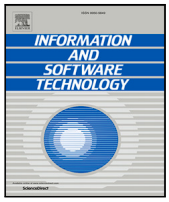
Information and Software Technology 142 (2022) 106733

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof



A mapping study on documentation in Continuous Software Development

Theo Theunissen [a,*], Uwe van Heesch [a,b], Paris Avgeriou [b]

[a] HAN University of Applied Sciences, The Netherlands
[b] University of Groningen, The Netherlands

ARTICLE INFO

Keywords:
Systematic mapping studies
Systematic reviews
Continuous Software Development
Lean
Agile
DevOps
Documentation

ABSTRACT

Context: With an increase in Agile, Lean, and DevOps software methodologies over the last years (collectively referred to as Continuous Software Development (CSD)), we have observed that documentation is often poor.
Objective: This work aims at collecting studies on documentation challenges, documentation practices, and tools that can support documentation in CSD.
Method: A systematic mapping study was conducted to identify and analyze research on documentation in CSD, covering publications between 2001 and 2019.
Results: A total of 63 studies were selected. We found 40 studies related to documentation practices and challenges, and 23 studies related to tools used in CSD. The challenges include: informal documentation is hard to understand, documentation is considered as waste, productivity is measured by working software only, documentation is out-of-sync with the software and there is a short-term focus. The practices include: non-written and informal communication, the usage of development artifacts for documentation, and the use of architecture frameworks. We also made an inventory of numerous tools that can be used for documentation purposes in CSD. Overall, we recommend the usage of executable documentation, modern tools and technologies to retrieve information and transform it into documentation, and the practice of minimal documentation upfront combined with detailed design for knowledge transfer afterwards.
Conclusion: It is of paramount importance to increase the quantity and quality of documentation in CSD. While this remains challenging, practitioners will benefit from applying the identified practices and tools in order to mitigate the stated challenges.

1. Introduction

In recent years, we have seen an increase in the adoption of Lean and Agile software development, as well as DevOps. In our previous work [387,S389,S402], we have introduced the term *Continuous Software Development* (CSD) as an umbrella term to collectively refer to such development processes and other processes that share the following characteristics:

(a) it covers the values, principles and practices from Agile [48], Lean [317] and DevOps.
(b) it embraces activities from the whole life cycle of a software product, from concept to end-of-life. In addition to Agile and Lean software development, it includes maintenance activities. In addition to DevOps, it includes continuous architecting activities [41].
(c) it considers the continuously changing state of the software product and progress, such as progressive insights (e.g. regarding process, design, implementation), changes in contextual factors, new features, bug fixes, or other unforeseen factors.
(d) it distributes information about software development across multiple tools, because of demands for fast time-to-market, as well as the need for a software development ecosystem for automated tests, deployment, and monitoring. Thus, no central repository for all information is available.

One of the main challenges in CSD is that documentation is poor [48,317,194]. This challenge hinders knowledge transfer [S58], has a bad impact on maintenance [S284] and introduces a steep learning curve [S377] for new team members. We elaborate further on these consequences. First, knowledge transfer is hindered when knowledge about the software product, such as decisions, bugs, context, and practices, remains implicit in the minds of developers and is only informally written in whiteboard sketches. As a result, knowledge walks literally out the door at the end of a daily stand-up or even leaves the company (many companies have high staff turn-over) [205,280,59]. Second, it is hard to act when bugs show up, new features or non-functional requirements arise. Developers are forced to make

* Corresponding author.
E-mail address: theo.theunissen@gmail.com (T. Theunissen).

https://doi.org/10.1016/j.infsof.2021.106733
Received 6 May 2020; Received in revised form 2 September 2021; Accepted 9 September 2021
Available online 9 October 2021

assumptions about decisions, interfaces, or priorities; such assumptions are often wrong [251,43,166]. Third, the system is hard to understand for the different stakeholders, including developers. Especially, when the team scales up, or team members switch to other projects, newcomers go through numerous trial-and-error attempts before they can contribute well [107,150,125,S149].

There is plenty of information in the different tools that are used, but that is mostly related to implementation, deployment and operations. The following, exclusively distinctive types of information are often lacking, incomplete, out-of-date, or of low quality [388]:

(1) *Stakeholders and their concerns*. This is key in prioritizing requirements and mitigating risks. A stakeholder is anyone who has an effect on the system or is affected by the system [96,187].
(2) *Risks*. Risks can endanger the project [141], and manifest as incomplete information, lack of information, or factors that are out of control of the development team.
(3) *Assumptions and constraints*. Both delimit the solution space, but are very often tacit or implicit [S268].
(4) *Context and environment*. This includes anything that has an effect on the system but is not included in the primary goals, such as legal[1] and environmental issues[2] [200].
(5) *Design decisions and their rationale*. The rationale typically concerns trade-offs between qualities, business factors, in-house expertise etc. [394,S400].
(6) *Design and/or architecture specifications*. Even if design specifications are created, they are typically not updated according to changes in requirements and context, and thus become out-of-sync with the actual code [101].

As a first step in addressing the problem of poor documentation in CSD, we decided to look into the current state of practice as reported in scientific literature. Specifically, we conducted a systematic mapping study on identifying the challenges of documentation in CSD as well as the practices and tools that can potentially support documentation. We selected to study these aspects in order to shed light into both the problem (documentation challenges) and the solution (practices and tools); we note that practices and tools are the primary means for architecture documentation [384]. Our aim is to shed light on what is currently on offer for documentation purposes in a CSD context, as well as what is still lacking.

Our results indicate that documentation is considered waste in Lean development when it does not contribute to the end product. Consequently, developers tend to minimize documentation or leave it out. Furthermore, documentation is often out-of-sync with the software, irrespective of whether documentation is within the source code or documented in wiki-like systems. Moreover, the focus is only short-term: knowledge about design decisions, practices, and lessons learned are within a team, primarily when the team is gathered in a single geographical location. The practices we discovered are that written documentation is left out, and communication is informal, while development artifacts are used as a specification. Finally, the use of architecture frameworks can also support sound documentation.

We decided to conduct a systematic mapping study (SMS) instead of a systematic literature review (SLR). Systematic Mapping Study (SMS) are typically used for newer research topics where there are few or no secondary studies and the main objective is to classify and conduct a thematic analysis of literature [232,312]. Further motivation for the use of SMS versus SLR is provided in the beginning of Section 2.

[1] For instance privacy as defined in the General Data Protection Regulation (GDPR).

[2] E.g. low CPU consumption.

1.1. Research questions

We formulate the goal of the study using the format of the Goal-Question-Metric (GQM) approach [39]: **Analyze** literature **for the purpose of** exploration, characterization and analysis **with respect to** documentation challenges, practices, and tools **from the point of view of** researchers and industry practitioners **in the context of** Continuous Software Development. Based on the aforementioned goal, we have set the following research questions:

RQ1 What are the documentation challenges and specific practices in CSD?

We already know of several challenges in CSD. We have established that poor documentation hinders knowledge transfer [S58], which, in turn, has a bad impact on maintenance [S284] and introduces a steep learning [S377] curve for new team members. Furthermore, documentation seems to have a lower value in CSD, than in traditional software development processes such as RUP [245]. For example, the Agile Manifesto explicitly values working code over written documentation; face-to-face communication is considered the most effective way of conveying information [48]. In Lean software development, documentation is often considered waste, as it does not directly contribute to customer satisfaction [317]. In DevOps, infrastructure is key to fast deployment and information is represented as code [194], rather than written documentation. With this research question, we aim at understanding in more depth such challenges that work against documentation in CSD. We also strive to uncover potential practices that result in successful documentation in the context of CSD.

RQ2 Which tools from the Continuous Software Development ecosystem can be used for documentation purposes?

CSD relies heavily on tooling in order to achieve faster deployment, continuous testing, and monitoring quality [S224]. These tools contain much information about source code and configuration (e.g. git), test cases (e.g. Cucumber), deployment (e.g. Docker or Jenkins) and quality (e.g. SonarQube). This information would typically also be documented in software design description documents. With this research question, we want to understand which tools are used in CSD and how they could additionally be exploited for documentation purposes.

1.2. Related secondary studies

There are several secondary studies on the topics of Lean, Agile and DevOps. In the following, we describe those studies that discuss *documentation* in Lean, Agile and DevOps. We also present the reason why they are related and which gap our study attempts to address. These five studies address issues, describe industry practices, and propose and explore processes, tools and methods.

Rodríguez et al. analyze the body of knowledge in Continuous Deployment [328]. They give an overview of concepts and typical characteristics of continuous deployment, such as fast and frequent releases, and continuous automated testing. The authors emphasize the importance of tools for supporting continuous integration and continuous delivery, but they do not address the relation between tooling and documentation.

Diebold et al. looked into agile practices in the industry under different circumstances, such as different project types, domains, or processes [116]. They found that agile practices appear in most projects across several industry domains. Such agile practices are used in methods like Scrum, Kanban, and eXtreme Programming (XP). The study focuses specifically on the development activities that lead to the first major release, whereas maintenance concerns are not particularly taken

into consideration. The study does not cover documentation in agile projects.

In two different secondary studies on requirements engineering in agile software development, Heikkilä et al. [178] and Curcio et al. [104] independently found that there is no clear line on how requirements engineering activities should be performed in agile processes; the overall understanding of requirements engineering in agile software development is still rather immature. Both studies report that an agile development team usually comprises highly skilled and experienced developers who act on their knowledge and skills; this knowledge and the thought process of developers is usually not written down in documents, i.e. agile teams rely mainly on tacit knowledge. Furthermore, these highly skilled professionals are often required for other jobs and frequently switch teams. New team members, who might be less qualified and experienced, do not know the decisions and actions taken. Heikkilä et al. suggest that knowledge should be written down for new team members [178]. Neither the study of Heikilla et al. [178], nor the one of Curcio et al. [104] discussed documentation practices, and tools.

Shafiq et al. found that agile development teams often make use of predefined document templates as a means for efficient standardization [355]. For instance, Feature-Driven Development (FDD) uses templates for use cases and functional requirements, Scrum uses user stories (as <role>, I want <objective> because of <rationale>). Generally, agile teams avoid recording long, complex, strictly-defined or rigid pieces of information in textual documents.

In summary, documentation concerns in CSD are gaining attention within the research community. However, there is currently no consensus on concrete documentation practices. There is no practice or documented tooling that can be used for documentation purposes; instead, information is distributed across software development tools.

1.3. Paper structure and reference styles

The remainder of this document is structured as follows: in Section 2, we present the study design. Section 3 provides demographic information about the selected primary studies. In Section 4, we discuss the results and provide our own interpretation, as well as implications for researchers and practitioners. Finally, we discuss potential threats to the validity of this work in Section 5.

We use two styles of references in this study. One style refers to the primary studies that are analyzed to answer the research questions. These references are denoted with an S (for study) and a number within square brackets, e.g. [S123] refers to study 123 that is shown in Appendix A. The other style of reference is without the 'S'; it refers to papers that do not belong to our set of primary studies but are used for other purposes (for instance in the Related Secondary Studies section) and can be found in the References.

2. Study design

As a method to conduct this literature study, we considered a systematic literature review (SLR), as well as a systematic mapping study (SMS). Table 1 is adapted from Kitchenham et al. [232] and compares typical characteristics of the two methods.

Using these seven characteristics, we justify why we used the systematic mapping study as follows:

(a) **Goals.** We want to present a broad overview of literature and to categorize this literature in dimensions.
(b) **Research Questions.** We address broader research questions regarding the trends in documentation challenges, practices, and tools in CSD.
(c) **Search Process.** We are looking into a specific topic area: documentation in CSD.
(d) **Scope.** We focus on both empirical and non-empirical studies. The topics of Agile, Lean and DevOps are very practitioner-oriented, thus we expect that, at least part of literature is not empirical.
(e) **Search strategy requirements.** We are looking at trends, so we can afford to be less stringent.
(f) **Quality Evaluation.** The combination of non-empirical and empirical studies makes it complicated to evaluate the quality of primary studies.
(g) **Results.** We aim at classifying papers into dimensions.

Based on these reasons, we chose a systematic mapping study over a systematic literature review. We follow the guidelines of Petersen et al. for systematic mapping studies [313]. Fig. 1 depicts the steps of the study as well as the steps of the study protocol. Arrows pointing in both directions indicate that steps were performed iteratively. In the following sections, we briefly describe each of the steps of the study protocol (right part); the steps of "Phase 2: Execute study"(left part) are elaborated in Section 2.4.

Fig. 1 shows the contributions for all team members for each process step. The team comprises four researchers (labeled A, B, C, and D), varying in seniority. Two researchers selected studies, read the title, keywords, abstract and full paper. This resulted in a raw result set with candidate studies. Two other researchers read only the title, keywords and abstracts of the studies in the raw result. Studies that did not contribute to answering the research questions were rejected from the final result set. Finally, all team members read papers from the final result set. In Fig. 4, we made a distinction between raw results with candidate studies and final result sets with studies that contribute to answering research questions. Thus, it shows results per step and studies that were read concerning titles, keywords, abstracts, and full papers. By making a distinction between raw results and final results, we established a process for reaching consensus.

2.1. Search strategy

The search process combines a manual process with automated search. A manual search process typically has a higher accuracy than automated search, because it focuses on targeted venues, but it also has a risk of bias, because of the researcher's personal preferences. Additionally, it is more time-consuming. Other criteria such as transparency and reproducibility are hard to achieve with a manual search, even if all quality and evaluation criteria are explicitly defined [228,235]. Furthermore, automated search is typically more comprehensive than manual searches [228,235]. We therefore decided to apply a combination of both methods. The manual search process, as well as the automated search will be further elaborated in Section 2.4.

2.2. Scope of search and sources searched

The scope for this study is limited by the following criteria:

1. The study is published between January 2001 (i.e. the publication of the Agile Manifesto [48] and February 2019 when the writing of this report started;
2. The study can be found in scientific databases in the field of software engineering, that include journals, conference papers, and workshop papers; the following sources were used: ACM, IEEE, ScienceDirect, SpringerLink and WebOfScience. These databases are quite commonly used in secondary studies in Software Engineering [67].

2.3. Inclusion and exclusion criteria

Inclusion and exclusion criteria help to make a transparent and reproducible selection of papers in the mapping study. Papers are included if they meet all of the inclusion criteria and excluded if they meet any of the exclusion criteria. The criteria are exhibited in Tables 2 and 3.

Table 1
Comparison of typical characteristics in literature research methods.

Characteristic	Systematic Literature Review (SLR)	SMS
Goals	Identification of best practices	Classification and thematic analysis of literature
Research Questions	Specific — related to outcomes of empirical studies	Generic — related to research trends
Search process	Based on research questions	Defined by topic area
Scope	Focused — outcomes of empirical studies	Broad — includes non-empirical studies
Search strategy requirements	Exhaustive — all relevant studies should be found	Less stringent
Quality evaluation	Important. Results must be based on best-quality evidence	Not essential. Non-empirical studies may make quality evaluation hard
Results	Using outcomes of primary studies to answer specific research questions	Categorization of papers into dimensions

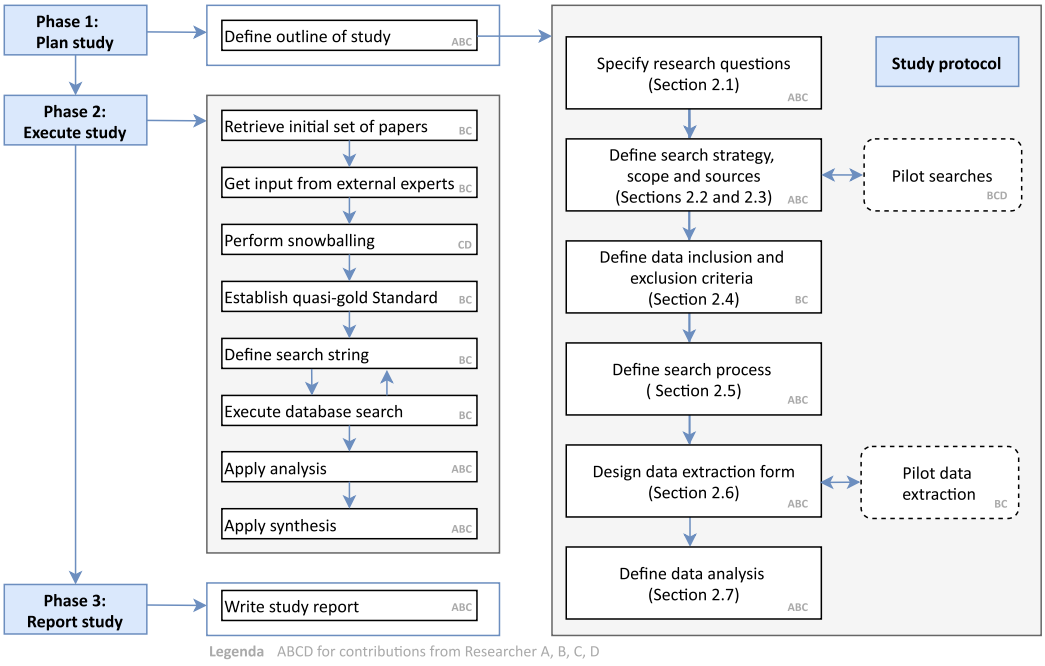


Fig. 1. The study process and the protocol.

Table 2
Inclusion criteria.

ID	Inclusion criteria
I1	PDF or full text must be available
I2	Domain or discipline must be software engineering
I3	Study must be written in English
I4	Study must be peer reviewed
I5	The search terms must appear in title, keywords or abstract

Table 3
Exclusion criteria.

ID	Exclusion criteria
E1	Study is a duplicate of another study in scope

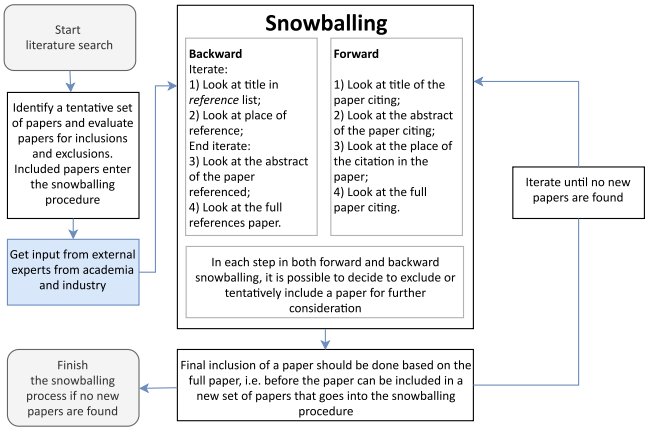


Fig. 2. The snowballing search process.
Source: Adopted from Wohlin [427].

2.4. Search process

The search process follows the steps in the *execution phase* (see Fig. 1, left part). The study was conducted by four researchers. The data collection was done by the corresponding author with the assistance of a master student; the analysis and interpretation was performed by all authors. We began the process with the snowballing technique by following Wohlin's guidelines (see Fig. 2) [427]. Specifically, we formed an initial set of papers on subjects, topics, and authors we found relevant for this SMS. Additionally, we asked subject-matter experts from academia and industry to come up with papers they deem primarily relevant for this study (see Appendix C). With the resulting set of papers, we conducted the snowballing technique until no more relevant publications could be found.

The resulting set of papers was used to define a quasi-gold-standard (QGS), which is a "*well-known*" set of papers that are relevant to evaluate the results and to establish a search string for an automated

search [436]. The search string was based on the QGS. The performance of the search string was performed by comparing the results of the automated search with the QGS: all papers from the QGS were returned from the automated search.

Subsequently, we defined the search string, based on the words from the title, keywords and abstract from the papers in the QGS. We used the n-gram procedure to assist us in establishing the search string [81]. Specifically, we first removed 1000 common English stop words.[3] Next, from the remaining words we took sequences of words to cover the domain (CSD) and the research questions. A manual step was required to adjust the search string to make it more efficient by removing unnecessary terms. Especially for RQ2, we added a wildcard to leave the single "document" out, as searching for "document" resulted in too many irrelevant hits. The resulting search string we used for the automated search is:

```
-- domain
( lean
OR agile
OR DevOps
OR "continuous software"
OR scrum
OR "extreme programming"
)
-- RQ1
AND (
documenti*
OR documenta*
)
-- RQ2
AND (tool*)
)
```

With the search string defined, the execution of the database search was performed. For this, we scraped the meta-data from the online libraries to store it locally in our database. The reason for local storage is to compare studies equally. In the first place, the online search engines all do have a different query language which look similar when it comes to syntax, but the one online library is more precise in targeting than the other online library, especially the discipline (e.g. SpringerLink) or domain (e.g. ACM). Second, the online libraries do not use the same data model for the bibliographical data, for instance, the meta-data has a different format (BibTeX, RIS). Another difference is the type of the fields, such as the fields for "authors", "titles" and "keywords" might either be a string or a list. The third reason is to be able to add tags, labels and comments for answering research questions.

2.5. Data extraction

For each study, the information shown in Table 4 was collected.

The attributes for the *title* (F1), *keywords* (F2) and *abstract* (F3) where used for snowballing and calibration of the search string. We used Mendeley[4] for storing and tagging the papers during the initial phases. With the tags it was easy to select the papers. We commented the papers with keywords and comments for relevance, as well as terms that can be used for the search string. The suggestions for studies from external experts were also stored in Mendeley and tagged accordingly. During snowballing and establishing the Quasi-Gold Standard, Mendeley was used to store, comment the papers and add keywords. With the automated search, the results were too many to store in Mendeley, thus we used a database to store them. The database was structured according to the basic BibTex bibliographic references,[5] supplemented with extra fields for additional keywords, categories, and concepts.

The attribute *full text* (F4) was used in exploring the research area in the pilot search. The attribute values for *publishers database* (F5) and *year* (F6) were required by the inclusion and exclusion criteria. Attributes F7 and F8 were used for answering the two research questions.

2.6. Data analysis

For the analysis of quantitative data, we used descriptive statistics. For the analysis of qualitative data, we adapted the approach of Miles et al. [277], as depicted in Fig. 3. As supporting tooling, we used Atlas.ti.[6] We started with the studies from the final result set. Next, we read the studies and marked text when it answered or contributed to a research question. This resulted in marked text. In the second step, we coded the marked text with keywords that characterize the fragment. The keywords could be individual words from the marked text but also other words that are typical for the marked text. We also used an online thesaurus and used Google to come up with additional or alternative keywords. The result of the coding step is a list of keywords. The third activity was grouping keywords into categories. Categories are a higher-order abstraction of the keywords. The result of the grouping of keywords is a list of categories. The fourth activity concerns identifying relations between categories. The relations denote connections among categories. The types of relations are derived from Unified Modeling Language (UML): activity edges, associations, dependencies, generalizations, realizations, and transitions. We kept the number of relations to a minimum to have clear distinctions between the resulting concepts. The activities for keywords, categories, relations, and concepts were iterated until no more refinement was possible. The last activity was the mapping of the concepts on the systematic map (see Fig. 7).

3. Results

This section describes the results of the mapping study, according to the guidelines of Petersen et al. [313]. First, we show the demographic data of the identified studies (Section 3.1). Then we classify the studies according to our research questions using a facet map (the systematic map) in Section 3.2. Finally, we discuss the results in the context of each individual research question: RQ1 in Section 3.3 and RQ2 in Section 3.4.

3.1. Demographic data

As described in 2.4, we used a two-fold search strategy comprising (1) a purely manual search based on input from subject-matter experts and snowballing and (2) an automated search. The results from both search types were merged, resulting in 58 unique papers that were used for answering our research questions. Fig. 4 illustrates this process again together with the numbers of papers resulting from each individual step. The initial set of four papers was relevant content-wise, but did not pass our inclusion criteria, because they are not scientific studies. Nevertheless, they served as a basis for the snowballing procedure, together with the input from the external experts, who suggested 39 articles in total (see Appendix C), out of which three papers matched our criteria. The snowballing procedure delivered 92 studies. We studied these articles to select a set of nine studies (shown in Table 5) that we consider a Quasi-Gold Standard.

