelling - Volume 79* (2008), APCCM '08, Australian Computer Society, Inc, pp. 103–112.

[76] PETERSEN, K., FELDT, R., MUJTABA, S., AND MATTSSON, M. Systematic mapping studies in software engineering. In *Proceedings of the 12th international conference on Evaluation and Assessment in Software Engineering* (2008), EASE'08, British Computer Society, pp. 68–77.

[77] RACHEV, B., SMRIKAROV, A., DĄBROWSKI, R., STENCEL, K., AND TIMOSZUK, G. Improving software quality by improving architecture management. In *Proceedings of the 13th International Conference on Computer Systems and Technologies - CompSysTech '12* (2012), ACM Press, p. 208.

[78] RAJLICH, V., AND GOSAVI, P. Incremental change in object-oriented programming. *IEEE Software 21*, 4 (2004), 62–69.

[79] RIEHLE, D., AGUIAR, A., AND DAVID, G. Wikiwiki weaving heterogeneous software artifacts. In *Proceedings of the 2005 international symposium on Wikis - WikiSym '05* (2005), ACM Press, pp. 67–74.

[80] RUBIN, E., AND RUBIN, H. Supporting agile software development through active documentation. *Requirements Engineering 16*, 2 (2011), 117–132.

[81] RUDORFER, A., STENZEL, T., AND HEROLD, G. A business case for feature-oriented requirements engineering. *IEEE Software 29*, 5 (2012), 54–59.

[82] SILVA DA SILVA, T., MARTIN, A., MAURER, F., AND SILVEIRA, M. User-centered design and agile methods: A systematic review. In *2011 AGILE Conference* (2011), IEEE, pp. 77–86.

[83] SJOBERG, D. I., JOHNSEN, A., AND SOLBERG, J. Quantifying the effect of using kanban versus scrum: A case study. *IEEE Software 29*, 5 (2012), 47–53.

[84] SUTHERLAND, J., JAKOBSEN, C. R., AND JOHNSON, K. Scrum and cmmi level 5: The magic potion for code warriors. In *AGILE 2007 (AGILE 2007)* (2007), IEEE, pp. 272–278.

[85] SUTHERLAND, J., JAKOBSEN, C. R., AND JOHNSON, K. Scrum and cmmi level 5: The magic potion for code warriors. In *Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)* (2008), IEEE, p. 466.

[86] SUTHERLAND, J., SCHOONHEIM, G., RUSTENBURG, E., AND RIJK, M. Fully distributed scrum: The secret sauce for hyperproductive offshored development teams. In *Agile 2008 Conference* (2008), IEEE, pp. 339–344.

[87] SWAMINATHAN, B., AND JAIN, K. Implementing the lean concepts of continuous improvement and flow on an agile software development project: An industrial case study. In *2012 Agile India* (2012), IEEE, pp. 10–19.

[88] SZÖKE, Á. Conceptual scheduling model and optimized release scheduling for agile environments. *Information and Software Technology 53*, 6 (2011), 574–591.

[89] TAYLOR, R. N., GALL, H., MEDVIDOVIĆ, N., CHRISTENSEN, H. B., AND HANSEN, K. M. Towards architectural information in implementation. In *Proceeding of the 33rd international conference on Software engineering - ICSE '11* (2011), ACM Press, p. 928.

[90] TENGSHE, A., AND NOBLE, S. Establishing the agile pmo: Managing variability across projects and portfolios. In *AGILE 2007 (AGILE 2007)* (2007), IEEE, pp. 188–193.

[91] UNITED STATES. Federal acquisition regulation [electronic resource], 2005.

[92] VAN ROMPAEY, B., AND DEMEYER, S. Establishing traceability links between unit test cases and units under test. In *2009 13th European Conference on Software Maintenance and Reengineering* (2009), IEEE, pp. 209–218.

[93] VANDERLEEST, S. H., AND BUTER, A. Escape the waterfall: Agile for aerospace. In *2009 IEEE/AIAA 28th Digital Avionics Systems Conference* (2009), IEEE, pp. 6.D.3–1–6.D.3–16.

[94] VERSIONONE. State of agile survey: The state of agile software development: 6th annual 2011.

[95] WIERINGA, R., MAIDEN, N., MEAD, N., AND COLETTE, R. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. *Requirements Engineering 11*, 1 (2005), 102–107.

[96] WILKERSON, J. W., NUNAMAKER, J. F., AND MERCER, R. Comparing the defect reduction benefits of code inspection and test-driven development. *IEEE Transactions on Software Engineering 38*, 3 (2012), 547–560.

[97] YAP, M. Value based extreme programming. In *AGILE 2006 (AGILE'06)* (2006), IEEE, pp. 175–184.

[98] ZACHRY, M., SPINUZZI, C., MCNELY, B. J., GESTWICKI, P., BURKE, A., AND GELMS, B. Articulating everyday actions. In *Proceedings of the 30th ACM international conference on Design of communication - SIGDOC '12* (2012), ACM Press, p. 95.

[99] ZETTEL, J., MAURER, F., MÜNCH, J., AND WONG, L. Lipe:a lightweight process for e-business startup companies based on extreme programming. In *Product Focused Software Process Improvement*, F. Bomarius and S. Komi-Sirviö, Eds., vol. 2188 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2001, pp. 255–270.

[100] ZHANG, G., SHEN, L., PENG, X., XING, Z., AND ZHAO, W. Incremental and iterative reengineering towards software product line: An industrial case study. In *2011 27th IEEE International Conference on Software Maintenance (ICSM)* (2011), IEEE, pp. 418–427.