The QGS was used to validate the search string for the automated search, i.e. we tweaked the search string iteratively until all studies in the QGS ended up in the results of the search. Table 6 shows the search

[3] https://gist.github.com/deekayen/4148741.
[4] https://www.mendeley.com.
[5] https://en.wikibooks.org/wiki/LaTeX/Bibliography_Management.
[6] https://atlasti.com/.

Table 4
Data extraction form.

ID	Attribute	Usage
F1	Title	snowballing, search string calibration, synthesis
F2	Keywords	snowballing, search string calibration
F3	Abstract	snowballing, search string calibration
F4	Full text	exploration
F5	Publishers database	inclusion/exclusion
F6	Year	inclusion/exclusion
F7	Documentation, documenting, architecture framework	RQ1
F8	Tools, tooling	RQ2

Table 5
The Quasi-Gold Standard.

ID	Publication
[S407]	*A Study of Documentation in Agile Software Projects* - Voigt, Stefan; von Garrel, Jorg; Muller, Julia; Wirth, Dominic
[S372]	*A study of the documentation essential to software maintenance* - de Souza, Sergio Cozzetti B; Anquetil, Nicolas; de Oliveira, Káthia M
[S373]	*Agile Documentation: A Pattern Guide to Producing Lightweight Documents for Software Projects*; A. Rüping
[S402]	*Software Specification and Documentation in Continuous Software Development: A Focus Group Report* - Van Heesch, U.; Theunissen, T.; Zimmermann, O.; Zdun, U.
[S389]	*Specification in Continuous Software Development* - Theunissen, T., Van Heesch, U.
[S406]	*SprintDoc: Concept for an agile documentation tool* - Voigt, Stefan; Huttemann, Detlef; Gohr, Andreas
[S334]	*Supporting agile software development through active documentation* - Rubin, Eran; Rubin, Hillel
[S159]	*Towards the essentials of architecture documentation for avoiding architecture erosion* - Gerdes, Sebastian; Jasser, Stefanie; Riebisch, Matthias; Schröder, Sandra; Soliman, Mohamed; Stehle, Tilmann
[S346]	*Using design rationales for agile documentation* - Sauer, T

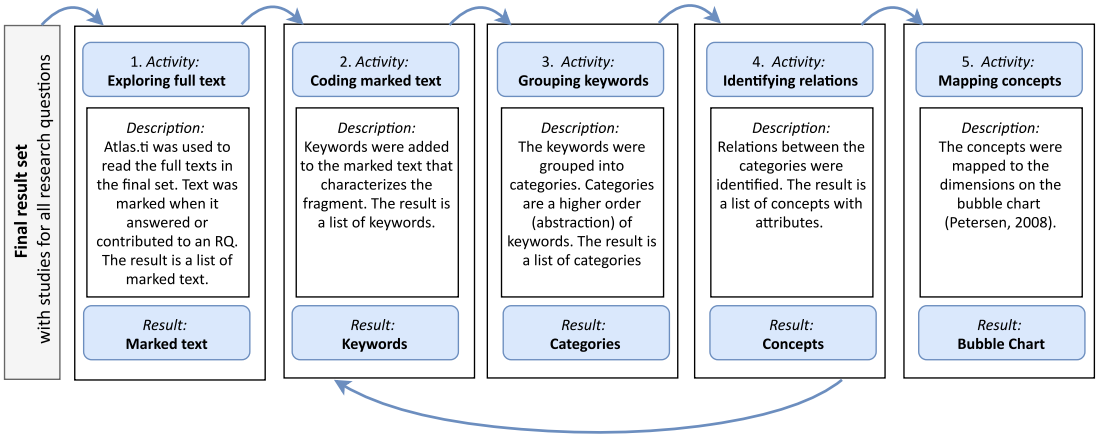


Fig. 3. Qualitative classification process on the final set (Miles et al. [277]).

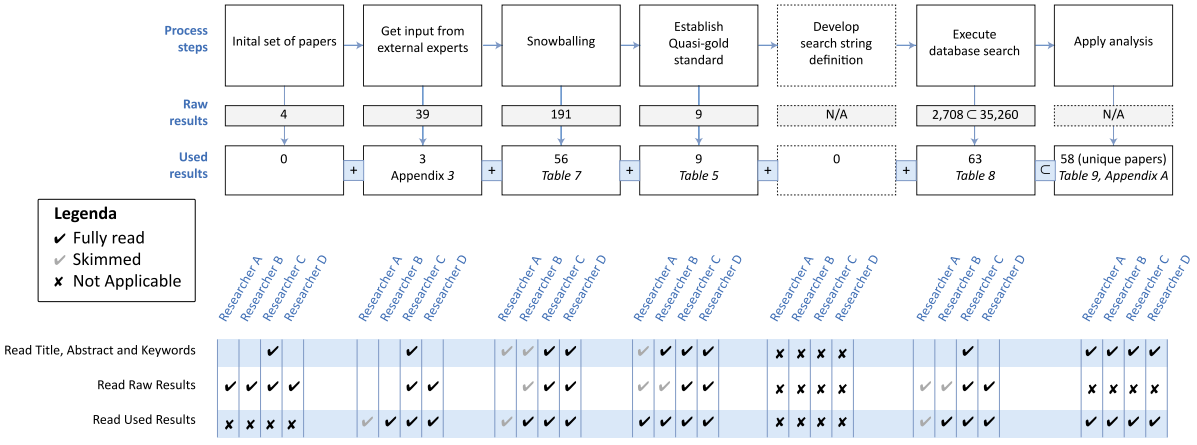


Fig. 4. Overview of search results per step.

string, its relation to the research questions, and the number of hits in abstracts, keywords, or title, respectively. The column *Intersection* shows the number of papers in which the search term was found in all three parts of the studies. In total, we ended up with 58 unique papers that relate to our research questions (i.e. 40 for RQ1 plus 23 for RQ2 makes a total of 63 non-unique papers).

Table 6
Automated search results per RQ and abstract, keywords or title.

RQ	Query	Abstract	Keywords	Title	Union	Intersection
(BASE QUERY)	`(lean OR agile OR DevOps OR 'continuous software' OR scrum OR 'extreme programming')`	8,286	4,552	4,361	9,817	2,708
RQ1	`QUERY AND ('documenti*' OR 'documenta*')`	138	32	29	143	29
RQ2	`QUERY AND (tool*)`	465	68	89	485	29

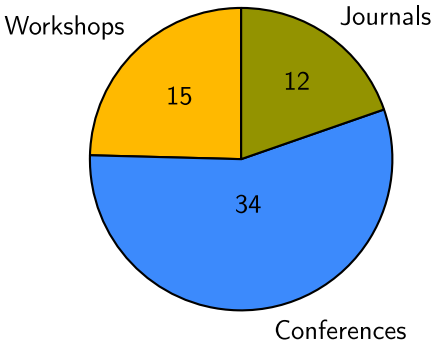


Fig. 5. Distribution per publication type.

Table 7 shows the papers identified during the manual search, Table 8 shows the automated search results. The union resulted in 58 unique studies (see Table 9) that were analyzed in the next step.

The distribution of studies according to publication types is displayed in Fig. 5. About 80% studies are from conferences.

Fig. 6 plots the publication years of all identified studies. Details are printed in Table 10. Clearly, the topic is increasingly gaining attention in the research community. decided to look into the current state of practice

3.2. Classification scheme using a systematic map

As a next step in the analysis, we classified the studies using a systematic map, as described by Petersen et al. [313,312]. Each study was categorized using three facets: (1) a contribution facet covering the type of contribution to the software engineering domain, (2) a research type facet describing the type of study and (3) a context facet that maps the content of the studies to our research questions. Fig. 7 shows the resulting map using two bubble charts. The size of the bubble represents the number of studies falling into the corresponding categories. The absolute number of studies is shown in the centers of the bubbles followed by a letter that refers to the list of studies that can be found in Appendix B. For example, in the coordinate plane between the *Context Facet* "Research" and *Research* Facet "Solution Proposal", the bubble represented by the letter "k" shows that 17 studies have been found. Studies can appear in multiple facets. The total number of 58 unique studies has been mapped 200 times (see Table 10).

For the contribution and research facets, we used existing classification schemes by Petersen et al. [313], and Wieringa [423], respectively. The classification scheme for the **Contribution facets** from Peterson [313] describes the potential categories of a paper's contribution: Metric, Tool, Model, Method, and Process. For our Systematic Mapping Study, we moved the Tool category to the Context facet, because RQ2 concerns tools. The classification scheme for the **Research facets** from Wieringa [423] includes six types, four of which were found in our primary studies:

1. **Validation Research.** The investigation of a problem or implementation of a technique in practice.
2. **Evaluation Research.** The validation of a solution proposal that has not yet been investigated.
3. **Solution Proposal.** This type of papers contains solution proposals without validation.
4. **Experience Papers.** This type of papers describe *what* has been experienced by the author as matters of fact, and do not describe the reasons *why*.

The categories of the context facet evolved while doing the data extraction. We merged categories where appropriate to keep the number of categories small so we could plot them against the other two facets. In the following, those categories are briefly described. Additionally, we assign each category to one of the research questions:

1. *Documentation life cycle*: aspects of creation, maintenance and management of documentation artifacts (RQ1).
2. *Documentation subjects: architecture*: architecture related documentation such as design, solutions and architecture description (RQ1).
3. *Documentation subjects: source-code*: the documentation of source-code and source-code related aspects such as version control (RQ1).
4. *Documentation subjects: autogenerated documentation*: the storing and retrieval of documentation that is scattered throughout a software ecosystem and is stored in tools such as git commits, Jira tasks or wiki-like documents (RQ1). Please note that we omitted the term "documentation" in the bubble chart to ease readability.
5. *Documentation subjects: decisions*: software architecture decisions and their rationale (RQ1).
6. *Tool*: how tools are used in supporting documentation in continuous software development (RQ2).

Fig. 7 shows that architecture documentation is a popular topic within the studies (42 papers in total), predominantly as solution proposals (21 papers) and evaluation research (18 papers). The documentation life cycle is also found in many studies (28 papers), as well as source-code documentation (25 papers), again primarily in the shape of solution proposals or evaluation research.

The most frequent contribution types of the identified studies are method (44 papers), metrics (35 papers), and models (27 papers); all three are mostly found on architecture documentation. It is notable that the least number of studies map to the tool category, which we consider counter-intuitive as continuous software development is a discipline that makes vast use of tool-ecosystems and automation.

3.3. Results for RQ1: Documentation challenges and practices

This section describes the results of our analysis on studies assigned to RQ1 (*What are documentation practices and resulting challenges in CSD?*). Table 9 lists all studies considered in this analysis. Continuous software development, as mentioned in the Introduction section, is not a development process model on its own; it is rather an umbrella term for existing methods that share certain characteristics. The papers found in this mapping study cover the following specific process models and methods:

- Lean Software Development [317],

Table 7
Studies contributing to answering the research questions from the manual search.

RQ	Found	Studies
RQ1	38	[S23], [S55], [S56], [S57], [S58], [S68], [S89], [S90], [S372], [S114], [S156], [S159], [S167], [S181], [S206], [S221], [S255], [261], [S274], [S284], [S316], [S322], [S334], [S340], [S346], [S364], [S375], [S376], [S377], [S407], [S413], [S115], [S175], [S184], [S199], [355], [S389], [S410]
RQ2	18	[S8], [9], [S17], [S23], [S57], [S109], [S112], [S149], [S217], [S250], [S265], [S316], [S340], [S406], [S413], [S414], [S422], [S224]

Table 8
Studies contributing to answering the research questions from the database search.

RQ	Found	Studies
RQ1	40	[S23], [S55], [S56], [S57], [S58], [S68], [S89], [S90], [S372], [S114], [S156], [S159], [S167], [S181], [S206], [S221], [S255], [261], [S274], [S284], [S316], [S322], [S334], [S340], [S346], [S364], [S375], [S376], [S377], [S389], [S407], [S413], [S115], [S132], [S175], [S184], [S199], [355], [S389], [S410]
RQ2	23	[S8], [9], [S17], [S23], [S24], [S57], [S109], [S112], [S149], [S217], [S225], [S250], [S265], [S316], [S330], [S340], [S309], [S406], [S413], [S414], [S422], [S214], [S224]

Table 9
Final set of studies that contribute to answering the research questions.

RQ	Found	Studies
RQ1	40	[S23], [S55], [S56], [S57], [S58], [S68], [S89], [S90], [S372], [S114], [S156], [S159], [S167], [S181], [S206], [S221], [S255], [261], [S274], [S284], [S316], [S322], [S334], [S340], [S346], [S364], [S375], [S376], [S377], [S389], [S407], [S413], [S115], [S132], [S175], [S184], [S199], [355], [S389], [S410]
RQ2	23	[S8], [9], [S17], [S23], [S24], [S57], [S109], [S112], [S149], [S217], [S225], [S250], [S265], [S316], [S330], [S340], [S309], [S406], [S413], [S414], [S422], [S214], [S224]

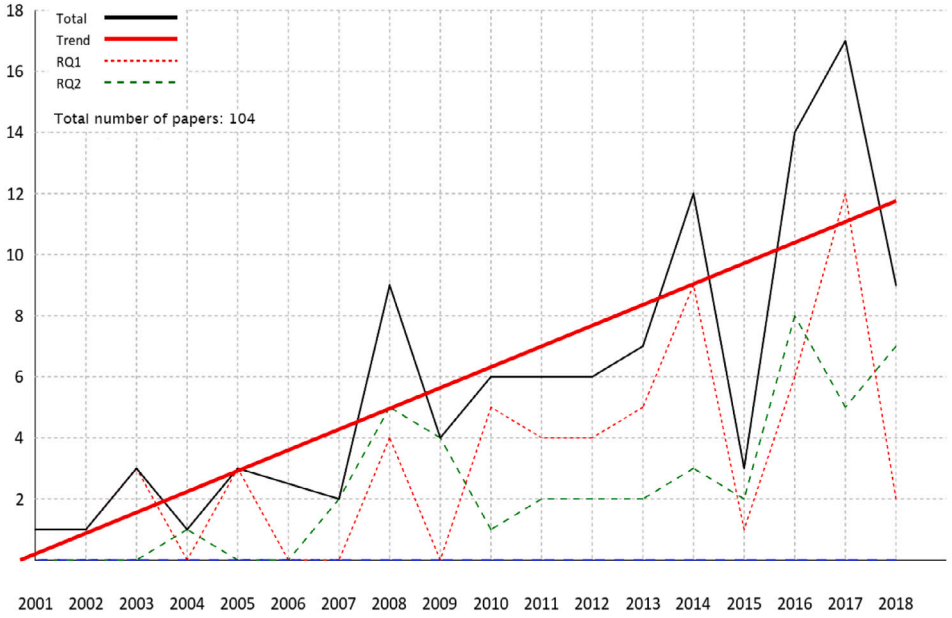


Fig. 6. Distribution per year.

- Scrum [48],
- Extreme Programming (XP) [S139],
- Feature-Driven Development (FDD) [S305],
- Crystal Clear [S95],
- Adaptive Software Development (ASD) [S182],
- Dynamic systems development method (DSDM) [S220,355],
- Microsoft Solution Framework (MSF) [S13,355],
- Agile Unified Process (AUP) [254,355],
- and Test Driven Development [S46].

In the context of these process models and methods, we identified the **documentation challenges** listed below. By documentation challenges, we refer to obstacles that developers face towards documenting knowledge about the system or keeping it up to date.

1. **Informal documentation is hard to understand.** As stated above, the sparse written documentation in CSD is often informal and volatile (e.g. white board sketches and drawings [S375, S156]). Hadar et al. refer to the different backgrounds from architects, reviewers and other stakeholders and note that it is rather cumbersome for one stakeholder to understand and improve the informal documentation of another [S167]. Such types of informal documentation artifacts require a kind of *"voice-over"* or additional explanation to be effective for knowledge transfer [S376].

Table 10
Publications per year for RQ1, RQ2 and totals per year and RQ.

Year	RQ1	RQ2	Total
2001	1	0	1
2002	1	0	1
2003	3	0	3
2004	0	1	1
2005	3	0	3
2006	0	0	0
2007	0	2	2
2008	4	5	9
2009	0	4	4
2010	5	1	6
2011	4	2	6
2012	4	2	6
2013	5	2	7
2014	9	3	12
2015	1	2	3
2016	6	8	14
2017	12	5	17
2018	2	7	9
Total	**60**	**44**	**104**

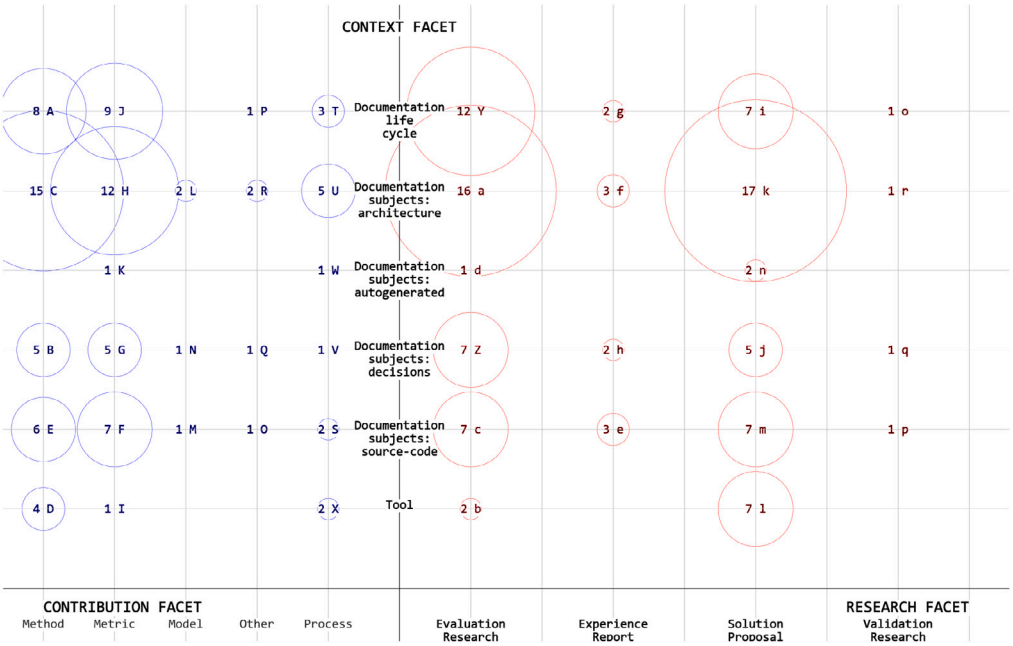


Fig. 7. Mapping of classification facets with papers.

2. **Documentation is considered waste.** Generally, documentation is considered waste when it does not contribute to the end product [317,S322]. Documentation is only created if it is required to create the end product, or to raise the quality of the end product. Prause et al. differentiate between documentation for developers and for end-users. Documentation for developers does not contribute to the end product and is therefore neglected. The source code itself is considered the "ultimate documentation". An example of documentation that does contribute to the end product is a user-manual, for instance [S322]. As a result, design knowledge, reasoning knowledge, as well as knowledge about the problem space are typically not preserved in CSD in any written form.
3. **Productivity is measured by the amount of working software only.** In CSD, productivity is measured by the amount of delivered working software over development time. Beck and other founders of the agile manifesto state that working software is valued over comprehensive documentation [48,226, S269,S410]. Thus, they emphasize that working code is the ultimate measure of productivity; documentation has value, but its comprehensiveness is less important. Stettina et al. note that documentation is rather seen as a burden, than a (co-)created artifact [S375]. This attitude causes developers to generally consider documentation as counter-productive, which in turns causes knowledge loss.
4. **Documentation is out-of-sync with the software.** In CSD, developers do not keep documentation in sync with the actual software [S290]. This applies to both documentation outside the code such as in Microsoft Word documents and wiki-like tools, but also to source code documentation, e.g. regarding the objectives of methods or their parameters. Especially source code documentation is often outdated because CSD emphasizes the continuous update of code, but not its documentation. This is an issue, as stakeholders lose confidence and trust in the documentation [S167], which makes the sparse documentation even less useful. A lack of up-to-date documentation is particularly problematic in the context of architecture design decisions, as it leads to a loss of rationale behind design choices and considered

alternatives; it thus become increasingly difficult to understand and judge solutions during software evolution [S177].

5. **Short-term focus.** Producing comprehensive documentation is a resource-intensive task that interferes with other short-term tasks like sketching diagrams or programming. Primarily, the short term goals of design, programming or maintenance tasks can be achieved without documenting important decisions, documenting rationale, consequences or alternatives [218,S377]. However, this focus on achieving mostly short-term goals has an adverse effect: all the knowledge that is required for those goals disappears over the following iterations with changing context, new objectives and new team members. Nawrocki et al. state that in XP there are three sources of knowledge about the software that are required but are hard to maintain in the long run [S284]: the code, test cases and the memory of the developers.

Despite, the aforementioned challenges, we were also able to observe **documentation practices**. In this study, a "practice" is defined as an activity that is usually or regularly conducted, e.g. as a habit, tradition, rule, or organizational culture.

1. **Non-written and informal communication.** In CSD, **verbal communication** is often used to achieve a mutual understanding between team members (e.g. in [S139]) rather than written documentation. Verbal communication is also one of the twelve principles in the agile manifesto [48], which states that face-to-face communication is both most effective and most efficient within a development team. For agile development in general and XP in particular, Prause et al. state that **knowledge is the result of collaboration** and is spread by different means [S322], other than written documentation. Often only **sketches and informal drawings** are used to support the verbal communication. One exception to this rule is requirements, which are typically documented in the format of user stories [98]. The sparse documentation that is deliberately created for documentation purposes is usually **created afterwards** and describes the state of the software "as is", rather than the software "to be" [S184]; this has the advantage of being up-to-date.
2. **Usage of development artifacts for documentation purposes.** Apart from artifacts created solely for the purpose of documentation, some artifacts created as part of the development process can also serve as a type of documentation. Test Driven Development (TDD) and Behavior Driven Development(BDD) [S389], for instance, lead to executable specifications of the software to be built [S28]. Another form of executable documentation is "infrastructure as code", as mentioned by Callanan et al. [S75]. Infrastructure-as-code refers to any executable description of the infrastructure that is not part of the application itself [194]. This can be achieved with tools like Ansible, or Puppet, for instance.
3. **Architecture frameworks**
   Although architecture knowledge often evaporates in CSD projects, we have seen one particular documentation format, namely architecture frameworks, being used in practice. We briefly describe two examples of frameworks that qualify as architecture frameworks according to the definition in ISO/IEC/IEEE 42010 [200], while specifically addressing concerns of continuous software development. Di Nitto et al. proposed a framework called SQUID (Specification Quality In Devops), an extension of Kruchten's 4+1 view model [S246] with additional viewpoints for dealing with DevOps-related concerns [S114]: an Operations Viewpoint, a Monitoring Viewpoint, a Deployment Viewpoint, and a Quality Verification Viewpoint. Similarly, the continuous integration and delivery architecture framework (Cinders), described by Ståhl en Bosch, is an architecture framework specifically designed to deal with architectural concerns in continuous integration and delivery [S379].

3.3.1. Interpretations of the results

We think that challenges lead to practices, and vice versa, as illustrated in Fig. 8. The first relation is between the challenge of short-term focus (1) that leads to non-written and informal communication. In turn, the non-written and informal communication leads to documentation that is hard to understand (2), and documentation being considered waste when it does not contribute to the end product (3). Using development artifacts for documentation purposes leads to the challenges that productivity is measured by the amount of software only (4) or that documentation is out-of-sync with the software (5). Furthermore, only the practice of architecture frameworks might be considered a practice to contribute to better documentation (see the green box in Fig. 8).

3.3.2. Implications for practitioners

The demand for fast time-to-market leads to fewer artifacts with lower quality. In small teams that are geographically located in one building or one room with long-term employees, informal knowledge is built up in the team. For larger teams, geographically distributed or with changing team members, building up knowledge about the software product and processes might be challenging. A typical practice for documentation is that a whiteboard sketch, and formal API documentation are considered sufficient instead of big upfront documentation with many UML diagrams. Apparently, these informal documentation practices are just enough to start an iteration. At the same time, however, these practices are not sufficient for operations, maintenance, or knowledge transfer. Another candidate approach to overcome these challenges is executable documentation; for TDD, this is a common practice.

3.3.3. Areas for future research

Future research is required to investigate if just enough upfront documentation can be limited to shaping thoughts by using informal whiteboard sketches, together with the codified API documentation. The upfront documentation should be accompanied by design-when-done after an iteration is completed. Anything that can be generated or reverse-engineered is not required to document because it is available anytime. Relevant for operations, maintenance, and knowledge transfer are decisions, considerations about the software product and process, and team organization.

A second area for future research is executable documentation. The practice of TDD, which is one example of executable specification, is a non-intrusive way of documenting requirements as part of the development process. This, however may not be the case for other types of executable specifications. In general, the question how executable documentation can be best produced and consumed by developers is subject to further research.

3.4. Results for RQ2: Tools used in CSD

In RQ2, we investigated, *which tools are used in CSD with respect to documentation*. The studies we considered for answering this research question are listed in Table 9. The studies from the final result are presented in Fig. 7 and Appendix B. As already discussed, the sparse documentation in CSD is scattered over the entire tool ecosystem, mainly in the form of executable specifications. This concerns up-front documentation (mainly requirements), as well as design, code, and deployment information. Table 11 lists tools typically used in CSD practices for documentation purposes, along with the *type* of documentation (mark-down, binary, drawings), and *what* is documented (decisions, annotations, commit messages).

An ideal tool for documentation, according to Cannizzo et al. would support four main practices: status visibility (build tools), extensive automation (i.e. automating as much as possible), effective communication (collaboration and communication tools, metrics tools), and tools for immediate feedback (test tools) [S78]. Other researchers

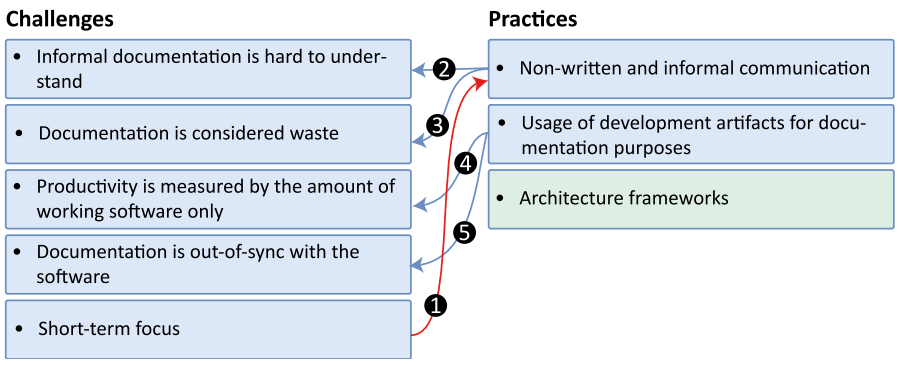


Fig. 8. Possible relations between challenges and practices. The green box indicates a positive effect on the contribution to better documentation.

have proposed simple solutions that can be readily used. For preserving reasoning information, Borrego et al. suggest a simple tagging mechanism [S58]. Buchmann et al. propose a tool for automatically transforming handwritten sketches on paper and whiteboards to high-fidelity UML drawings (tool: Valkyrie) [S73]. Aguiar prefers wikis for documentation, because wikis and agility share goals like simplicity, flexibility, and open collaboration, thus being natural documentation tools to agile projects [S8]. Waits et al. observe that binary files, such as Microsoft Word or PowerPoint files, are hard to maintain, hamper velocity and cannot make efficient use of a version control system (VCS), because changes cannot efficiently be tracked [S413].

Apart from tools used for the purpose of documentation, many tools used in CSD for other purposes also have documentary value. Kersten found that the number of different tools used in CSD is rapidly growing. He explains this phenomenon with a "democratization" of the toolchain, i.e. practitioners choose their own tools for different tasks rather than being obliged by a top-down control model for the tool ecosystem [S224]. This is also the case for documentation. There is no one-size-fits-all documentation tool; on the contrary, practitioners in CSD document what they like, wherever the like.

In the following, we discuss such CSD tools that can be used for documentation purposes. Specifically, we present a list of tool categories together with the type of documentation information associated with each category. The list is compiled from documentation usages found in four primary studies: Kersten presents a landscape for tools and tool-categories [S224]; Partial tool-chains are presented by Poth et al. [319], and Wettinger et al. [S422], who both focus on tools used in CI/CD pipelines; Mäkinen et al. present elements of a modern development toolchain [283].

1. **Development tools**
   (a) **Requirements management tools** (e.g. Blueprint, RequirementONE)
   The documentation information includes stakeholder concerns, risks, constraints and context formulated as *specifications*: these are typically codified instructions, sufficient for developers to start an iteration. Such specifications range from very informal and abstract (e.g. user stories and high-level use-case descriptions) to formal and concrete (e.g. detailed use-case descriptions with pre- and post-conditions, or Cucumber in combination with Gerkin).
   (b) **IDEs** (e.g. IntelliJ, Eclipse, Cloud9)
   The documentation information include the *source code* of the software, often *annotated* with comments, meant for developers to understand the code. The annotations in the code are also used for the automated generation of documentation for APIs.
   (c) **Agile Management tools** (e.g. Active.collab, Agile bench, JIRA)
   Agile management tools are used to support developers in applying agile methods like Scrum, Lean, or Kanban. The tools typically capture information about requirements (e.g. user stories or use-cases), tasks, progress (e.f. burn-down charts), planned and achieved goals for iterations, development speed (e.g. team velocity), and the provide traceability between requirements, tasks, and code (e.g. a JIRA board on which tasks fall under use-cases and are linked to source-code using Git branches and pull requests).
   (d) **Development Analytics tools** (e.g. SonarQube, Metrixware)
   The documentation information includes metrics, as well as *actionable data* that can be used by all roles in the software development including managers, developers, and maintainers. Typical examples of actionable data are quality measurements in SonarQube (a static source code analysis tool) along with concrete suggestions for fixing the quality rule violations.
   (e) **Repositories/Development Communities** (e.g. GitLab, StackShare, StackOverflow)
   The documentation information that repositories hold are the *source code* ready for review, or deployment and *commit messages* where every push to the repository (ideally) includes a meaningful comment on the changes. In development communities (e.g. Stack Overflow of GitHub), the information contains questions and answers for software development topics. As such, it can be considered as a *knowledge base*.
2. **Test tools**
   (a) **Performance/Load / Stress test tools** (e.g. JMeter and SmartStorm)
   These types of test codify specific QoS requirements (typically quality attribute requirements in the categories of performance efficiency, reliability, and security (see ISO/IEC/IEEE 25010 [198].
   (b) **Functional Automation, Virtualization tools** (e.g. Cucumber and Appium)
   The documentation information includes: the *test object* (method, component, API, UI); the test *strategy* with guiding values, principles, and objectives for stakeholders; and test *tactics, and types* with techniques (e.g. destructive/UI tests), processes (e.g. CI/CD) and approaches (e.g. manual/automated testing) applied to specific test *tasks*.
   (c) **Continuous testing tools** (e.g. test.ai, buildbot)
   Continuous testing is a software testing type that involves a process of testing early, often, everywhere, and automatically (to the best possible extent). The objective of continuous testing is to find defects and support immediate response to the defects during the whole development cycle, including requirement specification, development, and maintenance. The documentation information refers

Table 11
Documentation tools in CSD retrieved from the studies for RQ2.

Tool	Short description	Type of information	*What* is documented	Studies
Word, Excel, Powerpoint etc.	Used for reports and intended for long term usage	Binary files	Decisions, drawing, sketches, pictures of whiteboard drawings	[S78], [S364], [S419], [S56], [S57], [S58], [S85], [S156], [S279], [S316], [S322], [S330], [S375], [S377], [S402], [S406], [S411], [S413], [S12], [S38], [S49], [S78], [S99], [S115], [S126], [S164], [S175], [S184], [S188], [S197], [S199], [S220], [S320], [S325], [S332], [S349], [S359], [S361], [S389], [S424]
Wiki	Wiki-likes, Confluence	Short descriptions in the format of the tool, e.g. xml, html	Tasks, how-to's, SRS, SAD, SDD, info on PoCs, Prototypes, Releases	[S56], [S57], [S58], [S85], [S156], [S279], [S316], [S322], [S330], [S377], [S402], [S406], [S411], [S413], [S12], [S38], [S78], [S99], [S115], [S126], [S164], [S188], [S197], [S199], [S320], [S332], [S349], [S359], [S361], [S389], [S424], [S364], [S375], [S419], [S49], [S175], [S184], [S220], [S325]
Source control	Tools for Git, Mercurial, SVN	Source-code, annotations, commit messages	Source-code, commit messages	[S375], [S413], [S175], [S340], [S58],
Scripts	Executable lines of code to run tasks	Human readable text	Infrastructure-as-code: Tests, integration, and deployment including installation, configuration, data import, and security	[169], [S90], [S377], [S115], [S175], [S402], [S389], [S255], [S410]
Markdown editors	Light weight text editor, often used without editor but edited directly	Human readable text	Anything that can be documented in Word, Excel, Powerpoint, or Wikis including configuration files	[S413], [S237]
Verbal communication	The proverbial water-cooler conversation.	Face-to-face	No document, knowledge remains tacit	[S184], [S375], [S175], [S389], [S181], [355], [S410]

to any phase that delivers *measurable software* or *software artifacts*. Test-specifications written for automation purposes (e.g. unit-tests, integration tests and automated end-to-end tests) are functional specifications for source-code units. They can be seen as formal specifications of functional requirements. Often, test-cases also cover QoS-parameters, e.g. the maximum accepted response time of a rest-endpoint.

(d) **Service Virtualization testing tools** (e.g. Smartbear, Parasoft)
Service virtualization is used when the system makes use of an API that is not controlled by the development team. The Service virtualization emulates behavior of the external system, external APIs, cloud-based applications, service-oriented architectures or micro-services that are out of control of the development team. This documentation information includes data, (non-)functional tests or behavior that *emulates the external system*.

3. **Deployment tools**
   (a) **App Automation tools** (e.g. Ansible, Puppet, Chef)
   The documentation information includes instructions for installation, updates and configuration of software, the import of data, and hardening of systems. This information is typically defined in CI/CD scripts. This type of scripts assists in the automation of (complex) IT tasks into *repeatable playbooks*. Although the scripts contain information for a range of tasks, they are typically configured for a single task such as installation or phase such as test.
   (b) **CI/CD** (e.g. CircleCI, Jenkins)
   The documentation information includes the *relation between single tasks* and the *results of executing these tasks*. The automation considers the execution of single tasks from App Automation into a set of scripts for multiple tasks (e.g. installation, configuration, import, hardening) and for multiple stages (e.g. development, test, integration, deployment) whether on premise or in the cloud. The test results show developers or release managers the sanity of the builds in a comprehensive visual overview.
4. **Service execution tools**
   (a) **Cloud/Container Orchestration/Management** (e.g. Docker, Mesos)
   The documentation information includes *metrics* about non-functional requirements such as, but not limited to availability and reliability. It includes also installation and configuration scripts for the software as well as scripts for automated up- or down scaling. This can refer to a single container as well as the orchestration of containers. Infrastructure monitoring tools provide visibility of the complete infrastructure and allow for troubleshooting and resource optimization.
5. **Monitoring tools**
   (a) **Monitoring & Management** (e.g. DataDog, RunDeck)
   In CSD, a lot of processes, and (supporting) applications run at the same time which can lead to a chaotic software development ecosystem as well as a hard to manage deployment pipeline and production environment. The monitoring and management tools support the team to have control of processes and software products. The documentation information captured in the tools contains desired and actual quality-of-service parameters, as well as standard operating procedures for incidents.
6. **Security tools**
   (a) **Container Security** (e.g. AppArmor, Cloud Insights)
   The documentation information for container security involves the *build scripts* for the container and the additional *security policies* (such as non-root user, application isolation, and authentication/authorization).
   (b) **Application Security** (e.g. Threat Stack, HyTrust)

Containerized applications typically make use of a microservice architecture. The information comprises infrastructural security, information on the security aspects (confidentiality, integrity, non-repudiation, accountability and authenticity), information on the distribution of the multiple applications in multiple containers including functionality, data, subsystems, and APIs.

(c) **DevSecOps** (e.g. Cigital, CheckMarx)
DevSecOps refers to the processes and practices to merge the security that is used in development process into processes in operations and vice versa with the purpose of faster deployment with security measures in place. The documentation information includes authentication, and authorization with *roles for users* (where applications are also defined as a user that needs to be authenticated to obtain authorization). It also includes information about *technical (software and hardware) security*, as well as *test procedures* to force and validate security measures.

7. **API management tools and directories**

(a) **API management tools** (e.g. Smartbear, Mashape, RapidAPI, OpenAPI)
These tools document the definition of the *resource description, endpoints with methods, parameters*, often a *request and response example*, and sometimes a *playground* for testing the API.

(b) **API directories** (e.g. ProgrammableWeb)
These provide a *directory of external APIs* that include a description, documentation for developers, SDK, "How to" instructions, (optional) libraries, and information from the community and thus also capture general documentation information about APIs and underlying technologies.

As shown in the list above, documentation information is scattered throughout an entire eco-system of tools rather than being provided in a single self-contained document. As a consequence of the scattering, stakeholders who need to understand the vision and long term goals, architecture decisions, risks, constraints, interface definitions, deployment instructions etc., need to dig into the entire tool-stack [S372, S167]. There is no single source of truth, but there are many sources of truth, each holding information on a relevant part of the software product [328].

3.4.1. Interpretations of the results

There are many tools used in CSD, for every phase (e.g. design, implementation, and testing), as well as every activity (e.g. drawing, collaborating, writing, and constructing). The amount of structure of the information is strongly related to the tool. Some information is easy to capture and easy for human communication such as whiteboard sketches or conversations in chats. At the same time, these types of information are hard for automatic processing. Source code on the other hand, can be automatically processed.

3.4.2. Implications for practitioners

For the construction of the software product, the tools support the developers. As such, the tooling has a positive effect on the productivity. However, with the increase of the number of tools, information about the software product, processes and organization is distributed across these tools.

3.4.3. Areas for future research

A candidate area for future research could be to organize the documentation into yellow pages (wiki, git, markdown) that contains references to relevant documentation for designated stakeholders. For instance PowerPoint slides for conveying ideas about the software product, design documentation for developers and infrastructure-as-code for operations.

4. Discussion

In this section, we interpret our results and provide implications for practitioners and researchers. To begin with the interpretation of the results, the software engineering discipline has always been struggling with documentation. Parnas, for instance, reported back in 1994, that documentation – if written at all – is usually poorly organized, incomplete and imprecise [306]. The human factors that caused this problem back then, are – to the same extent – responsible for the issues reported over documentation in Continuous Software Development today. The difference is that missing or poor documentation was traditionally seen as the result of negligence or even misconduct of developers; in continuous software development it is deliberately promoted to a desired behavior. In other words, CSD puts many obstacles in the way of properly documenting the different aspects of created knowledge (e.g. considering documentation as waste, having a short-term focus, measuring productivity through working software only).

In CSD, with a few mentioned exceptions like documenting requirements, dedicated documentation (i.e. documentation that does not serve as development artifact also) is informal (e.g. white board sketches) and needs to be supported by face-to-face communication. This may not be ideal, but we argue that it can be an effective and efficient approach to support the design reasoning process. However, it cannot effectively preserve knowledge and thus not serve as documentation; this is not a surprise as informal artifacts are not created for the purpose of documenting, but for the purpose of supporting a design discussion.

Knowledge-preserving documentation that stands on its own requires a certain level of formalism and needs to be created for the purpose of describing something unambiguously. Such documentation is very rarely created in CSD projects. Thus, in our opinion, the documentation practices in CSD – or lack thereof – do not contribute to solving the traditional problems related to knowledge loss and missing information during maintenance activities. Unfortunately, we have not seen evidence of new or emerging practices that can alleviate this problem.

Although the results we found regarding dedicated documentation *practices* in CSD are sobering, there is also good news. With the rise of Lean, Agile and DevOps projects, we observe a drastic boost in *tool-ecosystems*, which mainly stems from the goal to automate software-related processes as much as possible. This also enables new ways of thinking about documentation. The specifications required for automating processes (we refer to them as executable specifications), at the same time serve as documentation of the process. This essentially urges us to refine Robert Martin's statement that the truth can only be found in the code: now the truth can also be found in test scripts, provisioning scripts, build pipeline configurations, and cloud platform configurations, to name just a few.

4.1. Characteristics of executable documentation

Executable specifications have a lot of potential for serving as documentation in CSD. Their characteristics are in line with the principles of CSD, and at the same time address the previously mentioned traditional problems that come with missing documentation. We highlight the following characteristics of executable documentation that require further research.

4.1.1. Types of executable documentation

Types of executable documentation include: requirements, such as "Specification by Example" [152], test-cases resulting from Domain-Driven-Design [404] or TDD [S292,S334,S175,S28], database scripts with Data Definition Language (DDL) and Data Manipulation Language (DML), deployment scripts with Ansible [S389] or infrastructure-as-code [72].

4.1.2. Executable documentation is never out-of-sync

Executable documentation is never out-of-sync, it's evolution is naturally connected to the evolution of the other parts of the software. Executable documentation does not just describe the software, but it is part of the software.

4.1.3. Executable documentation can be tested

Executable documentation can be tested. If it does not lead to the desired results, then something must be wrong. In that respect, executable documentation is just like source-code.

4.1.4. Executable documentation is non-intrusive

The process of creating executable documentation is not intrusive, i.e. developers do not stop their work to take care of documentation; coding and documenting are part of the same task.

In future research, these items will be investigated. Questions remain, for example, how can software development teams use such executable specifications? This could include a considerable amount of unstructured (and unrelated) data.

4.2. Implications for practitioners

In the following, we present some implications for practitioners who want to benefit from the potential of CSD to document the created knowledge.

4.2.1. Tools, tool-stacks, and software development ecosystems

Support your entire development process by a tool-chain that seamlessly supports all activities in the process. Eliminate manual or interactive steps in the development process to the greatest possible extent. Manuals for developers describing process steps to follow (or the need for such manuals) should be considered as bad smells [193] that should be transferred to executable specifications interpreted by tools. Executable specifications are always up-to-date and at the same time document processes in an unambiguous way that can be interpreted by both machines and humans.

4.2.2. Informal sketches

Use informal sketches (that are minimally intrusive) to support your design reasoning process and discussions with team members. The reasoning process and discussions ultimately lead to decisions that are implemented (e.g. in source-code or executable specifications). Consider briefly documenting the rationale behind those decisions that may not be obvious to other stakeholders (including future developers). Examples of obvious decisions are choices of tools or combinations of tools that are very popular for certain purposes, e.g. the combination of Elasticsearch, Logstash, and Kibana for distributed logging and analytics.

4.2.3. Use of version control

Keep everything under version control. Use project management tools or wikis as a central entry point for all information related to the project; otherwise, stakeholders may easily get lost in the great amount of project locations, tools and URLs. Also consider providing high-level overviews of the designed sub-systems, and link the respective executable specifications to the sub-systems to facilitate access for stakeholders.

4.3. Future research

In terms of research, the results of this mapping study have shown that documentation in CSD, has not yet gained the required attention by the research community.[7] In the following, we describe three areas for future research:

1. The individual tools in a CSD ecosystem are mostly created separately, thus having limited interoperability. However, the combination of information from different tools can be "more than the sum of its parts", i.e. it can provide insights that capture a greater part of the system and life cycle. Thus research is required to establish traceability links between the different types of tools and intelligently combine information from different kinds of executable documentation in dashboards.
2. Traditional architecture documentation approaches seem to come in direct conflict with the identified documentation challenges in CSD. Research is required to develop architecture documentation and specification approaches that integrate seamlessly in CSD-practices. For example, architecture frameworks could be developed that tap the potential of executable specifications, while preserving design rationale, explaining architecture to stakeholders, linking design decisions to concerns and architectural requirements and providing an informational basis for architectural evaluation.
3. The high degree of automation offers rich sources of information that can be mined using *Mining Software Repository* techniques, or in general *Data Science*. Examples of questions that could be addressed using such approaches in CSD are:
   - What is the current technical debt in source code, testing, requirements or other types?
   - What design decisions are likely to be outdated soon?
   - What is the cost–benefit ratio of specific features?
   - What is the optimal point in time for refactoring a specific sub-system?

5. Threats to validity

We use the framework of Ampatzoglou et al. [21] that describes potentials threats to validity for secondary studies. Specifically this framework classifies threats to validity in three categories, as illustrated in Table 12.

5.1. Study selection validity

Regarding the selection of digital libraries, we have to a large extent addressed this by including the most used digital libraries in this area (which are also commonly used in secondary studies in software engineering). The construction of the search string may lead to yielding too many primary studies or missing relevant studies. We mitigated this threat by calibrating the search string through the quasi-gold standard. Specifically the QGS was used to assess the performance of the search string and refine it until all primary studies of the QGS were returned from the search string. The QGS itself was built using the snowballing technique guidelines as proposed by Wolhin [427].

Furthermore, we have mitigated the risk of an arbitrary starting year, because it was related to the year of the publication of the Agile Manifesto. With this decision we excluded a historic overview of consecutive concepts that lead to the Agile Manifesto; however, we did not aim at such a historic overview but a systematic classification and thematic analysis of literature.

[7] In August 2019, "executable documentation" had the following results: ACM: 9; Google Scholar: 207; IEEE: 2; ScienceDirect: 15; SpringerLink: 35; Web of Science: 3

Table 12
Classification of validity threats (Ampatzoglou et al. [21]).

Category	Description
Study Selection Validity	These threats can be identified in the initial search process where the set of candidate papers for primary studies is selected and the study filtering where the final set of primaries studies is determined. Typical examples are the selection of digital libraries, search string construction and study selection bias.
Data Validity	These threats can be identified in the data extraction phase (a data set is populated) and data analysis phase (the data set is qualitatively or quantitatively analyzed). Typical examples include data collection bias and publication bias.
Research Validity	These threats can be identified over the whole mapping study and concern the design of the research. Typical examples are generalizability, and coverage of research questions.

The threat for non-English papers was not mitigated; these papers were excluded. We did however address the threat of studies not being accessible: we made sure we could access all studies. The threat of duplicate articles was mitigated by filtering on the Document Object Identifier. If a study appeared in multiple digital libraries, then the publishers' digital library was used and the duplicate was ignored. We excluded gray literature and included only studies from peer reviewed journals, conferences or workshops to have more rigor. Finally, the potential bias of study inclusion/exclusion was mitigated by discussion among the authors and accordingly revising the inclusion/exclusion criteria.

5.2. Data validity

The risk of retrieving a small sample was mitigated by constructing a search string that could zoom in from a domain with over approximately 35.000 studies to finally about 200 relevant papers to answer the research questions. The threat of choosing the correct variables to be extracted was addressed through extensive discussions between the authors. The threat of publication bias (the majority of identified primary studies coming from specific venues) was mitigated by using snowballing. Furthermore, we addressed the threat of inadequate validity of primary studies through the inclusion criteria by only looking at peer reviewed venues. The threat of biasing the classification schema is mitigated by going through several iterations to refine the RQs, and redefining the search string and the analysis process. The threat of researchers' bias was partially mitigated by doing the analysis with multiple researchers where research and review were different roles, and by using a combination of manual and automated search.

5.3. Research validity

The threat of repeatability is mitigated by meticulously documenting the study protocol. In addition, the retrieved studies, search strings and data are all available on https://theotheunissen.nl/SMS. The threat of the chosen research method bias is mitigated by extensive discussions among the authors and the rationale of our decision is clearly described in the study design section. Furthermore, the authors have also discussed in multiple iterations the choice and coverage of the research questions. Regarding the generizability of our results, they are only applicable within the scope of documentation in continuous software development.

6. Conclusions

We conducted a systematic mapping study to investigate the documentation practices and challenges, as well as the tooling used in continuous software development (CSD). The study has provided an overview of the relevant primary studies and has listed a number of challenges, practices, and tools that pertain to documentation in CSD.

Section 3.3 elaborates on our findings regarding documentation challenges and practices (RQ1). The challenges include: informal documentation is hard to understand, documentation is considered waste when it does not contribute to the end product, productivity is limited to the measured amount of working software, documentation is easily out-of-sync with the actual code, and there is short term focus. The practices include: a significant amount of communications happens verbally and informally; there is a positive usage of development artifacts for documentation purposes, such as TDD; and the use of architecture frameworks might positively influence documentation quality.

Section 3.4 discusses an increasing number of tool categories and tools that can be used to support development, operations, and maintenance in CSD (RQ2). CSD is a high-paced evolving and dynamic environment; without tools, development and deployment would not be possible. An interesting side-effect of the tooling that has not been adequately researched yet, is that with every tool that is being used, knowledge about the piece of software is stored, maintained and transferred as well. For example, commits in a repository describe the changes of the source code and test scripts in a test tool describe the required outcomes of software. Knowledge about the software is scattered throughout all the tools in a software ecosystem. There is not a single source of truth, but there are a lot of sources of truth, each holding a small piece of knowledge. The discovery of these pieces of knowledge has not been investigated and it could be interesting to do further research on how to locate and combine these information sources.

Finally, we identified several implications for practitioners regarding the use of executable specifications in combination with a high degree of automation. Additionally, we found that architecture frameworks streamlined for use in CSD and dashboards combining information from the entire development tool chain are important areas for future research.

CRediT authorship contribution statement

Theo Theunissen: Writing – original draft, Writing – review & editing. **Uwe van Heesch:** Writing – review & editing, Supervision. **Paris Avgeriou:** Writing – review & editing, Supervision.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.infsof.2021.106733.

Appendix A. List of studies

Please see Further reading section.

Appendix B. Studies in the bubble chart

This section lists all studies visualized in the bubble chart from Fig. 7. For each table, we present the number of studies along with the respective references to these studies. Appendix A contains the complete list of papers with full bibliographical details.

Category	A: Method, Documentation life cycle
Papers	[S340], [S284], [S23], [S17], [S206], [S316], [S346], [S376]
Found	8

Category	B: Method, Documentation subjects: decisions
Papers	[S340], [S55], [S17], [S206], [S376]
Found	5

Category	C: Method, Documentation subjects: architecture
Papers	[S340], [355], [S23], [S55], [S265], [S89], [S56], [S389], [S17], [S181], [S316], [S90], [S377], [S346], [S376]
Found	15

Category	D: Method, Tool
Papers	[S340], [S55], [S56], [S181]
Found	4

Category	E: Method, Documentation subjects: source-code
Papers	[355], [S55], [S17], [S181], [S377], [S376]
Found	6

Category	F: Metric, Documentation subjects: source-code
Papers	[S372], [S413], [S156], [S406], [S274], [S112], [S364]
Found	7

Category	G: Metric, Documentation subjects: decisions
Papers	[S372], [S156], [S406], [S274], [S112]
Found	5

Category	H: Metric, Documentation subjects: architecture
Papers	[S372], [S413], [S156], [S406], [S274], [S217], [S407], [S58], [S112], [S316], [S364], [S224]
Found	12

Category	I: Metric, Tool
Papers	[S372]
Found	1

Category	J: Metric, Documentation life cycle
Papers	[S413], [S156], [S406], [S217], [S407], [S58], [S316], [S68], [S364]
Found	9

Category	K: Metric, Documentation subjects: autogenerated
Papers	[S112]
Found	1

Category	L: Model, Documentation subjects: architecture
Papers	[S322], [S274]
Found	2

Category	M: Model, Documentation subjects: source-code
Papers	[S274]
Found	1

Category	N: Model, Documentation subjects: decisions
Papers	[S274]
Found	1

Category	O: Other, Documentation subjects: source-code
Papers	[S389]
Found	1

Category	P: Other, Documentation life cycle
Papers	[S389]
Found	1

Category	Q: Other, Documentation subjects: decisions
Papers	[S389]
Found	1

Category	R: Other, Documentation subjects: architecture
Papers	[S389], [S159]
Found	2

Category	S: Process, Documentation subjects: source-code
Papers	[S375], [S406]
Found	2

Category	T: Process, Documentation life cycle
Papers	[S375], [S406], [S410]
Found	3

Category	U: Process, Documentation subjects: architecture
Papers	[S375], [S406], [S57], [S410], [S90]
Found	5

Category	V: Process, Documentation subjects: decisions
Papers	[S406]
Found	1

Category	W: Process, Documentation subjects: autogenerated
Papers	[S57]
Found	1

Category	X: Process, Tool
Papers	[S57], [S410]
Found	2

Category	Y: Evaluation Research, Documentation life cycle
Papers	[S340], [S413], [S156], [S284], [S406], [S217], [S407], [S58], [S206], [S316], [S68], [S364]
Found	12

Category	Z: Evaluation Research, Documentation subjects: decisions
Papers	[S340], [S372], [S156], [S406], [S274], [S112], [S206]
Found	7

Category	a: Evaluation Research, Documentation subjects: architecture
Papers	[S340], [S372], [S413], [S156], [S406], [S265], [S274], [S89], [S217], [S407], [S58], [S112], [S316], [S159], [S364], [S224]
Found	16

Category	b: Evaluation Research, Tool
Papers	[S340], [S372]
Found	2

Category	c: Evaluation Research, Documentation subjects: source-code
Papers	[S372], [S413], [S156], [S406], [S274], [S112], [S364]
Found	7

Category	d: Evaluation Research, Documentation subjects: autogenerated
Papers	[S112]
Found	1

Category	e: Experience Report, Documentation subjects: source-code
Papers	[355], [S389], [S17]
Found	3

Category	f: Experience Report, Documentation subjects: architecture
Papers	[355], [S389], [S17]
Found	3

Category	g: Experience Report, Documentation life cycle
Papers	[S389], [S17]
Found	2

Category	h: Experience Report, Documentation subjects: decisions
Papers	[S389], [S17]
Found	2

Category	i: Solution Proposal, Documentation life cycle
Papers	[S340], [S375], [S23], [S410], [S255], [S346], [S376]
Found	7

Category	j: Solution Proposal, Documentation subjects: decisions
Papers	[S340], [S372], [S55], [S274], [S376]
Found	5

Category	k: Solution Proposal, Documentation subjects: architecture
Papers	[S340], [S372], [S375], [S23], [S322], [S55], [S57], [S274], [S89], [S56], [S389], [S181], [S410], [S90], [S377], [S346], [S376]
Found	17

Category	l: Solution Proposal, Tool
Papers	[S340], [S372], [S55], [S57], [S56], [S181], [S410]
Found	7

Category	m: Solution Proposal, Documentation subjects: source-code
Papers	[S372], [S375], [S55], [S274], [S181], [S377], [S376]
Found	7

Category	n: Solution Proposal, Documentation subjects: autogenerated
Papers	[S57], [S255]
Found	2

Category	o: Validation Research, Documentation life cycle
Papers Found	[S156] 1

Category	p: Validation Research, Documentation subjects: source-code
Papers Found	[S156] 1

Category	q: Validation Research, Documentation subjects: decisions
Papers Found	[S156] 1

Category	r: Validation Research, Documentation subjects: architecture
Papers Found	[S156] 1

Appendix C. Input from experts

The email we send to the experts had this content:

Dear reader,

I am conducting a systematic mapping study to research the literature on documentation, tooling and existing frameworks in continuous software development (or: agile, lean, DevOps, CI/CD).

Your input as an academic researcher or industry practitioner in this area is appreciated.

BACKGROUND
Documentation of software architecture, design, development and operations have a long tradition of both storing knowledge and communicating decisions. At the same time, documentation is a tedious, time-consuming task that is usually reduced to a minimum in continuous software development processes such as lean, agile and DevOps. Continuous software development has been discussed in. The focus of this mapping study is on documentation practices in continuous software development processes such as lean, agile and DevOps. These development processes are the de facto standards in many small startups as well as in large enterprises. A mapping study for documentation in continuous software development processes does not exist. Because documentation in these processes deviates from textbook standards that are taught during education, and there is no prescribed standard but just a practice of documentation, this study is relevant for both researchers, practitioners, and educators. This mapping study is an assessment of existing literature on development processes, documentation methods, and frameworks --- including tools. It aims to find and classify the primary studies in this specific topic area.

RESEARCH QUESTIONS
RQ1: What studies exist on documentation practices in continuous software development (CSD)?
Rationale: Documentation plays a major role in preserving knowledge and communicating decisions in software architecture and technical implementation. At the same time, documentation practices in CSD are lacking. With this research question, an overview of documentation methods will be presented.

RQ2: What studies exist on tools used in CSD?
Rationale: In the community of practice for continuous software development, tools are used to speed up development, monitor quality, and automatic deployment. This documentation is not exported to a central repository but kept with the tool, e.g. Jira, GitHub. The focus is primarily on tools that are described in the literature but will be extended to tools that are actually used for architects and developers.

1. *A Study of Enabling Factors for Rapid Fielding: Combined Practices to Balance Speed and Stability*, Stephany Bellomo and Robert L. Nord and Ipek Ozkaya, 2013
2. *A Study of the Documentation Essential to Software Maintenance*, Sergio Cozzetti B. de Souza and Nicolas Anquetil and Káthia M. de Oliveira, 2005
3. *Agile Architecture Interactions*, J. Madison, 2010
4. *Agile Architecture IS Possible You First Have to Believe!*, M. Isham, 2008
5. *Agile Documentation, Anyone?*, B. Selic, 2009
6. *Agile Documentation: A Pattern Guide to Producing Lightweight Documents for Software Projects*, Rüping, Andreas, 2005
7. *Agile in Distress: Architecture to the Rescue*, Robert L. NordIpek OzkayaPhilippe Kruchten, 2014
8. *Agile software development: the business of innovation*, J. Highsmith; A. Cockburn, 2001
9. *Agility and Architecture: Can They Coexist?*, P. Abrahamsson; M. A. Babar; P. Kruchten, 2010
10. *Analysis and Management of Architectural Dependencies in Iterative Release Planning*, Brown, Nanette and Nord, Robert L. and Ozkaya, Ipek and Pais, Manuel, 2011
11. *Architecting for Large-Scale Agile Development: A Risk-Driven Approach*, Ipek Ozkaya, Michael J. Gagliardi, Robert Nord, 2013
12. *Architecting for sustainable software delivery*, Koontz, Ronald J and Nord, Robert L, 2012
13. *Architecting in a Complex World: Eliciting and Specifying Quality Attribute Requirements*, Wojcik, Rob, 2013
14. *Architects as Service Providers*, R. Faber, 2010
15. *Beyond Scrum + XP: Agile Architecture Practice*, Ozkaya. Ipek, Robert L. Nord, Stephany Bellomo, and Heidi Brayer, 2013
16. *Climbing the "Stairway to Heaven" – A Mulitiple-Case Study Exploring Barriers in the Transition from Agile Development Towards Continuous Deployment of Software*, Olsson, Helena Holmstrom and Alahyari, Hiva and Bosch, Jan, 2012
17. *Combining architecture-centric engineering with the team software process*, Nord, Robert L and McHale, James and Bachmann, Felix, 2010
18. *DevOps: A Software Architect's Perspective*, Bass, Len and Weber, Ingo and Zhu, Liming, 2015
19. *Elaboration on an integrated architecture and requirement practice: Prototyping with quality attribute focus*, S. Bellomo; R. L. Nord; I. Ozkaya, 2013

20. *Enabling agility through architecture*, Brown, Nanette and Nord, Robert and Ozkaya, Ipek, 2010
21. *Enabling Incremental Iterative Development at Scale: Quality Attribute Refinement and Allocation in Practice*, 2015
22. *Evolutionary Improvements of Cross-Cutting Concerns: Performance in Practice*, Bellomo, Stephany and Ernst, Neil and Nord, Robert L and Ozkaya, Ipek, 2014
23. *Integrate End to End Early and Often*, Bachmann, Felix H and Carballo, Luis and McHale, James and Nord, Robert L, 2013
24. *Making Architecture Visible to Improve Flow Management in Lean Software Development*, R. L. Nord; I. Ozkaya; R. S. Sangwan, 2012
25. *Microservices Architecture Enables DevOps Migration to a Cloud-Native Architecture*, Balalaie, Armin and Heydarnoori, Abbas and Jamshidi, Pooyan, 2016
26. *Microservices tenets*, Olaf Zimmermann, 2017
27. *Migrating to Cloud-Native Architectures Using Microservices: An Experience Report*, Armin Balalaie, Abbas Heydarnoori, and Pooyan Jamshidi, 2015
28. *Peaceful Coexistence: Agile Developer Perspectives on Software Architecture*, D. Falessi; G. Cantone; S. A. Sarcia'; G. Calavaro; P. Subiaco; C. D'Amore, 2010
29. *Presenting a framework for agile enterprise architecture*, B. D. Rouhani; H. Shirazi; A. F. Nezhad; S. Kharazmi, 2008
30. *Software Architecture for Developers Technical leadership by coding, coaching, collaboration, architecture sketching and just enough up front design*, Brown, Simon, 2014
31. *Software Specification and Documentation in Continuous Software Development: A Focus Group Report*, U. Van Heesch and T. Theunissen and O. Zimmermann and U. Zdun, 2017
32. *Sustainable Architectural Design Decisions*, Zdun, Uwe and Capilla, Rafael and Tran, Huy and Zimmermann, Olaf, 2013
33. *The DevOps Handbook : How to Create World-Class Agility, Reliability, and Security in Technology Organizations*, Kim, Gene; Humble, Jez; Debois, Patrick; Willis, John, 2016
34. *The impact of agile practices on communication in software development*, M. PikkarainenJ. HaikaraO. SaloP. AbrahamssonJ. Still, 2008
35. *The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win*, Behr, Kevin; Kim, Gene; Spafford, George, 2014
36. *Toward Design Decisions to Enable Deployability: Empirical Study of Three Projects Reaching for the Continuous Delivery Holy Grail*, S. Bellomo; N. Ernst; R. Nord; R. Kazman, 2014
37. *Understanding the Role of Constraints on Architecturally Significant Requirements*, Neil Ernst, Ipek Ozkaya, Robert Nord, Julien Delange, Stephany Bellomo, Ian Gorton, 2013
38. *Variations on Using Propagation Cost to Measure Architecture Modifiability Properties*, Nord, Robert L. and Ozkaya, Ipek and Sangwan, Raghvinder S. and Delange, Julien and González, Marco and Kruchten, Philippe, 2013
39. *Working Together: The Team Software Process and Architecture-Centric Engineering*, 2013

References

[1] Nik Nailah Binti Abdullah, Shinichi Honiden, Helen Sharp, Bashar Nuseibeh, David Notkin, Communication patterns of agile requirements engineering, in: Proceedings of the 1st Workshop on Agile Requirements Engineering, in: AREW '11, Association for Computing Machinery, ISBN: 9781450308908, 2011, pp. 1:1–1:4, http://dx.doi.org/10.1145/2068783.2068784.

[2] Pekka Abrahamsson, Nilay Oza, Mikko T. Siponen, Agile software development methods: A comparative review, in: Torgeir Dingsøyr, Tore Dybå, Nils Brede Moe (Eds.), Agile Software Development, Springer, 2010, pp. 31–59, http://dx.doi.org/10.1007/978-3-642-12575-1.

[3] Pekka Abrahamsson, Outi Salo, Jussi Ronkainen, Juhani Warsta, Agile software development methods: Review and analysis, in: Espoo, Finland: Technical Research Centre of Finland, VTT Publications, (ISSN: 12350621) 2002, p. 478, http://dx.doi.org/10.1076/csed.12.3.167.8613.

[4] Pekka Abrahamsson, M.A. Babar, Philippe Kruchten, uhammad Ali Babar, Philippe Kruchten, Guest editors' introduction agility and architecture:, IEEE Softw. (ISSN: 0740-7459) 27 (2) (2010) 16–22, http://dx.doi.org/10.1109/MS.2010.36.

[5] Gojko Adzic, Specification by example, in: Book Your Training with Díaz & Hilterscheid!, 2011, p. 20.

[6] Agile 2008 Toronto: Agile infrastructure and operations presentation, 2019, http://www.jedi.be/blog/2008/10/09/agile-2008-toronto-agile-infrastructure-and-operations-presentation/ (Accessed on 04/01/2019).

[7] Ashish Agrawal, Mohd. Aurangzeb Atiq, L.S. Maurya, A current study on the limitations of agile methods in industry using secure google forms, in: Physics Procedia, 78, 2016, pp. 291–297, http://dx.doi.org/10.1016/j.procs.2016.02.056.

[9] Muhammad Ovais Ahmad, Denis Dennehy, Kieran Conboy, Markku Oivo, Kanban in software engineering: A systematic mapping study, J. Syst. Softw. (ISSN: 01641212) 137 (2018) 96–113, http://dx.doi.org/10.1016/j.jss.2017.11.045.

[10] N. Ajmeri, R. Sejpal, S. Ghaisas, A semantic and collaborative platform for agile requirements evolution, in: 2010 Third International Workshop on Managing Requirements Knowledge, 2010, pp. 32–40, http://dx.doi.org/10.1109/MARK.2010.5623810.

[15] Muhammad Sarmad Ali, Muhammad Ali Babar, Lianping Chen, Klaas-Jan Stol, A systematic review of comparative evidence of aspect-oriented programming, Inf. Softw. Technol. 52 (9) (2010) 871–887.

[18] Scott Ambler, Agile Modeling: Effective Practices for EXtreme Programming and the Unified Process, John Wiley & Sons, 2002, p. 400, http://dx.doi.org/10.1017/CBO9780511817533.018.

[19] Scott W. Ambler, Agile/lean documentation: Strategies for agile software development, 2012.

[20] Scott W. Ambler, Mark Lines, The disciplined agile process decision framework, in: Lecture Notes in Business Information Processing, Vol. 238, 2016, pp. 3–14, http://dx.doi.org/10.1007/978-3-319-27033-3_1.

[21] Apostolos Ampatzoglou, Stamatia Bibi, Paris Avgeriou, Marijn Verbeek, Alexander Chatzigeorgiou, Identifying, categorizing and mitigating threats to validity in software engineering secondary studies, Inf. Softw. Technol. (2018) http://dx.doi.org/10.1016/j.infsof.2018.10.006.

[22] Anita, N. Chauhan, A regression test selection technique by optimizing user stories in an agile environment, in: 2014 IEEE International Advance Computing Conference (IACC), 2014, pp. 1454–1458, http://dx.doi.org/10.1109/IAdCC.2014.6779540.

[26] Erik Arisholm, Lionel C Briand, Siw Elisabeth Hove, Yvan Labiche, The impact of UML documentation on software maintenance: An experimental evaluation, IEEE Trans. Softw. Eng. (ISSN: 00985589) 32 (6) (2006) 365–381, http://dx.doi.org/10.1109/TSE.2006.59.

[27] Elvira Maria Arvanitou, Apostolos Ampatzoglou, Alexander Chatzigeorgiou, Matthias Galster, Paris Avgeriou, A mapping study on design-time quality attributes and metrics, J. Syst. Softw. (ISSN: 01641212) 127 (2017) 52–77, http://dx.doi.org/10.1016/j.jss.2017.01.026.

[29] Barbara A. Association for Computing Machinery, Tore A.C.M. Sigsoft, Magne IEEE Computer Society, Technical Council on Software Engineering, Proceedings, 26th International Conference on Software Engineering : ICSE 2004 : May 23-28, 2004, Edinburgh International Conference Centre, Edinburgh, Scotland, Association for Computing Machinery, 2004, p. 786.

[30] M. Auer, T. Tschurtschenthaler, S. Biffl, A flyweight UML modelling tool for software development in heterogeneous environments, in: 2003 Proceedings 29th Euromicro Conference, 2003, pp. 267–272, http://dx.doi.org/10.1109/EURMIC.2003.1231600.

[33] Felix H Bachmann, Luis Carballo, James McHale, Robert L Nord, Integrate end to end early and often, IEEE Softw. 30 (4) (2013) 9–14.

[34] Armin Balalaie, Abbas Heydarnoori, Pooyan Jamshidi, Migrating to cloud-native architectures using microservices: An experience report, in: Antonio Celesti, Philipp Leitner (Eds.), Advances in Service-Oriented and Cloud Computing, Springer International Publishing, ISBN: 978-3-319-33313-7, 2016, pp. 201–215, http://dx.doi.org/10.1007/978-3-319-33313-7_15.

[36] P. Baldan, A. Corradini, F. Gadducci, Specifying and verifying UML activity diagrams via graph transformation, in: C. Priami, P. Quaglia (Eds.), Global Computing, in: Lecture Notes in Computer Science, Vol. 3267, SPRINGER-VERLAG BERLIN, 2005, pp. 18–33, http://dx.doi.org/10.1109/ICSEA.2007.36.

[37] Soon K. Bang, Sam Chung, Young Choh, Marc Dupuis, A grounded theory analysis of modern web applications: Knowledge, skills, and abilities for DevOps, in: Proceedings of the 2nd Annual Conference on Research in Information Technology, in: RIIT '13, Association for Computing Machinery, ISBN: 9781450324946, 2013, pp. 61–62, http://dx.doi.org/10.1145/2512209.2512229.

[39] Victor R. Basili, V.R.B.G Caldiera, H.D. Rombach, The goal question metric approach, in: John J. Marciniak (Ed.), Encyclopedia of Software Engineering, Vol. 2, Wiley-Interscience, 1994, pp. 528–532, 10.1.1.104.8626.

[40] L. Bass, P. Clements, R. Kazman, Software Architecture in Practice, Addison-Wesley Professional, 2003.

[41] Len Bass, Ingo Weber, Liming Zhu, DevOps: A Software Architect's Perspective, first ed., Addison-Wesley Professional, 2015.

[42] H. Baumeister, Combining formal specifications with test driven development, in: C. Zannier, H. Erdogmus, L. Lindstrom (Eds.), Extreme Programming and Agile Methods - Xp/ Agile Universe 2004, Proceedings, in: Lecture Notes in Computer Science, Vol. 3134, SPRINGER-VERLAG BERLIN, 2004, pp. 1–12.

[43] Anil Bazaz, James D. Arthur, Joseph G. Tront, Modeling security vulnerabilities: A constraints and assumptions perspective, in: 2006 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing, IEEE, 2006, pp. 95–102.

[44] Kent Beck, A theory of programming, Dr. Dobb's J. 21 (2007).

[45] Kent Beck, Extreme Programming: Embracing Change, Vol. 28, (C) 2014, p. 1997.

[47] Kent Beck, Cynthia Andres, Extreme Programming Explained: Embrace Change, Second ed., ISBN: 0-321-27865-8, 2004, p. 224.

[48] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas, Manifesto for agile software development twelve principles of agile software, 2001.

[50] Stephany Bellomo, Robert L. Nord, Ipek Ozkaya, A study of enabling factors for rapid fielding: Combined practices to balance speed and stability, in: David Notkin, Betty H.C. Cheng Michigan State University, Klaus Pohl (Eds.), Proceedings of the 2013 International Conference on Software Engineering, in: ICSE '13, IEEE Press, ISBN: 9781467330763, 2013, pp. 982–991.

[51] Stephany Bellomo, Neil Ernst, Robert L Nord, Ipek Ozkaya, Evolutionary improvements of cross-cutting concerns: Performance in practice, in: Software Maintenance and Evolution (ICSME), 2014 IEEE International Conference on, IEEE, 2014, pp. 545–548.

[52] Herbert D. Benington, Production of large computer programs, IEEE Ann. History Comput. 5 (4) (1983) 350–361.

[53] Elizabeth Bjarnason, Michael Unterkalmsteiner, Markus Borg, Emelie Engstrom, A multi-case study of agile requirements engineering and the use of test cases as requirements, Inf. Softw. Technol. (ISSN: 0950-5849) 77 (2016) 61–79, http://dx.doi.org/10.1016/j.infsof.2016.03.008.

[54] Stuart Blair, Richard Watt, Tim Cull, Responsibility-driven architecture, Ieee Softw. (ISSN: 0740-7459) 27 (2) (2010) 26–32, http://dx.doi.org/10.1109/MS.2010.52.

[59] Gilberto Borrego, Alberto L. Morán, Ramón R. Palacio, Aurora Vizcaíno, Félix O. García, Towards a reduction in architectural knowledge vaporization during agile global software development, Inf. Softw. Technol. (ISSN: 0950-5849) 112 (2019) 68–82, http://dx.doi.org/10.1016/j.infsof.2019.04.008.

[60] Jan Bosch, Continuous Software Engineering: An Introduction, Vol. 9783319112, Springer, 2014, pp. 3–13, http://dx.doi.org/10.1007/978-3-319-11283-1-1.

[61] Gustav Boström, Jaana Wäyrynen, Marine Bodén, Konstantin Beznosov, Philippe Kruchten, Extending XP practices to support security requirements engineering, in: Proceedings of the 2006 International Workshop on Software Engineering for Secure Systems, in: SESS '06, ACM, 2006, pp. 11–18, http://dx.doi.org/10.1145/1137627.1137631.

[62] Hamid Bouabid, Revisiting citation aging: a model for citation distribution and life-cycle prediction, Scientometrics (ISSN: 1588-2861) 88 (1) (2011) 199, http://dx.doi.org/10.1007/s11192-011-0370-5.

[63] K. Boukhelfa, F. Belala, A. Choutri, H. Douibi, For more understandable UML diagrams, in: ACS/IEEE International Conference on Computer Systems and Applications - AICCSA 2010, 2010, pp. 1–7, http://dx.doi.org/10.1109/AICCSA.2010.5586987.

[64] James Bowman, Continuous delivery tool landscape, 2017.

[65] Eric Braude, Incremental UML for agile development: Embedding UML class models in source code, in: Proceedings - 2017 IEEE/ACM 3rd International Workshop on Rapid Continuous Software Engineering, RCoSE 2017, in: RCoSE '17, IEEE Press, 2017, pp. 27–31, http://dx.doi.org/10.1109/RCoSE.2017.1.

[66] Susanne Braun, Frank Elberzhager, Konstantin Holl, Automation support for mobile app quality assurance - a tool landscape, in: Procedia Computer Science, Vol. 110, 2017, pp. 117–124, http://dx.doi.org/10.1016/j.procs.2017.06.129.

[67] Pearl Brereton, Barbara Kitchenham, David Budgen, Mark Turner, Mohamed Khalil, Lessons from applying the systematic literature review process within the software engineering domain, J. Syst. Softw. (ISSN: 01641212) 80 (4) (2007) 571–583, http://dx.doi.org/10.1016/j.jss.2006.07.009.

[69] Frederick P. Brooks Jr., The Mythical Man-Month (Anniversary Ed.), Addison-Wesley Longman Publishing Co., Inc., 1995.

[70] Nanette Brown, Robert Nord, Ipek Ozkaya, Enabling agility through architecture, Tech. rep., Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, 2010.

[71] Nanette Brown, Robert L. Nord, Ipek Ozkaya, Manuel Pais, Analysis and management of architectural dependencies in iterative release planning, in: Proceedings of the 2011 Ninth Working IEEE/IFIP Conference on Software Architecture, in: WICSA '11, IEEE Computer Society, 2011, pp. 103–112, http://dx.doi.org/10.1109/WICSA.2011.22.

[72] Simon Brown, Software Architecture for Developers Technical Leadership By Coding, Coaching, Collaboration, Architecture Sketching and Just Enough Up Front Design, Vol. 7, (1–6) Leanpub, 2014.

[76] Amadeu Silveira Campanelli, Fernando Silva Parreiras, Agile methods tailoring - a systematic literature review, J. Syst. Softw. (ISSN: 01641212) 110 (2015) 85–100, http://dx.doi.org/10.1016/j.jss.2015.08.035.

[80] Rafael Capilla, Anton Jansen, Antony Tang, Paris Avgeriou, Muhammad Ali Babar, 10 years of software architecture knowledge management: Practice and future, J. Syst. Softw. (ISSN: 01641212) 116 (SI) (2016) 191–205, http://dx.doi.org/10.1016/j.jss.2015.08.054.

[81] William B. Cavnar, John M. Trenkle, N-gram-based text categorization, in: Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval, Las Vegas NV Nevada Univ (United States), Las Vegas, US, 1994, pp. 161–175.

[86] Lianipng Chen, Muhammad Ali Babar, He Zhang, Towards evidence-based understanding of electronic data sources, in: EASE'10 Proceedings of the 14th International Conference on Evaluation and Assessment in Software Engineering, 2010, pp. 135–138.

[91] Gerry Gerard Claps, Richard Berntsson Svensson, Aybüke Aurum, On the journey to continuous deployment: Technical and social challenges along the way, Inf. Softw. Technol. 57 (1) (2015) 21–31, http://dx.doi.org/10.1016/j.infsof.2014.07.009.

[92] Kenneth Lee Clarkson, Algorithms for closest-point problems (computational geometry), (Ph.D. thesis), Stanford University, Stanford, CA, USA, 1985.

[93] Jane Cleland-Huang, Safety stories in agile development, IEEE Softw. (ISSN: 07407459) 34 (4) (2017) 16–19, http://dx.doi.org/10.1109/MS.2017.108.

[94] Alistair Cockburn, Agile Software Development: The Cooperative Game, Pearson Education, 2006.

[96] Alistair Cockburn, Writing Effective Use Cases, Addison-Wesley Professional, 2000.

[97] Morris A. Cohen, Jehoshua Eliasberg, Teck-Hua Ho, New product development: The performance and time-to-market tradeoff, Manag. Sci. (ISSN: 0025-1909) 42 (2) (1996) 173–186, http://dx.doi.org/10.1287/mnsc.42.2.173.

[98] Mike Cohn, User Stories Applied: For Agile Software Development, Addison-Wesley Professional, 2004, pp. 230–231.

[100] Software & Systems Engineering Standards Committee, 1061-1998-IEEE standard for a software quality metrics methodology, 1998.

[101] Standards Committee, IEEE Std 1016-2009 (Revision of IEEE Std 1016-1998), IEEE Standard for Information Technology—Systems Design—Software Design Descriptions, Vol. 2009, (July) 2009, pp. c1–40, http://dx.doi.org/10.1109/IEEESTD.2009.5167255.

[102] James O. Coplien, Gertrud Bjrnvig, Lean Architecture for Agile Software Development, Wiley Publishing, 2010, p. 376.

[104] Karina Curcio, Tiago Navarro, Andreia Malucelli, Sheila Reinehr, Requirements engineering, J. Syst. Softw. 139 (C) (2018) 32–50.

[105] Karina Curcio, Tiago Navarro, Andreia Malucelli, Sheila Reinehr, Requirements engineering: A systematic mapping study in agile software development, J. Syst. Softw. (ISSN: 01641212) 139 (2018) 32–50, http://dx.doi.org/10.1016/j.jss.2018.01.036.

[106] Paulo Anselmo Da Mota Silveira Neto, Ivan Do Carmo MacHado, John D. McGregor, Eduardo Santana De Almeida, Silvio Romero De Lemos Meira, A systematic mapping study of software product lines testing, Inf. Softw. Technol. 53 (5) (2011) 407–423, http://dx.doi.org/10.1016/j.infsof.2010.12.003.

[107] Barthélémy Dagenais, Harold Ossher, Rachel KE Bellamy, Martin P Robillard, Jacqueline P De Vries, Moving into a new software project landscape, in: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1, 2010, pp. 275–284.

[108] Subhajit Datta, Proshanta Sarkar, Sutirtha Das, Sonu Sreshtha, Prasanth Lade, Subhashis Majumder, How many eyeballs does a bug need? An empirical validation of linus' law, in: Lecture Notes in Business Information Processing, 179 LNBIP, Springer, 2014, pp. 242–250, http://dx.doi.org/10.1007/978-3-319-06862-6_17.

[116] Philipp Diebold, Marc Dahlem, Agile practices in practice: a mapping study, in: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, ACM, 2014, p. 30.

[117] Gordana Dodig-Crnkovic, Scientific methods in computer science, in: Proceedings of the Conference for the Promotion of Research in IT At New Universities and At University Colleges in Sweden, Skövde, Suecia, 2002, pp. 126–130.

[119] First Public Draft, PRISM : Publishing requirements for industry standard metadata, PRism 2 (2003) 1–95.

[122] T. Dybå, G.R. Bergersen, D.I.K. Sjøberg, B A Kitchenham, T Dyba, M Jorgensen, Evidence-based software engineering, in: Proceedings. 26th International Conference on Software Engineering, 2004, pp. 273–281, http://dx.doi.org/10.1109/ICSE.2004.1317449.

[123] Tore Dybå, Torgeir Dingsøyr, Empirical studies of agile software development: A systematic review, Inf. Softw. Technol. (ISSN: 09505849) 50 (9–10) (2008) 833–859, http://dx.doi.org/10.1016/j.infsof.2008.01.006.

[124] Jessica Díaz, Jennifer Pérez, Juan Garbajosa, Agustín Yagüe, Change-impact driven agile architecting, in: Proceedings of the Annual Hawaii International Conference on System Sciences, E_Informatica, 2013, pp. 4780–4789, http://dx.doi.org/10.1109/HICSS.2013.127.

[125] Christof Ebert, Maria Paasivaara, Scaling agile, IEEE Softw. 34 (6) (2017) 98–103.

[129] Frank Elberzhager, Alla Rosbach, Jürgen Münch, Robert Eschbach, Reducing test effort: A systematic mapping study on existing approaches, Inf. Softw. Technol. (ISSN: 09505849) 54 (10) (2012) 1092–1106, http://dx.doi.org/10.1016/j.infsof.2012.04.007.

[130] IEEE (Institute of Electrical and Electronics Engineers), ISO/IEC/IEEE 24765:2017(E) - systems and software engineering: Vocabulary, in: Iso/Iec/Ieee 24765:2017(E), Vol. 2017, 2017, pp. 1–541, http://dx.doi.org/10.1109/IEEESTD.2017.8016712.

[131] zangemeister Ellis (Ed.), Citation @ dl.acm.org, 1991, http://dx.doi.org/10.1145/2536536.2536556.

[133] Software Engineering and Standards Committee, IEEE Recommended Practice for Software Requirements Speciï£¡cations, Vol. 1998, 1998, p. 37.

[134] Hakan Erdogmus, Architecture meets agility, IEEE Softw. (ISSN: 07407459) 26 (5) (2009) 2–4, http://dx.doi.org/10.1109/MS.2009.121.

[135] Michael Ernst, Choosing a venue : conference or journal?, 2006.

[136] N.A. Ernst, I. Gorton, Using AI to model quality attribute tradeoffs, in: 2014 IEEE 1st International Workshop on Artificial Intelligence for Requirements Engineering (AIRE), 2014, pp. 51–52, http://dx.doi.org/10.1109/AIRE.2014.6894856.

[140] Roland Faber, Architects as service providers, IEEE Softw. (ISSN: 07407459) 27 (2) (2010) 33–40, http://dx.doi.org/10.1109/MS.2010.37.

[141] George Fairbanks, Just Enough Software Architecture: A Risk-Driven Approach, Marshall & Brainerd, 2010.

[142] Matthew E Falagas, Eleni I Pitsouni, George A Malietzis, Georgios Pappas, Comparison of PubMed, scopus, web of science, and google scholar: strengths and weaknesses, FASEB J. (ISSN: 0892-6638) 22 (2) (2007) 338–342, http://dx.doi.org/10.1096/fj.07-9492LSF.

[143] Davide Falessi, Giovanni Cantone, Salvatore Alessandro Sarcia, Giuseppe Calavaro, Paolo Subiaco, Cristiana D'Amore, Peaceful coexistence: Agile developer perspectives on software architecture, IEEE Softw. (ISSN: 07407459) 27 (2) (2010) 23–25, http://dx.doi.org/10.1109/MS.2010.49.

[144] R Farenhorst, J F Hoorn, P Lago, H Van Vliet, R Farenhorst, P Lago, H van Vliet, The lonesome architect, Knowl. Acquis. 84 (9) (2011) 1424–1435.

[146] W.M. Farid, The NORMAP methodology: Lightweight engineering of non-functional requirements for agile processes, in: 2012 19th Asia-Pacific Software Engineering Conference, Vol. 1, 2012, pp. 322–325, http://dx.doi.org/10.1109/APSEC.2012.23.

[147] Weam M. Farid, Frank J. Mitropoulos, Novel lightweight engineering artifacts for modeling non-functional requirements in agile processes, in: Conference Proceedings - IEEE SOUTHEASTCON, 2012, pp. 1–7, http://dx.doi.org/10.1109/SECon.2012.6196988.

[148] Norman E. Fenton, Martin Neil, Software metrics: successes, failures and new directions, J. Syst. Softw. (ISSN: 0164-1212) 47 (2) (1999) 149–157, http://dx.doi.org/10.1016/S0164-1212(99)00035-7.

[150] Brian Fitzgerald, Klaas-Jan Stol, Ryan O'Sullivan, Donal O'Brien, Scaling agile methods to regulated environments: An industry case study, in: 2013 35th International Conference on Software Engineering (ICSE), IEEE, 2013, pp. 863–872.

[151] Frederik M. Fowler, Frederik M. Fowler, The product backlog, in: Frederik M. Fowler (Ed.), Navigating Hybrid Scrum Environments, IEEE Press, 2018, pp. 59–66, http://dx.doi.org/10.1007/978-1-4842-4164-6_9.

[152] Martin Fowler, Specificationbyexample, 2020, https://martinfowler.com/bliki/SpecificationByExample.html (Accessed on 03/24/2020).

[154] Mike Gagliardi, Bill Wood, Uncovering architectural challenges in a system of systems, Tech. rep., Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, 2013.

[157] Jair García, Kelly Garcés, Improving understanding of dynamically typed software developed by agile practitioners, in: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2017, in: ESEC/FSE 2017, ACM, 2017, pp. 908–913, http://dx.doi.org/10.1145/3106237.3117772.

[158] Fabien Gaucher, Yves Génevaux, Debugging embedded systems requirements before the design begins, ACM SIGAda Ada Lett. (ISSN: 10943641) 36 (2) (2017) 58–59, http://dx.doi.org/10.1145/3092893.3092904.

[162] Ron Goldman, +Ron Goldman, Innovation Happens Elsewhere : Open Source As Business Strategy, Morgan Kaufmann, 2005, p. 427.

[163] Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, MIT Press, 2016.

[166] Ibrahim Habli, Tim Kelly, Capturing and replaying architectural knowledge through derivational analogy, in: Second Workshop on Sharing and Reusing Architectural Knowledge-Architecture, Rationale, and Design Intent (SHARK/ADI'07: ICSE Workshops 2007), IEEE, 2007, p. 4.

[168] Neal Robert Haddaway, Alexandra Mary Collins, Deborah Coughlin, Stuart Kirk, The role of google scholar in evidence reviews and its applicability to grey literature searching, in: K. Brad Wray (Ed.), PLoS ONE (ISSN: 19326203) 10 (9) (2015) e0138237, http://dx.doi.org/10.1371/journal.pone.0138237.

[169] Jack Hakim, Tom Spitzer, John Armitage, Sprint: Agile specifications in shockwave and flash, in: Proceedings of the 2003 Conference on Designing for User Experiences - DUX '03, in: DUX '03, ACM, 2003, p. 1, http://dx.doi.org/10.1145/997078.997111.

[170] Niklas Hallberg, Richard Andersson, Christina Ölvander, Agile architecture framework for model driven development of c 2 systems, Syst. Eng. (ISSN: 10981241) 13 (2) (2009) n/a–n/a, http://dx.doi.org/10.1002/sys.20141.

[171] Saba Hamdan, Suad Alramouni, A quality framework for software continuous integration, Proc. Manuf. (ISSN: 23519789) 3 (2015) 2019–2025, http://dx.doi.org/10.1016/j.promfg.2015.07.249.

[174] Mark Harman, Why source code analysis and manipulation will always be important (keynote), in: 10th IEEE International Working Conference on Source Code Analysis and Manipulation, IEEE, 2010, pp. 7–19.

[176] Heena, Ranjna, A comparative study of UML tools, in: Proceedings of the International Conference on Advances in Computing and Artificial Intelligence, in: ACAI '11, ACM, New York, NY, USA, 2011, pp. 1–4, http://dx.doi.org/10.1145/2007052.2007053.

[178] V.T. Heikkilä, D. Damian, C. Lassenius, M. Paasivaara, A mapping study on requirements engineering in agile software development, in: 2015 41st Euromicro Conference on Software Engineering and Advanced Applications, 2015, pp. 199–207, http://dx.doi.org/10.1109/SEAA.2015.70.

[180] Susan C. Herring, Content analysis for new media : Rethinking the paradigm, in: New Research for New Media: Innovative Research Methodologies Symposium Working Papers and Reading, 2004, pp. 47–66, http://dx.doi.org/10.1177/1461444804039906.

[183] R. Hilliard, Systems and software engineering — Architecture description ISO/IEC/IEEE 42010, 2019, http://www.iso-architecture.org/ieee-1471/afs/ (Accessed: 2019-05-11).

[185] Rashina Hoda, Norsaremah Salleh, John Grundy, Hui Mien Tee, Systematic literature reviews in agile software development: A tertiary study, Inf. Softw. Technol. (ISSN: 09505849) 85 (2017) 1339–1351, http://dx.doi.org/10.1016/j.infsof.2017.01.007.

[186] Daniel Hoffman, Paul Strooper, Api documentation with executable examples, 2003, http://dx.doi.org/10.1016/S0164-1212(02)00055-9.

[187] Christine Hofmeister, Philippe Kruchten, Robert L Nord, Henk Obbink, Alexander Ran, Pierre America, A general model of software architecture design derived from five industrial approaches, J. Syst. Softw. 80 (1) (2007) 106–126.

[189] Jeroen van den Hoven, John Weckert, Information Technology and Moral Philosophy, 2008, pp. 1–415, http://dx.doi.org/10.1017/CBO9780511498725.

[190] Shihong Huang, Scott Tilley, Towards a documentation maturity model, in: Susan B. Jones, David G. Novick (Eds.), ACM Special Interest Group for Design. of Commun.; SIGDOC 2003: Finding Real-World Solutions for Doc.: How Theory Informs Pract. and Pract. Informs Theory. Proc. of the 21st Annu. Int. Conf. on Doc., ACM, 2003, pp. 93–99, http://dx.doi.org/10.1145/944868.944888.

[191] Jezz Humble, David Farley, Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation, Addison-Wesley Boston, 2015, p. 497, http://dx.doi.org/10.1007/s13398-014-0173-7.2.

[193] D.M. Hutton, Clean Code: A Handbook of Agile Software Craftsmanship20092Robert C. Martin, in: Clean Code: A Handbook of Agile Software Craftsmanshi, 27, (99) Prentice-Hall, ISBN: 9-780-13235-088-4, 2008, http://dx.doi.org/10.1108/03684920910973252, In: Kybernetes 38.6 (June 2009), pp. 1035–1035.

[194] Michael Hüttermann, DevOps for Developers, APress, 2012, http://dx.doi.org/10.1007/978-1-4302-4570-4.

[198] Iso, Systems and software engineering systems and software quality requirements and evaluation, 2011.

[200] ISO - ISO/IEC/IEEE 42010:2011 - systems and software engineering — architecture description, 2011, https://www.iso.org/standard/50508.html (Accessed on 02/04/2021).

[201] ISO/IEC/IEEE, Systems and software engineering – architecture description, in: ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000), 2011, 2011, pp. 1–46, http://dx.doi.org/10.1109/ieeestd.2011.6129467.

[202] Ramtin Jabbari, Nauman bin Ali, Kai Petersen, Binish Tanveer, What is DevOps? in: Proceedings of the Scientific Workshop Proceedings of XP2016 on - XP '16 Workshops, ACM Press, New York, New York, USA, 2016, pp. 1–11, http://dx.doi.org/10.1145/2962695.2962707.

[203] Péter Jacsó, Google scholar: the pros and the cons, Online Inf. Rev. (ISSN: 1468-4527) 29 (2) (2005) 208–214, http://dx.doi.org/10.1108/14684520510598066.

[204] Samireh Jalali, Claes Wohlin, Systematic literature studies: Database searches vs. Backward snowballing, in: ESEM'12: Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, 2012, pp. 29–38, http://dx.doi.org/10.1145/2372251.2372257.

[205] Anton Jansen, Jan Bosch, Software architecture as a set of architectural design decisions, in: 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05), IEEE, 2005, pp. 109–120.

[208] Andreas Jedlitschka, Pasi Kuvaja, Marco Kuhrmann, Tomi Männistö, JüRgen Münch, Mikko Raatikainen, Product-focused software process improvement: 15th international conference, PROFES 2014 helsinki, Finland, december 10-12, 2014 proceedings, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 8892, (ISSN: 16113349) 2014, p. 27, http://dx.doi.org/10.1007/978-3-319-13835-0.

[209] Ron Jeffries, Essential XP: card, conversation, confirmation, XP Mag. 30 (2001).

[210] Ron Jeffries, Ann Anderson, Chet Hendrickson, Extreme Programming Installed, Addison-Wesley Professional, 2001.

[211] RolfNjor Jensen, et al., Developer stories: Improving architecture in agile practice, in: Software and Data Technologies SE - 13, Vol. 22, Springer, 2009, pp. 172–184, http://dx.doi.org/10.1007/978-3-540-88655-6_13.

[215] L. Jun, W. Qiuzhen, G. Lin, Application of agile requirement engineering in modest-sized information systems development, in: 2010 Second World Congress on Software Engineering, Vol. 2, 2010, pp. 207–210, http://dx.doi.org/10.1109/WCSE.2010.105.

[216] Jose Luis Barros Justo, Nelson Martinez Araujo, Alejandro Gonzalez Garcia, Software reuse and continuous software development: A systematic mapping study, IEEE Latin Am. Trans. 16 (5) (2018) 1539–1546, http://dx.doi.org/10.1109/TLA.2018.8408452.

[218] Mira Kajko-Mattsson, Problems in agile trenches, in: Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '08, in: ESEM '08, ACM, 2008, p. 111, http://dx.doi.org/10.1145/1414004.1414025.

[219] Ahmad Waqas Kamal, Paris Avgeriou, Modeling architectural patterns' behavior using architectural primitives, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 5292 LNCS, (ISSN: 03029743) 2008, pp. 164–179, http://dx.doi.org/10.1007/978-3-540-88030-1_13.

[222] Michael Keeling, Architecture Haiku: A case study in lean documentation, IEEE Softw. (ISSN: 07407459) 32 (3) (2015) 35–39, http://dx.doi.org/10.1109/MS.2015.59.

[223] Sandra Kelly, Towards an evolutionary framework for agile requirements elicitation, in: Proceedings of the 2nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems - EICS '10, in: EICS '10, ACM, New York, NY, USA, 2010, p. 349, http://dx.doi.org/10.1145/1822018.1822078.

[226] Thorsten Keuler, Stefan Wagner, Bernhard Winkler, Architecture-aware programming in agile environments, in: Proceedings of the 2012 Joint Working Conference on Software Architecture and 6th European Conference on Software Architecture, WICSA/ECSA 2012, IEEE, 2012, pp. 229–233, http://dx.doi.org/10.1109/WICSA-ECSA.212.35.

[228] B. Kitchenham, P. Brereton, M. Turner, M. Niazi, S. Linkman, R. Pretorius, D. Budgen, The impact of limited search procedures for systematic literature reviews — A participant-observer case study, in: 2009 3rd International Symposium on Empirical Software Engineering and Measurement, 2009, pp. 336–345, http://dx.doi.org/10.1109/ESEM.2009.5314238.

[229] Barbara Kitchenham, Stuart Charters, Guidelines for performing systematic literature reviews in software engineering, Engineering (ISSN: 00010782) 2 (4ve) (2007) 1051, http://dx.doi.org/10.1145/1134285.1134500.

[230] Barbara Kitchenham, Lesley Pickard, Shari Lawrence Pfleeger, Case studies for method and tool evaluation, IEEE Softw. (ISSN: 07407459) 12 (4) (1995) 52–62, http://dx.doi.org/10.1109/52.391832.

[231] Barbara Kitchenham, Rialette Pretorius, David Budgen, O. Pearl Brereton, Mark Turner, Mahmood Niazi, Stephen Linkman, Systematic literature reviews in software engineering-a tertiary study, 2010, http://dx.doi.org/10.1016/j.infsof.2010.03.006.

[232] Barbara A. Kitchenham, David Budgen, O. Pearl Brereton, Using mapping studies as the basis for further research–a participant-observer case study, Inf. Softw. Technol. 53 (6) (2011) 638–651.

[233] Barbara A Kitchenham, O Pearl Brereton, David Budgen, Zhi Li, An evaluation of quality checklist proposals: a participant-observer case study, in: Proceedings of the 13th International Conference on Evaluation and Assessment in Software Engineering, in: EASE'09, BCS Learning & Development Ltd., 2009, pp. 55–64.

[234] Barbara A. Kitchenham, Shari Lawrence Pfleeger, Lesley M. Pickard, Peter W. Jones, David C. Hoaglin, Khaled El Emam, Jarrett Rosenberg, Preliminary guidelines for empirical research in software engineering, IEEE Trans. Softw. Eng. (ISSN: 00985589) 28 (8) (2002) 721–734, http://dx.doi.org/10.1109/TSE.2002.1027796.

[235] Barbara A Kitchenham, Pearl Brereton, Mark Turner, Mahmood K Niazi, Stephen Linkman, Rialette Pretorius, David Budgen, Refining the systematic literature review process–two participant-observer case studies, Empirical Softw. Eng. 15 (6) (2010) 618–653.

[236] E. Knauss, K. Schneider, K. Stapel, A game for taking requirements engineering more seriously, in: 2008 Third International Workshop on Multimedia and Enjoyable Requirements Engineering - beyond Mere Descriptions and with more Fun and Games, 2008, pp. 22–26, http://dx.doi.org/10.1109/MERE.2008.1.

[238] Donald E. Knuth, Seminumerical Algorithms, second ed., The Art of Computer Programming, Vol. 2, Addison-Wesley, 1981.

[239] Ronald J. Koontz, Robert L. Nord, Architecting for sustainable software delivery, Tech. rep., Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, 2012.

[241] D.A. Kosower, J.J. Lopez-Villarejo, S. Roubtsov, Flowgen: Flowchart-based documentation framework for c++, in: 2014 IEEE 14th International Working Conference on Source Code Analysis and Manipulation, 2014, pp. 59–64, http://dx.doi.org/10.1109/SCAM.2014.35.

[244] Philippe Kruchten, An ontology of architectural design decisions in software-intensive systems, in: Proceedings of the 2nd Groningen Workshop on Software Variability, 2004, pp. 1–8.

[245] Philippe Kruchten, The Rational Unified Process: An Introduction, Addison-Wesley Professional, 2004.

[247] Jaroslav Král, Michal emlika, Requirements specification: What strategy under what conditions, in: Proceedings - SERA 2007: Fifth ACIS International Conference on Software Engineering Research, Management, and Applications, 2007, pp. 401–408, http://dx.doi.org/10.1109/SERA.2007.112.

[249] Manish Kumar, Nirav Ajmeri, Smita Ghaisas, Towards knowledge assisted agile requirements evolution, in: Proceedings of the 2Nd International Workshop on Recommendation Systems for Software Engineering, in: RSSE '10, ACM, New York, NY, USA, 2010, pp. 16–20, http://dx.doi.org/10.1145/1808920.1808924.

[251] Patricia Lago, Hans Van Vliet, Explicit assumptions enrich architectural models, in: Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005, IEEE, 2005, pp. 206–214.

[252] Craig Larman, V.R. Basili, Iterative and incremental developments. a brief history, Computer (ISSN: 0018-9162) 36 (6) (2003) 47–56, http://dx.doi.org/10.1109/MC.2003.1204375.

[253] Thomas D. LaToza, Gina Venolia, Robert DeLine, Maintaining mental models, in: Proceeding of the 28th International Conference on Software Engineering - ICSE '06, in: ICSE '06, ACM, New York, NY, USA, 2006, p. 492, http://dx.doi.org/10.1145/1134285.1134355.

[254] Claire Le Goues, Shin Yoo, Search-based software engineering 6th international symposium, ssbse 2014, fortaleza, Brazil, august 26-29, 2014. Proceedings, in: Conference Proceedings SSBSE, Springer, 2014, p. 164.

[256] Zengyang Li, Paris Avgeriou, Peng Liang, A systematic mapping study on technical debt and its management, J. Syst. Softw. (ISSN: 01641212) 101 (2015) 193–220, http://dx.doi.org/10.1016/j.jss.2014.12.027.

[257] Zengyang Li, Peng Liang, Paris Avgeriou, Application of knowledge-based approaches in software architecture: A systematic mapping study, 2013, http://dx.doi.org/10.1016/j.infsof.2012.11.005.

[259] Olga Liskin, Kurt Schneider, Fabian Fagerholm, Jürgen Münch, Understanding the role of requirements artifacts in kanban, in: Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering, in: CHASE 2014, ACM, New York, NY, USA, 2014, pp. 56–63, http://dx.doi.org/10.1145/2593702.2593707.

[260] Gloria Yolanda Lopez Herrera, Juan Carlos Jimenez Sanz, Best practices for requirements identification, specification, and validation to guide software implementation and maintenance processes for applications in an electricity supply company, Sistemas Telem. (ISSN: 1692-5238) 13 (35) (2015) 53–76, http://dx.doi.org/10.18046/syt.v13i35.2152.

[261] Martin Lopez-Nores, Jose J Pazos-Arias, Jorge Garcia-Duque, Yolanda Blanco-Fernandez, Rebeca P Diaz-Redondo, Ana Fernandez-Vilas, Alberto Gil-Solla, Manuel Ramos-Cabrer, Bringing the agile philosophy to formal specification settings, Int. J. Softw. Eng. Knowl. Eng. (ISSN: 0218-1940) 16 (6) (2006) 951–986, http://dx.doi.org/10.1142/S0218194006003075.

[262] Lucy Ellen Lwakatare, Pasi Kuvaja, Markku Oivo, Relationship of DevOps to agile, lean and continuous deployment, in: International Conference on Product-Focused Software Process Improvement, Springer, 2016, pp. 399–415, http://dx.doi.org/10.1007/978-3-319-49094-6_27.

[264] James Madison, Agile architecture interactions, IEEE Softw. (ISSN: 07407459) 27 (2) (2010) 41–48, http://dx.doi.org/10.1109/MS.2010.35.

[266] N. Maiden, S. Jones, Agile requirements can we have our cake and eat it too?, IEEE Softw. (ISSN: 0740-7459) 27 (3) (2010) 87–88, http://dx.doi.org/10.1109/MS.2010.67.

[270] Phyllis Marbach, Larri Rosser, Gundars Osvalds, David Lempia, Principles for agile development, INCOSE Int. Symp. 25 (1) (2015) 524–537, http://dx.doi.org/10.1002/j.2334-5837.2015.00079.x.

[277] Matthew B. Miles, A. Michael Huberman, Qualitative Data Analysis: An Expanded Sourcebook, 2nd ed., Sage Publications, Inc, Thousand Oaks, CA, US, 1994, p. xiv,338.

[278] Albert J. Mills, Gabrielle Durepos, Elden Wiebe, Exploratory Case Study in: Encyclopedia of Case Study Research, Vol. 1, SAGE Publications, Inc, 2010, http://dx.doi.org/10.4135/9781412957397.

[280] ABDULMAJID HISSEN Mohamed, Capturing software-engineering tacit knowledge, in: Proceedings of the 2nd Conference on European Computing Conference (ECC'08:), 2008.

[281] Diljith Muthuvana, Renuka Prasad, Agility in systems engineering, Tech. rep., SAE Technical Paper, 2015, http://dx.doi.org/10.4271/2015-01-0144.

[283] Simo Mäkinen, Marko Leppänen, Terhi Kilamo, Anna-Liisa Mattila, Eero Laukkanen, Max Pagels, Tomi Männistö, Improving the delivery cycle: A multiple-case study of the toolchains in finnish software intensive enterprises, Inf. Softw. Technol. 80 (2016) 175–194.

[285] Sridhar Nerur, VenuGopal Balijepally, Theoretical reflections on agile development methodologies, Commun. ACM (ISSN: 00010782) 50 (3) (2007) 79–83, http://dx.doi.org/10.1145/1226736.1226739.

[286] Oscar Nierstrasz, Jan Kurš, Parsing for agile modeling, Sci. Comput. Prog. (ISSN: 01676423) 97 (P1) (2015) 150–156, http://dx.doi.org/10.1016/j.scico.2013.11.011.

[287] Nan Niu, Re in the age of continuous deployment, in: Proceedings - 2017 IEEE 25th International Requirements Engineering Conference, RE 2017, 2017, pp. 568–569, http://dx.doi.org/10.1109/RE.2017.89.

[288] Robert L. Nord, Nanette Brown, Ipek Ozkaya, Architecting with just enough information, in: Proceedings of the 6th International Workshop on SHAring and Reusing Architectural Knowledge, 2011, pp. 9–12.
[289] Robert L. Nord, James McHale, Felix Bachmann, Combining architecture-centric engineering with the team software process, Tech. rep., Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, 2010.
[293] Robert L Nord, Ipek Ozkaya, Raghvinder S Sangwan, Julien Delange, Marco González, Philippe Kruchten, Variations on using propagation cost to measure architecture modifiability properties, in: Proceedings of the 2013 IEEE International Conference on Software Maintenance, in: ICSM '13, IEEE Computer Society, Washington, DC, USA, 2013, pp. 400–403, http://dx.doi.org/10.1109/ICSM.2013.57.
[294] Dave Novak, Solder man, in: ACM SIGGRAPH 2003 Video Review on Animation Theater Program: Part I - Vol. 145 (July 27–27, 2003), ACM Press, 2003, p. 4, 99.9999/woot07-S422.
[295] A. Ochsner, Introduction to Scientific Publishing, Springer, 2013.
[297] Helena Holmstrom Olsson, Hiva Alahyari, Jan Bosch, Climbing the "stairway to heaven" – a mulitiple-case study exploring barriers in the transition from agile development towards continuous deployment of software, in: Proceedings of the 2012 38th Euromicro Conference on Software Engineering and Advanced Applications, in: SEAA '12, IEEE Computer Society, Washington, DC, USA, 2012, pp. 392–399, http://dx.doi.org/10.1109/SEAA.2012.54.
[298] Omg, Semantics of business vocabulary and business rules, 1.2, (May) 2013, p. 422.
[299] K. Orr, Agile requirements: opportunity or oxymoron?, IEEE Softw. (ISSN: 0740-7459) 21 (3) (2004) 71–73, http://dx.doi.org/10.1109/MS.2004.1293075.
[300] Ipek Ozkaya, Achieving agility and stability in large-scale software development, Tech. rep., Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, 2013.
[301] Ipek Ozkaya, Michael Gagliardi, Robert L. Nord, Architecting for large scale agile software development: A risk-driven approach, Tech. rep., Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, 2013.
[302] Stephany Bellomo Ozkaya, Ipek Robert L. Nord, Heidi Brayer, Beyond scrum + XP: Agile architecture practice, Tech. rep., (6) Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, 2013, pp. 6–9.
[303] Julian Padget, Emad Eldeen Elakehal, Ken Satoh, Fuyuki Ishikawa, On requirements representation and reasoning using answer set programming, in: 2014 IEEE 1st International Workshop on Artificial Intelligence for Requirements Engineering, AIRE 2014 - Proceedings, 2014, pp. 35–42, http://dx.doi.org/10.1109/AIRE.2014.6894854.
[306] David Lorge Parnas, Software aging, in: Taylor Richard N., Coutaz Joëlle (Eds.), Proceedings of the 16th International Conference on Software Engineering, in: ICSE '94, IEEE Computer Society Press, ISBN: 081865855X, 1994, pp. 279–287.
[307] David Lorge Parnas, Paul C. Clements, A rational design process: How and why to fake it, Trans. Softw. Eng. (ISSN: 0098-5589) 12 (2) (1986) 251–257, http://dx.doi.org/10.1007/3-540-15199-0_6.
[308] Mark Paulk, Capability maturity model for software, Encycl. Softw. Eng. (2002) http://dx.doi.org/10.1002/0471028959.sof589.
[310] Dewayne E. Perry, Alexander L. Wolf, Foundations for the study of software architecture, ACM SIGSOFT Softw. Eng. Notes 17 (4) (1992) 40–52.
[311] Kai Petersen, Cigdem Gencel, Worldviews, research methods, and their relationship to validity in empirical software engineering research, in: Proceedings of the 2013 Joint Conference of the 23nd International Workshop on Software Measurement (IWSM) and the 8th International Conference on Software Process and Product Measurement, IEEE Computer Society, 2013, pp. 81–89, http://dx.doi.org/10.1109/IWSM-Mensura.2013.22.
[312] Kai Petersen, Sairam Vakkalanka, Ludwik Kuzniarz, Guidelines for conducting systematic mapping studies in software engineering: An update, 64, 2015, pp. 1–18, http://dx.doi.org/10.1016/j.infsof.2015.03.007,
[313] Kai Petersen, Robert Feldt, Shahid Mujtaba, Michael Mattsson, Systematic mapping studies in software engineering, (ISSN: 02181940) 17 (2008) 10, http://dx.doi.org/10.1142/S0218194007003112.
[314] Charles J. Petrie, New algorithms for dependency-directed backtracking, (Master's thesis) (Ph.D. thesis), University of Texas at Austin, Austin, TX, USA, 1986.
[315] M Pikkarainen, J Haikara, O Salo, P Abrahamsson, J Still, The impact of agile practices on communication in software development, Empirical Softw. Eng. (ISSN: 13823256) 13 (3) (2008) 303–337, http://dx.doi.org/10.1007/s10664-008-9065-9.
[317] Mary Poppendieck, Tom Poppendieck, Lean software development: an agile toolkit, Computer (ISSN: 0018-9162) 36 (8) (2003) 89, http://dx.doi.org/10.1109/MC.2003.1220585.
[319] Alexander Poth, Mark Werner, Xinyan Lei, How to deliver faster with CI/CD integrated testing services? in: European Conference on Software Process Improvement, Springer, 2018, pp. 401–409, http://dx.doi.org/10.1007/978-3-319-97925-0_33.
[321] Christian R. Prause, Reputation-based self-management of software process artifact quality in consortium research projects, in: Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering - SIGSOFT/FSE '11, in: ESEC/FSE '11, ACM, New York, NY, USA, 2011, p. 380, http://dx.doi.org/10.1145/2025113.2025166.
[326] K. Read, F. Maurer, Issues in scaling agile using an architecture-centric approach: A tool-based solution, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2753, Springer, 2003, pp. 142–150, http://dx.doi.org/10.1007/978-3-540-45122-8_16.
[327] M.J. Rees, A feasible user story tool for agile software development? in: Proceedings - Asia-Pacific Software Engineering Conference, APSEC, 2002-Janua, 2002, pp. 22–30, http://dx.doi.org/10.1109/APSEC.2002.1182972.
[328] Pilar Rodríguez, Alireza Haghighatkhah, Lucy Ellen Lwakatare, Susanna Teppola, Tanja Suomalainen, Juho Eskeli, Teemu Karvonen, Pasi Kuvaja, June M. Verner, Markku Oivo, Continuous deployment of software intensive products and services: A systematic mapping study, J. Syst. Softw. (ISSN: 01641212) 123 (2017) 263–291, http://dx.doi.org/10.1016/j.jss.2015.12.015.
[329] Bernice E. Rogowitz, Lloyd A. Treinish, Steve Bryson, How not to lie with visualization, Comput. Phys. (ISSN: 0894-1866) 10 (3) (1996) 268–273.
[331] Rasmus Ros, Elizabeth Bjarnason, Per Runeson, A machine learning approach for semi-automated search and selection in literature studies, in: Emilia Mendes, Steve Counsell, Kai Petersen (Eds.), EASE'17: Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering, Association for Computing Machinery, ACM, Karlskrona, Sweden, ISBN: 9781450348041, 2017, pp. 118–127, http://dx.doi.org/10.1145/3084226.
[333] Babak Darvish Rouhani, Hossein Shirazi, All Farahmand Nezhad, Sadegh Kharazmi, Presenting a framework for agile enterprise architecture, in: Proceedings of the 2008 1st International Conference on Information Technology, IT 2008, IEEE, 2008, pp. 1–4, http://dx.doi.org/10.1109/INFTECH.2008.4621684.
[335] B.H. Rudall, Lecture Notes in Computer Science. Vol. 15—L-Systems, Vol. 9, (3) 1978, p. 242, http://dx.doi.org/10.1016/0020-7101(78)90038-7.
[336] Per Runeson, Martin Höst, Guidelines for conducting and reporting case study research in software engineering, Empirical Softw. Eng. (ISSN: 1382-3256) 14 (2) (2009) 131–164, http://dx.doi.org/10.1007/s10664-008-9102-8.
[338] N. Saher, F. Baharom, O. Ghazali, Requirement change taxonomy and categorization in agile software development, in: 2017 6th International Conference on Electrical Engineering and Informatics (ICEEI), 2017, pp. 1–6, http://dx.doi.org/10.1109/ICEEI.2017.8312441.
[342] Dina Salah, Richard F. Paige, Paul Cairns, A systematic literature review for agile development processes and user centred design integration, in: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE '14, (2006) British Computer Society, 2014, pp. 1–10, http://dx.doi.org/10.1145/2601248.2601276.
[343] M. Salama, R. Bahsoon, N. Bencomo, Managing trade-offs in self-adaptive software architectures, in: Managing Trade-Offs in Adaptable Software Architectures, Elsevier, 2017, pp. 249–297, http://dx.doi.org/10.1016/B978-0-12-802855-1.00011-3.
[345] Deki Satria, Dana Indra Sensuse, Handrie Noprisson, A systematic literature review of the improved agile software development, in: International Conference on Information Technology Systems and Innovation (ICITSI), 2017, pp. 23–24, http://dx.doi.org/10.1109/ICITSI.2017.8267925.
[347] Gerald Schermann, Jürgen Cito, Philipp Leitner, Uwe Zdun, Harald C. Gall, We're doing it live: A multi-method empirical study on continuous experimentation, 2018, http://dx.doi.org/10.1016/j.infsof.2018.02.010.
[348] Kurt Schneider, Kai Stapel, Eric Knauss, Beyond documents: Visualizing informal communication, in: Proceedings of the 2008 Requirements Engineering Visualization, IEEE Computer Society, 2008, pp. 31–40, http://dx.doi.org/10.1109/REV.2008.1.
[351] Eva Maria Schön, Jörg Thomaschewski, María José Escalona, Agile requirements engineering: A systematic literature review, Comput. Stand. Interfaces (ISSN: 09205489) 49 (2017) 79–91, http://dx.doi.org/10.1016/j.csi.2016.08.011.
[353] Bran Selic, Agile documentation, anyone?, IEEE Softw. (ISSN: 07407459) 26 (6) (2009) 11–12, http://dx.doi.org/10.1109/MS.2009.167.
[354] Sara Shafiee, Lars Hvam, Anders Haug, Michael Dam, Katrin Kristjansdottir, The documentation of product configuration systems: A framework and an IT solution, Adv. Eng. Inf. (ISSN: 14740346) 32 (2017) 163–175, http://dx.doi.org/10.1016/j.aei.2017.02.004.
[355] Muneeb Shafiq, U. sman Waheed, Documentation in agile development a comparative analysis, in: 2018 IEEE 21st International Multi-Topic Conference (INMIC), IEEE, 2018, pp. 1–8, http://dx.doi.org/10.1109/inmic.2018.8595625.
[357] Helen Sharp, Hugh Robinson, Marian Petre, The role of physical artefacts in agile software development: Two complementary perspectives, Interacting Comput. (ISSN: 09535438) 21 (1–2) (2009) 108–116, http://dx.doi.org/10.1016/j.intcom.2008.10.006.
[358] Mary Shaw, Writing good software engineering research papers: minitutorial, in: Proceedings of the 25th International Conference on Software Engineering, IEEE Computer Society, 2003, pp. 726–736, http://dx.doi.org/10.1007/s10009-002-0083-4.
[360] Ofira Shmueli, Boaz Ronen, Excessive software development: Practices and penalties, Int. J. Project Manag. (ISSN: 02637863) 35 (1) (2017) 13–27, http://dx.doi.org/10.1016/j.ijproman.2016.10.002.
[362] A. Sillitti, M. Ceschi, B. Russo, G. Succi, Managing uncertainty in requirements: a survey in documentation-driven and agile companies, in: 11th IEEE International Software Metrics Symposium (METRICS'05), 2005, pp. 10 pp.–17, http://dx.doi.org/10.1109/METRICS.2005.29.

[363] SOA reference architecture – integration layer, 2020, https://www.opengroup.org/soa/source-book/soa_refarch/p13.htm (Accessed on 03/28/2020).
[365] Charalampidou Sofia, Ampatzoglou Apostolos, Karountzos Evangelos, Avgeriou Paris, Empirical studies on software traceability: A mapping study, J. Softw.: Evol. Process 33 (2) (2020) http://dx.doi.org/10.1002/smr.2294.
[366] Edgewall Software, Trac, 2012, ${$}http://trac.edgewall.org/.
[367] Rini van Solingen, Vic Basili, Gianluigi Caldiera, H. Dieter Rombach, Goal question metric (GQM) approach, Encycl. Softw. Eng. (2002) http://dx.doi.org/10.1002/0471028959.sof142.
[368] Stephan Sonnenburg, Creativity in communication: A theoretical framework for collaborative product creation, Creativ. Innov. Manag. 13 (4) (2004) 254–262.
[369] Amina Souag, Raúl Mazo, Camille Salinesi, Isabelle Comyn-Wattiau, Reusable knowledge in security requirements engineering: a systematic mapping study, Requir. Eng. 21 (2) (2016) 251–283, http://dx.doi.org/10.1007/s00766-015-0220-8.
[370] Shvetha Soundararajan, James D. Arthur, A soft-structured agile framework for larger scale systems development, in: Proceedings of the International Symposium and Workshop on Engineering of Computer Based Systems, 2009, pp. 187–195, http://dx.doi.org/10.1109/ECBS.2009.21.
[374] Matt Stephens, The Case Against Extreme Programming, APress, 2002, pp. 1–20, http://dx.doi.org/10.1007/978-1-4302-0810-5.
[378] Karsten Stocker, Hironori Washizaki, Yoshiaki Fukazawa, Closing the gap between unit test code and documentation, in: 2017 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2017, pp. 304–308, http://dx.doi.org/10.1109/ICSTW.2017.56.
[380] Daniel Ståhl, Jan Bosch, Modeling continuous integration practice differences in industry software development, J. Syst. Softw. (ISSN: 01641212) 87 (1) (2014) 48–59, http://dx.doi.org/10.1016/j.jss.2013.08.032.
[383] Antony Tang, Jun Han, Pin Chen, MENDELEY_ID Software Engineering Conference, 2004. 11th Asia-Pacific, Technical Report: SUTIT-TR2004.01, in: A Comparative Analysis of Architecture Frameworks, Swinburne University of Technology, 2004, pp. 640–647.
[384] Antony Tang, Peng Liang, Hans Van Vliet, Software architecture documentation: The road ahead, in: 2011 Ninth Working IEEE/IFIP Conference on Software Architecture, IEEE, 2011, pp. 252–255.
[386] T Tenso, A H Norta, H Rootsi, K Taveter, I Vorontsova, Enhancing requirements engineering in agile methodologies by agent-oriented goal models: Two empirical case studies, in: 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW), 2017, pp. 268–275, http://dx.doi.org/10.1109/REW.2017.24.
[387] Theo Theunissen, Uwe van Heesch, The disappearance of technical specifications in web and mobile applications, in: Software Architecture: 10th European Conference, ECSA 2016, Copenhagen, Denmark, November 28–December 2, 2016, Proceedings 10, Vol. 9839 LNCS, Springer, Cham, 2016, pp. 265–273, http://dx.doi.org/10.1007/978-3-319-48992-6_20.
[388] Theo Theunissen, Stijn Hoppenbrouwers, Sietse Overbeek, In continuous software development, tools are the message for documentation, in: Proceedings of the 23rd International Conference on Enterprise Information Systems, SCITEPRESS - Science and Technology Publications, 2021, http://dx.doi.org/10.5220/0010367901530164.
[391] Dan Tofan, Matthias Galster, Paris Avgeriou, Wes Schuitema, Past and future of software architectural decisions - a systematic mapping study, 2014, http://dx.doi.org/10.1016/j.infsof.2014.03.009.
[392] Adam Trendowicz, Jürgen Münch, Factors influencing software development productivity - state-of-the-art and industrial experiences, Adv. Comput. (ISSN: 00652458) 77 (2009) 185–241, http://dx.doi.org/10.1016/s0065-2458(09)01206-6.
[393] Wen Lung Tsai, Chung Yang Chen, Chun Shuo Chen, Snowman: Agile development method with institutionalized communication and documentation for capstone projects, Asia Pacific Manag. Rev. (ISSN: 10293132) 23 (1) (2018) 12–19, http://dx.doi.org/10.1016/j.apmrv.2017.01.002.
[394] J. Tyree, A. Akerman, Architecture decisions: Demystifying architecture, IEEE Softw. (ISSN: 0740-7459) 2 (2005) http://dx.doi.org/10.1109/MS.2005.27, 22no22005pp19–27.
[397] Timur R. Usmanov, Nargiz I. Bagmanova, Rashida F. Vafina, Kazan imperial university - socio-pedagogical space of legal education formation in the kazan province of the early 19th century, in: Man in India, 97, (8) ACM, 2017, pp. 275–281, http://dx.doi.org/10.1145/2361999.2362023.
[399] Raoul Vallon, Bernardo José da Silva Estácio, Rafael Prikladnicki, Thomas Grechenig, Systematic literature review on agile practices in global software development, Inf. Softw. Technol. (ISSN: 09505849) 96 (2017) 161–180, http://dx.doi.org/10.1016/j.infsof.2017.12.004.
[401] U. Van Heesch, P. Avgeriou, A. Tang, Does decision documentation help junior designers rationalize their decisions? A comparative multiple-case study, J. Syst. Softw. (ISSN: 01641212) 86 (6) (2013) 1545–1565, http://dx.doi.org/10.1016/j.jss.2013.01.057.
[403] Crystian Sadiel Venegas-Barrera, Javier Manjarrez, Patrones Espaciales de la Riqueza Espec??Fica de Las Culebras Thamnophis En M??Xico, vol. 82, (1) 2011, pp. 179–191, http://dx.doi.org/10.1234/12345678.
[404] Vaughn Vernon, Implementing Domain-Driven Design, Addison-Wesley Professional, 2013.
[412] Hairunnizam Wahid, Sanep Ahmad, Mohd Ali Mohd Nor, Maryam Abd Rashid, Prestasi Kecekapan Pengurusan Kewangan Dan Agihan Zakat: Perbandingan Antara Majlis Agama Islam Negeri Di Malaysia, Vol. 51, (2) Addison-Wesley Professional, 2017, pp. 39–54, http://dx.doi.org/10.1017/CBO9781107415324.004.
[423] Roel Wieringa, Neil Maiden, Nancy Mead, Colette Rolland, Requirements engineering paper classification and evaluation criteria: a proposal and a discussion, Requir. Eng. (ISSN: 09473602) 11 (1) (2006) 102–107, http://dx.doi.org/10.1007/s00766-005-0021-6.
[426] C Wohlin, P Runeson, M Höst, M C Ohlsson, B Regnell, A Wesslén, Experimentation in Software Engineering, Springer Publishing Company, 2012.
[427] Claes Wohlin, Guidelines for snowballing in systematic literature studies and a replication in software engineering, in: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE '14, (ISSN: 09505849) 2014, pp. 1–10, http://dx.doi.org/10.1145/2601248.2601268.
[428] Claes Wohlin, Rafael Prikladnicki, Systematic literature reviews in software engineering, Inf. Softw. Technol. (ISSN: 09505849) 55 (6) (2013) 919–920, http://dx.doi.org/10.1016/j.infsof.2013.02.002.
[429] Rob Wojcik, Architecting in a complex world: Eliciting and specifying quality attribute requirements, Tech. rep., Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, 2013.
[430] Rob Wojcik, Eliciting and specifying quality attribute requirements, Tech. rep., Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, 2013.
[431] Working together: The team software process and architecture-centric engineering, 2013.
[432] Sezin Gizem Yaman, Tanja Sauvola, Leah Riungu-Kalliosaari, Laura Hokkanen, Pasi Kuvaja, Markku Oivo, Tomi Männistö, Customer involvement in continuous deployment: A systematic literature review, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 9619, Springer, 2016, pp. 249–265, http://dx.doi.org/10.1007/978-3-319-30282-9_18.
[433] Chen Yang, Peng Liang, Paris Avgeriou, A systematic mapping study on the combination of software architecture and agile development, J. Syst. Softw. (ISSN: 01641212) 111 (2016) 157–184, http://dx.doi.org/10.1016/j.jss.2015.09.028.
[434] Robert Yin, Case Study Research: Design and Methods, 3rd Edition (Applied Social Research Methods, Vol. 5), Third ed., Sage Publications, Inc., 2002, http://dx.doi.org/10.1086/421629.
[435] Uwe Zdun, Rafael Capilla, Huy Tran, Olaf Zimmermann, Sustainable architectural design decisions, IEEE Softw. (ISSN: 0740-7459) 30 (6) (2013) 46–53, http://dx.doi.org/10.1109/MS.2013.97.
[436] He Zhang, Muhammad Ali Babar, Paolo Tell, Identifying relevant studies in software engineering, Inf. Softw. Technol. (ISSN: 09505849) 53 (6) (2011) 625–637, http://dx.doi.org/10.1016/j.infsof.2010.12.010.
[437] Junji Zhi, Vahid Garousi-Yusifo lu, Bo Sun, Golara Garousi, Shawn Shahnewaz, Guenther Ruhe, Cost, benefits and quality of software development documentation: A systematic mapping, J. Syst. Softw. (ISSN: 01641212) 99 (2015) 175–198, http://dx.doi.org/10.1016/j.jss.2014.09.042.
[438] Xin Zhou, Yuqin Jin, He Zhang, Shanshan Li, Xin Huang, A map of threats to validity of systematic literature reviews in software engineering, in: Proceedings - Asia-Pacific Software Engineering Conference, APSEC, 2017, pp. 153–160, http://dx.doi.org/10.1109/APSEC.2016.031.
[440] Michał mia, Jacek Bojarski, Wiktor Nowakowski, Tomasz Straszak, Scenario construction tool based on extended UML metamodel, in: Proceedings of the 8th International Conference on Model Driven Engineering Languages and Systems, in: MoDELS'05, Springer-Verlag, 2005, pp. 414–429, http://dx.doi.org/10.1007/11557432_31.
[441] Darja Šmite, Pär J. Ågerfalk, Nils Brede Moe, Agility Across Time and Space: Implementing Agile Methods in Global Software Projects, 2010, pp. 1–341, http://dx.doi.org/10.1007/978-3-642-12442-6.

Further reading

[S8] Ademar Aguiar, Tutorial on agile documentation with wikis, in: Proceedings of the 5th International Symposium on Wikis and Open Collaboration - WikiSym '09, in: WikiSym '09, ACM, New York, NY, USA, 2009, p. 1, http://dx.doi.org/10.1145/1641309.1641365.
[S11] Farzaneh Akbari, Sayed Mehran Sharafi, A review to the usage of concepts of software architecture in agile methods, in: 2012 International Symposium on Instrumentation & Measurement, Sensor Network and Automation (IMSNA), Vol. 2, IEEE, IEEE Instrumentation and Measurement Society, ISBN: 9781467324663, 2012, pp. 389–392, http://dx.doi.org/10.1109/MSNA.2012.6324602.
[S12] Suha Akman, Elif Berru Aksuyek, Onur Kaynak, Alm tool infrastructure with a focus on DevOps culture, in: European Conference on Software Process Improvement, Springer, 2018, pp. 291–303, http://dx.doi.org/10.1007/978-3-319-97925-0_24.

[S13] Jai Vigneshwar Alavandhar, Oksana ÅĚikiforova, Several ideas on integration of SCRUM practices within microsoft solutions framework, Appl. Comput. Syst. 21 (1) (2017) 71–79, http://dx.doi.org/10.1515/acss-2017-0010.

[S14] Anum Ali, Mariam Rehman, Maria Anjum, Framework for applicability of agile scrum methodology: A perspective of software industry, Int. J. Adv. Comput. Sci. Appl. 8 (9) (2017) 225–232, http://dx.doi.org/10.14569/IJACSA.2017.080932.

[S16] Yehia Ibrahim Alzoubi, Asif Qumer Gill, Bruce Moulton, A measurement model to analyze the effect of agile enterprise architecture on geographically distributed agile development, J. Softw. Eng. Res. Dev. 6 (1) (2018) 4, http://dx.doi.org/10.1186/s40411-018-0048-2.

[S17] Hannani Aman, Rosziati Ibrahim, XML-DocTracker: Generating software requirements specification (SRS) from XML schema, in: ICISS 2016 - 2016 International Conference on Information Science and Security, 2017, pp. 1–5, http://dx.doi.org/10.1109/ICISSEC.2016.7885872.

[S23] Ronit Ankori, Automatic requirements elicitation in agile processes, in: Proceedings - IEEE International Conference on Software - Science, Technology and Engineering 2005, SwSTE '05, 2005, pp. 101–109, http://dx.doi.org/10.1109/SWSTE.2005.8.

[S24] Pablo Oliveira Antonino, Thorsten Keuler, Nicolas Germann, Brian Cronauer, A non-invasive approach to trace architecture design, requirements specification and agile artifacts, in: Proceedings, 2014 23rd Australasian Software Engineering Conference : ASWEC 2014 : Sydney, New South Wales, Australia, 7-10 April 2014, in: ASWEC '14, IEEE Computer Society, Institute of Electrical Australian Computer Society and Electronics Engineers, Washington, DC, USA, 2014, pp. 220–229, http://dx.doi.org/10.1109/ASWEC.2014.30.

[S25] Alireza Anvari, Norzima Zulkifli, Omid Arghish, Application of a modified VIKOR method for decision-making problems in lean tool selection, Int. J. Adv. Manuf. Technol. 71 (5–8) (2014) 829–841, http://dx.doi.org/10.1007/s00170-013-5520-x.

[S28] Waqar Aslam, Farah Ijaz, A quantitative framework for task allocation in distributed agile software development, IEEE Access 6 (2018) 15380–15390, http://dx.doi.org/10.1109/ACCESS.2018.2803685.

[S31] Ashish Avasthi, Gaurav Mishra, A new framework for the agile software development method, in: 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), IEEE, IEEE Electron Devices Society; Institute of Electrical RVS Technical Campus and Electronics Engineers, ISBN: 9781538609651, 2018, pp. 436–438, http://dx.doi.org/10.1109/ICECA.2018.8474737.

[S32] Muhammad Ali Babar, Making software architecture and agile approaches work together: Foundations and approaches, in: Muhammad Ali Babar, Alan W. Brown, Ivan Mistrik (Eds.), Agile Software Architecture. Aligning Agile Processes and Software Architectures, Elsevier, 2014, pp. 1–22, http://dx.doi.org/10.1016/B978-0-12-407772-0.00001-0.

[S35] C. Balan, S. Dija, Divya S. Vidyadharan, The need to adopt agile methodology in the development of cyber forensics tools, in: 2010 IEEE International Conference on Computational Intelligence and Computing Research, IEEE, 2010, pp. 1–4, http://dx.doi.org/10.1109/ICCIC.2010.5705815.

[S38] Joaquim Baptista, Agile documentation with uscrum, in: Proceedings of the 26th Annual ACM International Conference on Design of Communication, ACM, 2008, pp. 275–276, http://dx.doi.org/10.1145/1456536.1456596.

[S46] Kent Beck, Test-Driven Development: By Example, Addison-Wesley Professional, 2003.

[S49] Woubshet Behutiye, Pertti Karhapää, Dolors Costal, Markku Oivo, Xavier Franch, Non-functional requirements documentation in agile software development: challenges and solution proposal, in: International Conference on Product-Focused Software Process Improvement, Springer, 2017, pp. 515–522, http://dx.doi.org/10.1007/978-3-319-69926-4_41.

[S55] Andreas Bollin, Dominik Rauner-Reithmayer, Formal specification comprehension: the art of reading and writing z, in: Proceedings of the 2nd FME Workshop on Formal Methods in Software Engineering - FormaliSE 2014, in: FormaliSE 2014, ACM Press, 2014, pp. 3–9, http://dx.doi.org/10.1145/2593489.2593491.

[S56] Gilberto Borrego, Condensing architectural knowledge from unstructured textual media in agile GSD teams, in: Proceedings - 11th IEEE International Conference on Global Software Engineering Companion Proceedings, ICGSEW 2016, 2016, pp. 69–72, http://dx.doi.org/10.1109/ICGSEW.2016.16.

[S57] Gilberto Borrego, Alberto L. Moran, Ramon Palacio, Preliminary evaluation of a tag-based knowledge condensation tool in agile and distributed teams, in: Proceedings - 2017 IEEE 12th International Conference on Global Software Engineering, ICGSE 2017, 2017, pp. 51–55, http://dx.doi.org/10.1109/ICGSE.2017.14.

[S58] Gilberto Borrego, Alberto L. Morán, Ramón René Palacio Cinco, Oscar Mario Rodríguez-Elias, Eloísa García-Canseco, Review of approaches to manage architectural knowledge in agile global software development, IET Softw. (ISSN: 1751-8806) 11 (3) (2017) 77–88, http://dx.doi.org/10.1049/iet-sen.2016.0197.

[S68] L.C. Briand, Software documentation: How much is enough? in: Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR, 2003, pp. 13–15, http://dx.doi.org/10.1109/CSMR.2003.1192406.

[S73] Thomas Buchmann, Towards tool support for agile modeling, in: Proceedings of the 2012 Extreme Modeling Workshop on - XM '12, ACM, 2012, pp. 9–14, http://dx.doi.org/10.1145/2467307.2467310.

[S74] Sabine Buckl, Florian Matthes, Ivan Monahov, Sascha Roth, Christopher Schulz, Christian M. Schweda, Towards an agile design of the enterprise architecture management function, in: Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOC, IEEE, 2011, pp. 322–329, http://dx.doi.org/10.1109/EDOCW.2011.33.

[S75] Matt Callanan, Alexandra Spillane, Devops: Making it easy to do the right thing, IEEE Softw. 33 (3) (2016) 53–59, http://dx.doi.org/10.1109/MS.2016.66.

[S77] Mert Canat, Núria Pol Català, Alexander Jourkovski, Svetlomir Petrov, Martin Wellme, Robert Lagerström, Enterprise architecture and agile development: Friends or foes? in: 2018 IEEE 22nd International Enterprise Distributed Object Computing Workshop (EDOCW), IEEE, 2018, pp. 176–183, http://dx.doi.org/10.1109/EDOCW.2018.00033.

[S78] Fabrizio Cannizzo, Gabriela Marcionetti, Paul Moser, Evolution of the tools and practices of a large distributed agile team, in: Proceedings of the Agile 2008, IEEE Computer Society, 2008, pp. 513–518, http://dx.doi.org/10.1109/Agile.2008.32.

[S79] Giovanni Cantone, Michele Marchesi, Agile Processes in Software Engineering and Extreme Programming: 15th International Conference, XP 2014, Rome, Italy, May 26-30, 2014, Proceedings, Vol. 179, Springer, 2014, http://dx.doi.org/10.1007/978-3-319-06862-6_5.

[S82] Oisın Cawley, Ita Richardson, Xiaofeng Wang, Marco Kuhrmann, A conceptual framework for lean regulated software development, in: Dietmar Pfahl (Ed.), 2015 International Conference on Software and Systems Process (ICSSP) : Proceedings : August 24-26, 2015, Tallinn, Estonia, in: ICSSP 2015, ACM, 2015, pp. 167–168, http://dx.doi.org/10.1145/2785592.2794401.

[S83] Frank K.Y. Chan, James Y.L. Thong, Acceptance of agile methodologies: A critical review and conceptual framework, Decis. Support Syst. (ISSN: 0167-9236) 46 (4) (2009) 803–814, http://dx.doi.org/10.1016/j.dss.2008.11.009.

[S84] Ahuja Chandni, Kaur Parminder, Singh Hardeep, Software architecture evaluation in agile environment, in: M. Hoda, N. Chauhan, S. Quadri, P. Srivastava (Eds.), Advances in Intelligent Systems and Computing, Springer Singapore, 2018, pp. 335–356, http://dx.doi.org/10.1007/978-981-10-8848-3_32.

[S85] Hong-Mei Chen, Rick Kazman, Serge Haziyev, Agile big data analytics development: An architecture-centric approach, in: Tung Bui (Ed.), Proceedings of the 49th Annual Hawaii International Conference on System Sciences (HICSS): 5-8 January 2016, Kauai, Hawaii, in: HICSS '16, IEEE Computer Society, Washington, DC, USA, 2016, pp. 5378–5387, http://dx.doi.org/10.1109/HICSS.2016.665.

[S87] Lianping Chen, Muhammad Ali Babar, Towards an evidence-based understanding of emergence of architecture through continuous refactoring in agile software development, in: Proceedings - Working IEEE/IFIP Conference on Software Architecture 2014, WICSA 2014, 2014, pp. 195–204, http://dx.doi.org/10.1109/WICSA.2014.45.

[S88] Howard Chivers, Richard F. Paige, Xiaocheng Ge, Agile security using an incremental security architecture, in: Proceedings of the 6th International Conference on Extreme Programming and Agile Processes in Software Engineering, Springer-Verlag, 2005, pp. 57–65, http://dx.doi.org/10.1007/11499053_7.

[S89] Jaya Choudhury, B. Thushara, Software documentation in a globally distributed environment, in: 2014 IEEE 9th International Conference on Global Software Engineering, 2014, pp. 90–94, http://dx.doi.org/10.1109/ICGSE.2014.23.

[S90] Henrik BÃęrbak Christensen, Klaus Marius Hansen, Towards architectural information in implementation, in: Proceeding of the 33rd International Conference on Software Engineering - ICSE '11, 2011, p. 928, http://dx.doi.org/10.1145/1985793.1985948.

[S95] Alistair Cockburn, Crystal Clear [Electronic Resource] : A Human-Powered Methodology for Small Teams, Pearson Education, 2005.

[S99] Irina Diana Coman, Giancarlo Succi, An exploratory study of developers' toolbox in an agile team, in: International Conference on Agile Processes and Extreme Programming in Software Engineering, Springer, 2009, pp. 43–52, http://dx.doi.org/10.1007/978-3-642-01853-4_7.

[S103] Thais Cristina Sampaio Machado, Pl Pinheiro, Marcony Leal de Lima Marcelo, Henrique Farias Landim, et al., Towards a verbal decision analysis on the selecting practices of framework SCRUM, in: Proceedings of the Second International Conference on Information Computing and Applications, Springer-Verlag, 2011, pp. 585–594, http://dx.doi.org/10.1007/978-3-642-25255-6_74.

[S109] Claudia O. De Melo, Daniela S. Cruzes, Fabio Kon, Reidar Conradi, Interpretative case studies on agile team productivity and management, Inf. Softw. Technol. 55 (2) (2013) 412–427, http://dx.doi.org/10.1016/j.infsof.2012.09.004.

[S110] Claudio De Meo, Nicola Siena, Luca Riccardi, Francesco Nocera, Angelo Parchitelli, Marina Mongiello, Eugenio Di Sciascio, Niko Mäkitalo, Liquidade: a liquid-based distributed agile and adaptive development environment (DADE) multi-device tool, in: Proceedings of the 1st ACM SIGSOFT International Workshop on Ensemble-Based Software Engineering, ACM, 2018, pp. 9–12, http://dx.doi.org/10.1145/3281022.3281024.

[S111] Adler Diniz De Souza, Rodrigo Duarte Seabra, Juliano Marinho Ribeiro, Lucas E da S Rodrigues, SCRUMI: a board serious virtual game for teaching the SCRUM framework, in: Proceedings of the 39th International Conference on Software Engineering Companion, IEEE Press, 2017, pp. 319–321, http://dx.doi.org/10.1109/ICSE-C.2017.124.

[S112] Lorena Delgadillo, Orlena Gotel, Story-wall: A concept for lightweight requirements management, in: Proceedings - 15th IEEE International Requirements Engineering Conference, RE 2007, 2007, pp. 377–378, http://dx.doi.org/10.1109/RE.2007.41.

[S113] Jean-Marc Desharnais, Bugra Kocaturk, Alain Abran, Using the COSMIC method to evaluate the quality of the documentation of agile user stories, in: Proceedings of the 2011 Joint Conference of the 21st International Workshop on Software Measurement and the 6th International Conference on Software Process and Product Measurement, IEEE Computer Society, 2011, pp. 269–272, http://dx.doi.org/10.1109/IWSM-MENSURA.2011.45.

[S114] Elisabetta Di Nitto, Pooyan Jamshidi, Michele Guerriero, Ilias Spais, Damian A Tamburri, A software architecture framework for quality-aware DevOps, in: Proceedings of the 2nd International Workshop on Quality-Aware DevOps - QUDOS 2016, in: QUDOS 2016, ACM, 2016, pp. 12–17, http://dx.doi.org/10.1145/2945408.2945411.

[S115] J. Andres Diaz-Pace, Matias Nicoletti, Silvia Schiaffino, Santiago Vidal, Producing just enough documentation: The next SAD version problem, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 8636 LNCS, Springer, 2014, pp. 46–60, http://dx.doi.org/10.1007/978-3-319-09940-8_4.

[S118] Uwe Dombrowski, Thomas Richter, The lean production system 4.0 framework–enhancing lean methods by industrie 4.0, in: IFIP International Conference on Advances in Production Management Systems, Springer, 2018, pp. 410–416, http://dx.doi.org/10.1007/978-3-319-99707-0_51.

[S120] Paul Drews, Ingrid Schirmer, Bettina Horlach, Carsten Tekaat, Bimodal enterprise architecture management: The emergence of a new eam function for a BizDevOps-based fast IT, in: 2017 IEEE 21st International Enterprise Distributed Object Computing Workshop (EDOCW), IEEE, 2017, pp. 57–64, http://dx.doi.org/10.1109/EDOCW.2017.18.

[S121] Zoya Durdik, Towards a process for architectural modelling in agile software development, in: Proceedings of the Joint ACM SIGSOFT Conference – QoSA and ACM SIGSOFT Symposium – ISARCS on Quality of Software Architectures – QoSA and Architecting Critical Systems – ISARCS - QoSA-ISARCS '11, ACM, 2011, p. 183, http://dx.doi.org/10.1145/2000259.2000291.

[S126] Matthias Eckhart, Johannes Feiner, How scrum tools may change your agile software development approach, in: International Conference on Software Quality, Springer, 2016, pp. 17–36, http://dx.doi.org/10.1007/978-3-319-27033-3_2.

[S127] Jutta Eckstein, Architecture in large scale agile development, in: International Conference on Agile Software Development, Springer, 2014, pp. 21–29, http://dx.doi.org/10.1007/978-3-319-14358-3_3.

[S128] Peter Eeles, Building a platform for innovation: architecture and agile as key enablers, in: Muhammad Ali Babar, Alan W. Brown, Ivan Mistrik (Eds.), Agile Software Architecture, Elsevier, 2014, pp. 315–333, http://dx.doi.org/10.1016/B978-0-12-407772-0.00012-5.

[S132] Veli-Pekka Eloranta, Kai Koskimies, Lightweight architecture knowledge management for agile software development, in: Muhammad Ali Babar, Alan W. Brown, Ivan Mistrik (Eds.), Agile Software Architecture. Aligning Agile Processes and Software Architectures, Elsevier, 2014, pp. 189–213, http://dx.doi.org/10.1016/B978-0-12-407772-0.00007-1.

[S137] Imane Essebaa, Salima Chantit, Scrum and v lifecycle combined with model-based testing and model driven architecture to deal with evolutionary system issues, in: El Hassan Abdelwahed, Ladjel Bellatreche, Mattéo Golfarelli, Dominique Méry, Carlos Ordonez (Eds.), International Conference on Model and Data Engineering, Springer, 2018, pp. 77–91, http://dx.doi.org/10.1007/978-3-030-00856-7_5.

[S138] M. Eswaramoorthi, G.R. Kathiresan, P.S.S. Prasad, P.V. Mohanram, A survey on lean practices in Indian machine tool industries, Int. J. Adv. Manuf. Technol. 52 (9–12) (2011) 1091–1101, http://dx.doi.org/10.1007/s00170-010-2788-y.

[S139] Extreme programming: A gentle introduction, 2019, http://www.extremeprogramming.org/ (Accessed on 08/02/2019).

[S145] Luana Marques Souza Farias, Luciano Costa Santos, Cláudia Fabiana Gohr, Lucas Carvalho de Oliveira, Matheus Henrique da Silva Amorim, Criteria and practices for lean and green performance assessment: Systematic review and conceptual framework, J. Clean. Prod. (2019) http://dx.doi.org/10.1016/j.jclepro.2019.02.042.

[S149] Brian Fitzgerald, Klaas Jan Stol, Continuous software engineering: A roadmap and agenda, J. Syst. Softw. (ISSN: 01641212) 123 (2017) 176–189, http://dx.doi.org/10.1016/j.jss.2015.06.063.

[S153] Ilenia Fronza, Nabil El Ioini, Luis Corral, Teaching computational thinking using agile software engineering methods: A framework for middle schools, ACM Trans. Comput. Educ. (ISSN: 1946-6226) 17 (4) (2017) 19:1–19:28, http://dx.doi.org/10.1145/3055258.

[S155] Matthias Galster, Samuil Angelov, Understanding the use of reference architectures in agile software development projects, in: European Conference on Software Architecture, Springer, 2015, pp. 268–276, http://dx.doi.org/10.1007/978-3-319-23727-5_22.

[S156] Prashant Gandhi, Nils C. Haugen, Mike Hill, Richard Watt, Creating a living specification using FIT documents, in: Proceedings - AGILE Confernce 2005, 2005, pp. 253–258, http://dx.doi.org/10.1109/ADC.2005.19.

[S159] Sebastian Gerdes, Stefanie Jasser, Matthias Riebisch, Sandra Schröder, Mohamed Soliman, Tilmann Stehle, Towards the essentials of architecture documentation for avoiding architecture erosion, in: Proccedings of the 10th European Conference on Software Architecture Workshops - ECSAW '16, in: ECSAW '16, ACM, 2016, pp. 1–4, http://dx.doi.org/10.1145/2993412.3004844.

[S160] Georges Bou Ghantous, Asif Qumer Gill, Devops reference architecture for multi-cloud IOT applications, in: 2018 IEEE 20th Conference on Business Informatics (CBI), Vol. 1, IEEE, 2018, pp. 158–167, http://dx.doi.org/10.1109/CBI.2018.00026.

[S161] Asif Qumer Gill, Agile enterprise architecture modelling, Inf. Softw. Technol. (ISSN: 0950-5849) 67 (C) (2015) 196–206, http://dx.doi.org/10.1016/j.infsof.2015.07.002.

[S164] Greg Goth, Agile tool market growing with the philosophy, IEEE Softw. 26 (2) (2009) 88–91, http://dx.doi.org/10.1109/MS.2009.30.

[S165] Anil Gupta, T.K. Kundra, A review of designing machine tool for leanness, Sadhana 37 (2) (2012) 241–259, http://dx.doi.org/10.1007/s12046-012-0062-8.

[S167] Irit Hadar, Sofia Sherman, Ethan Hadar, John J. Harrison, Less is more: Architecture documentation for agile development, in: 2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE 2013 - Proceedings, 2013, pp. 121–124, http://dx.doi.org/10.1109/CHASE.2013.6614746.

[S172] Muhammad Hammad, Irum Inayat, Integrating risk management in scrum framework, in: 2018 International Conference on Frontiers of Information Technology (FIT), IEEE, 2018, pp. 158–163, http://dx.doi.org/10.1109/FIT.2018.00035.

[S173] Sebastian Hanschke, Jan Ernsting, Herbert Kuchen, Integrating agile software development and enterprise architecture management, in: 48th Hawaii International Conference on System Sciences (HICSS), 2015 5 - 8 Jan. 2015, Kauai, Hawaii, in: HICSS '15, IEEE Computer Society, Washington, DC, USA, 2015, pp. 4099–4108, http://dx.doi.org/10.1109/HICSS.2015.492.

[S175] Lise Tordrup Heeager, How can agile and documentation-driven methods be meshed in practice? in: Agile Processes in Software Engineering and Extreme Programming, 179 LNBIP, Springer, 2014, pp. 62–77, http://dx.doi.org/10.1007/978-3-319-06862-6.

[S177] Uwe van Heesch, Paris Avgeriou, A pattern driven approach against architectural knowledge vaporization, in: Allan Kelly, Michael Weiss (Eds.), Proceedings of the 14th European Conference on Pattern Languages of Programs (EuroPLoP), Irsee, Conference Proceedings, 2009, pp. 1–12.

[S179] José Hernández-Reveles, Gabriela Sobrevilla-Dominguez, Perla Velasco-Elizondo, Silvia Soriano-Grande, Adding agile architecture practices to a cyber-physial system development, in: 2016 International Conference on Software Process Improvement (CIMPS), IEEE, 2016, pp. 1–6, http://dx.doi.org/10.1109/CIMPS.2016.7802800.

[S181] Anne Hess, Philipp Diebold, Norbert Seyff, Towards requirements communication and documentation guidelines for agile teams, in: Proceedings - 2017 IEEE 25th International Requirements Engineering Conference Workshops, REW 2017, 2017, pp. 415–418, http://dx.doi.org/10.1109/REW.2017.64.

[S182] James A. Highsmith, Adaptive Software Development: A Collaborative Approach To Managing Complex Systems, Vol. 12, Addison-Wesley, 2000, p. 392.

[S184] Rashina Hoda, James Noble, Stuart Marshall, How much is just enough?: Some documentation patterns on agile projects, in: Proceedings of the 15th European Conference on Pattern Languages of Programs, ACM, 2010, pp. 13:1–13:13, http://dx.doi.org/10.1145/2328909.2328926.

[S188] Emam Hossain, Paul L. Bannerman, D. Ross Jeffery, Scrum practices in global software development: a research framework, in: Proceedings of the 12th International Conference on Product-Focused Software Process Improvement, Springer-Verlag, 2011, pp. 88–102, http://dx.doi.org/10.1007/978-3-642-21843-9_9.

[S192] David Hutchison, John C. Mitchell, Unconventional Programming Paradigms, Vol. 3566, 2005, http://dx.doi.org/10.1007/11527800.

[S195] José Ignacio Fernández-Villamor, Laura Díaz-Casillas, Carlos Á. Iglesias, A comparison model for agile web frameworks, in: Proceedings of the 2008 Euro American Conference on Telematics and Information Systems - EATIS '08, ACM, 2008, pp. 1–8, http://dx.doi.org/10.1145/1621087.1621101.

[S196] Roumiana Ilieva, Kiril Anguelov, Mario Nikolov, A cynefin framework for agile decision making of AI BOTS, in: 2018 International Conference on High Technology for Sustainable Development (HiTech), IEEE, Tekhnicheski universitet–Sofiya; Institute of Electrical and Electronics Engineers. Bulgaria Section; Institute of Electrical and Engineers, Electronics, 2018, pp. 1–4, http://dx.doi.org/10.1109/HiTech.2018.8566411, A Cynefin Framework for Agile Decision Making of AI BOTS.

[S197] Mark Isham, Agile architecture IS possible–you first have to believe!, in: Agile 2008 Conference, IEEE, 2008, pp. 484–489, http://dx.doi.org/10.1109/Agile.2008.16.

[S199] IEC, IEEE ISO, Systems and software engineering – developing user documentation in an agile environment, ISO/IEC/IEEE, 2012, pp. 1–36, http://dx.doi.org/10.1109/IEEESTD.2012.6170923, tech report.

[S206] Aleksander Jarzębowicz, Katarzyna Połocka, Selecting requirements documentation techniques for software projects: a survey study, in: 2017 Federated Conference on Computer Science and Information Systems (FedCSIS), 2017, pp. 1189–1198, http://dx.doi.org/10.15439/2017F387.

[S207] Taghi Javdani Gandomani, Mina Ziaei Nafchi, An empirically-developed framework for agile transition and adoption, J. Syst. Softw. (ISSN: 0164-1212) 107 (C) (2015) 204–219, http://dx.doi.org/10.1016/j.jss.2015.06.006.

[S212] Li Jiang, Armin Eberlein, Towards a framework for understanding the relationships between classical software engineering and agile methodologies, in: Proceedings of the 2008 International Workshop on Scrutinizing Agile Practices Or Shoot-Out At the Agile Corral - APOS '08, ACM, 2008, pp. 9–14, http://dx.doi.org/10.1145/1370143.1370146.

[S213] Shuwei Jing, Zhanwen Niu, Pei-Chann Chang, The application of VIKOR for the tool selection in lean management, J. Intell. Manuf. (2019) 1–12, http://dx.doi.org/10.1007/s10845-015-1152-3.

[S214] Stephen Jones, Joost Noppen, Fiona Lettice, Management challenges for DevOps adoption within UK SMEs, in: Proceedings of the 2nd International Workshop on Quality-Aware DevOps, ACM, Association for Computing Machinery Special Interest Group on Software Engineering, 2016, pp. 7–11, http://dx.doi.org/10.1145/2945408.2945410.

[S217] J. Kaariainen, J. Koskela, P. Abrahamsson, J. Takalo, Improving requirements management in extreme programming with tool support - an improvement attempt that failed, in: Proceedings. 30th Euromicro Conference, 2004., in: EUROMICRO '04, IEEE Computer Society, Washington, DC, USA, 2004, pp. 342–351, http://dx.doi.org/10.1109/EURMIC.2004.1333389.

[S220] Teemu Karvonen, Tanja Suomalainen, Marko Juntunen, Tanja Sauvola, Pasi Kuvaja, Markku Oivo, The CRUSOE framework: A holistic approach to analysing prerequisites for continuous software engineering, in: International Conference on Product-Focused Software Process Improvement, Springer, 2016, pp. 643–661, http://dx.doi.org/10.1007/978-3-319-49094-6_52.

[S221] R.K. Kavitha, M.S. Irfan Ahmed, A knowledge management framework for agile software development teams, in: 2011 International Conference on Process Automation, Control and Computing, 2011, pp. 1–5, http://dx.doi.org/10.1109/PACC.2011.5978877.

[S224] Mik Kersten, A cambrian explosion of DevOps tools, IEEE Softw. (ISSN: 07407459) 35 (2) (2018) 14–17, http://dx.doi.org/10.1109/MS.2018.1661330.

[S225] Noureddine Kerzazi, Bram Adams, Who needs release and devops engineers, and why? in: Proceedings of the International Workshop on Continuous Software Evolution and Delivery, in: CSED '16, ACM, Austin, Texas, 2016, pp. 77–83, http://dx.doi.org/10.1145/2896941.2896957.

[S227] Md Asif Khalil, Bonthu Kotaiah, Implementation of agile methodology based on SCRUM tool, in: 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), IEEE, 2017, pp. 2351–2357, http://dx.doi.org/10.1109/ICECDS.2017.8389872.

[S237] Eric Knauss, Grischa Liebel, Jennifer Horkoff, Rebekka Wohlrab, Rashidah Kasauli, Filip Lange, Pierre Gildert, T-reqs: Tool support for managing requirements in large-scale agile system development, in: 2018 IEEE 26th International Requirements Engineering Conference (RE), IEEE, 2018, pp. 502–503, http://dx.doi.org/10.1109/RE.2018.00073.

[S240] Andreas Kornstädt, Joachim Sauer, Mastering dual-shore development: the tools and materials approach adapted to agile offshoring, in: Proceedings of the 1st International Conference on Software Engineering Approaches for Offshore and Outsourced Development, Springer-Verlag, 2007, pp. 83–95, http://dx.doi.org/10.1007/978-3-540-75542-5_7.

[S242] Steinar Kristoffersen, The-Hien Dang-Ha, Thien-Phuc Nguyen, Paperstack-a novel lean-interactive system for documentation sharing in maritime industries, in: 2013 International Conference on Collaboration Technologies and Systems (CTS), IEEE, 2013, pp. 546–552, http://dx.doi.org/10.1109/CTS.2013.6567285.

[S243] P. Kruchten, Software architecture and agile software development: a clash of two cultures? in: 2010 ACM/IEEE 32nd International Conference on Software Engineering, Vol. 2, 2010, pp. 497–498, http://dx.doi.org/10.1145/1810295.1810448.

[S246] Philippe B. Kruchten, The 4+1 view model of architecture, IEEE Softw. (ISSN: 07407459) 12 (6) (1995) 42–50, http://dx.doi.org/10.1109/52.469759.

[S248] M Upendra Kumar, D Sravan Kumar, B Padmaja Rani, K Venkateswar Rao, AV Krishna Prasad, D Shravani, Dependable solutions design by agile modeled layered security architectures, in: International Conference on Computer Science and Information Technology, Springer, 2012, pp. 510–519, http://dx.doi.org/10.1007/978-3-642-27299-8_53.

[S250] Kaisa Könnölä, Samuli Suomi, Tuomas Mäkilä, Tero Jokela, Ville Rantala, Teijo Lehtonen, Agile methods in embedded system development: Multiple-case study of three industrial cases, J. Syst. Softw. (ISSN: 01641212) 118 (2016) 134–150, http://dx.doi.org/10.1016/j.jss.2016.05.001.

[S255] Ana Isabella Muniz Leite, An approach to support the specification of agile artifacts in the development of safety-critical systems, in: Proceedings - 2017 IEEE 25th International Requirements Engineering Conference, RE 2017, 2017, pp. 526–531, http://dx.doi.org/10.1109/RE.2017.43.

[S258] Jun Lin, Han Yu, Zhiqi Shen, Chunyan Miao, Studying task allocation decisions of novice agile teams with data from agile project management tools, in: Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering, ACM, 2014, pp. 689–694, http://dx.doi.org/10.1145/2642937.2642959.

[S263] Ralf Lämmel, Simon Peyton Jones, Scrap your boilerplate, ACM SIGPLAN Not. (ISSN: 03621340) 38 (3) (2005) 26, http://dx.doi.org/10.1145/640136.604179.

[S265] Mark Mahoney, Telling stories about software evolution, in: 2011 AGILE Conference, 2011, pp. 127–130, http://dx.doi.org/10.1109/AGILE.2011.22.

[S267] Dennis Mancl, Steven Fraser, Bill Opdyke, E. Hadar, Architecture in an agile world, in: Proceedings of the 24th, ACM, 2009, pp. 841–843, http://dx.doi.org/10.1145/1639950.1640041.

[S268] Christian Manteuffel, Dan Tofan, Heiko Koziolek, Thomas Goldschmidt, Paris Avgeriou, Industrial implementation of a documentation framework for architectural decisions, in: Proceedings - Working IEEE/IFIP Conference on Software Architecture 2014, WICSA 2014, IEEE, 2014, pp. 225–234, http://dx.doi.org/10.1109/WICSA.2014.32.

[S269] Rafael P. Maranzato, Marden Neubert, Paula Herculano, Scaling scrum step by step: "the mega framework", in: Agile Alliance (Ed.), Proceedings of the 2012 Agile Conference, in: AGILE '12, IEEE Computer Society, Washington, DC, USA, 2012, pp. 79–85, http://dx.doi.org/10.1109/Agile.2012.22.

[S271] Antonio Martini, Jan Bosch, A multiple case study of continuous architecting in large agile companies: current gaps and the CAFFEA framework, in: 2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA), IEEE Computer Society, IEEE, Los Alamitos, CA, USA, 2016, pp. 1–10, http://dx.doi.org/10.1109/WICSA.2016.31.

[S272] Antonio Martini, Lars Pareto, Jan Bosch, Towards introducing agile architecting in large companies: the caffea framework, in: International Conference on Agile Software Development, Springer, 2015, pp. 218–223, http://dx.doi.org/10.1007/978-3-319-18612-2_20.

[S273] Ciro Fernandes Matrigrani, Talita Santos, Lineu Fernando Stege Mialaret, Adilson Marques da Cunha, Luiz Alberto Vieira Dias, Applying collective intelligence in the evolution of a project architecture using agile methods, in: Information Technology-New Generations, Springer, 2018, pp. 939–942, http://dx.doi.org/10.1007/978-3-319-54978-1_121.

[S274] Juliana Medeiros, Alexandre Vasconcelos, Miguel Goulão, Carla Silva, João Araújo, An approach based on design practices to specify requirements in agile projects, in: Proceedings of the Symposium on Applied Computing - SAC '17, in: SAC '17, ACM, New York, NY, USA, 2017, pp. 1114–1121, http://dx.doi.org/10.1145/3019612.3019753.

[S275] Thiago Souto Mendes, et al., Impacts of agile requirements documentation debt on software projects: a retrospective study, in: Proceedings of the 31st Annual ACM Symposium on Applied Computing, ACM, 2016, pp. 1290–1295, http://dx.doi.org/10.1145/2851613.2851761.

[S276] Gytenis Mikulenas, Rimantas Butleris, Lina Nemuraite, An approach for the metamodel of the framework for a partial agile method adaptation, Inf. Technol. Control (ISSN: 1392124X) 40 (1) (2011) 71–82, http://dx.doi.org/10.5755/j01.itc.40.1.194.

[S279] Christine Miyachi, Agile software architecture, ACM SIGSOFT Softw. Eng. Notes (ISSN: 01635948) 36 (2) (2011) 1, http://dx.doi.org/10.1145/1943371.1943388.

[S282] Mirna Muñoz, Jezreel Mejia, Brisia Corona, Jose A Calvo-Manzano, Tomas San Feliu, Juan Miramontes, Analysis of tools for assessing the implementation and use of agile methodologies in SMEs, in: International Conference on Software Process Improvement and Capability Determination, Springer, 2016, pp. 123–134, http://dx.doi.org/10.1007/978-3-319-38980-6_10.

[S284] Jerzy Nawrocki, Michał Jasiński, Bartosz Walter, Adam Wojciechowski, Extreme programming modified: Embrace requirements engineering practices, in: Proceedings of the IEEE International Conference on Requirements Engineering, 2002-Janua, 2002, pp. 303–310, http://dx.doi.org/10.1109/ICRE.2002.1048543.

[S290] Robert L. Nord, Ipek Ozkaya, Philippe Kruchten, Agile in distress: Architecture to the rescue, in: Lecture Notes in Business Information Processing, 2014, pp. 43–57, http://dx.doi.org/10.1007/978-3-319-14358-3_5.

[S291] Robert L. Nord, Ipek Ozkaya, Raghvinder S. Sangwan, Making architecture visible to improve flow management in lean software development, IEEE Softw. (ISSN: 07407459) 29 (5) (2012) 33–39, http://dx.doi.org/10.1109/MS.2012.109.

[S292] Robert L. Nord, James E. Tomayko, Software architecture-centric methods and agile development, IEEE Softw. (ISSN: 07407459) 23 (2) (2006) 47–53, http://dx.doi.org/10.1109/MS.2006.54.

[S296] Mika Ohtsuki, Kazuki Ohta, Tetsuro Kakeshita, Software engineer education support system ALECSS utilizing DevOps tools, in: Proceedings of the 18th International Conference on Information Integration and Web-Based Applications and Services, ACM, 2016, pp. 209–213, http://dx.doi.org/10.1145/3011141.3011200.

[S304] F. Paetsch, A. Eberlein, F. Maurer, Requirements engineering and agile software development, in: WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003., in: WETICE '03, IEEE Computer Society, Washington, DC, USA, 2003, pp. 308–313, http://dx.doi.org/10.1109/ENABL.2003.1231428.

[S305] Steve R. Palmer, Mac Felsing, A Practical Guide To Feature-Driven Development, Pearson Education, 2001.

[S309] S.S. Samarawickrama; I. Perera, Continuous scrum: A framework to enhance scrum with DevOps, in: 2017 Seventeenth International Conference on Advances in ICT for Emerging Regions (ICTer), IEEE Sri Lanka Section University of Colombo School of Computing, IEEE, Colombo, 2017, http://dx.doi.org/10.1109/ICTER.2017.8257808.

[S316] S. Pinna, P. Lorrai, M. Marchesi, N. Serra, Developing a tool supporting XP process, in: Extreme Programming and Agile Methods - XP/Agile Universe 2003, in: Lecture Notes in Computer Science, Vol. 2753, Springer-Verlag Berlin, 2003, pp. 151–160, http://dx.doi.org/10.1007/978-3-540-45122-8_17.

[S318] Jalal Joseph Possik, Aicha Anne Amrani, Gregory Zacharewicz, Development of a co-simulation system as a decision-aid in lean tools implementation, in: Proceedings of the 50th Computer Simulation Conference, in: SummerSim '18, Society for Computer Simulation International, Bordeaux, France, 2018, pp. 21:1–21:12.

[S320] AV Krishna Prasad, S Ramakrishna, B Padmaja Rani, M Upendra Kumar, D Shravani, Designing dependable business intelligence solutions using agile web services mining architectures, in: Information Technology and Mobile Communication, Springer, 2011, pp. 301–304, http://dx.doi.org/10.1007/978-3-642-20573-6_51.

[S322] Christian R. Prause, Zoya Durdik, Architectural design and documentation: Waste in agile development? in: 2012 International Conference on Software and System Process, ICSSP 2012 - Proceedings, in: ICSSP '12, IEEE Press, Piscataway, NJ, USA, 2012, pp. 130–134, http://dx.doi.org/10.1109/ICSSP.2012.6225956.

[S323] Jennifer Pérez, et al., Bridging user stories and software architecture: A tailored scrum for agile architecting, in: Muhammad Babar (Ed.), Agile Software Architecture : Aligning Agile Processes and Software Architectures, Elsevier/Morgan Kaufmann, 2014, pp. 215–241, http://dx.doi.org/10.1016/B978-0-12-407772-0.00008-3.

[S324] Jennifer Pérez, et al., Flexible working architectures: agile architecting using PPCs, in: Proceedings of the 4th European Conference on Software Architecture, Springer-Verlag, 2010, pp. 102–117, http://dx.doi.org/10.1007/978-3-642-15114-9_10.

[S325] Florian Raith, Ingo Richter, Robert Lindermeier, How project-management-tools are used in agile practice: Benefits, drawbacks and potentials, in: Proceedings of the 21st International Database Engineering & Applications Symposium, ACM, 2017, pp. 30–39, http://dx.doi.org/10.1145/3105831.3105865.

[S330] Sergio Gálvez Rojas, José Manuel Carrasco Mora, Source code documentation simul loco, in: Iberian Conference on Information Systems and Technologies, CISTI, 2012, pp. 669–673.

[S332] Dominik Rost, Balthasar Weitzel, Matthias Naab, Torsten Lenhart, Hartmut Schmitt, Distilling best practices for agile development from architecture methodology, in: European Conference on Software Architecture, Springer, 2015, pp. 259–267, http://dx.doi.org/10.1007/978-3-319-23727-5_21.

[S334] Eran Rubin, Hillel Rubin, Supporting agile software development through active documentation, Requirements Eng. (ISSN: 09473602) 16 (2) (2011) 117–132, http://dx.doi.org/10.1007/s00766-010-0113-9.

[S337] Daniel Russo, Paolo Ciancarini, Tommaso Falasconi, Massimo Tomasi, A meta-model for information systems quality, ACM Trans. Manag. Inf. Syst. (ISSN: 2158656X) 9 (3) (2018) 1–38, http://dx.doi.org/10.1145/3230713.

[S339] Abdelkebir Sahid, Yassine Maleh, Mustapha Belaissaoui, A practical agile framework for IT service and asset management ITSM/ITAM through a case study, J. Cases Inf. Technol. (ISSN: 1548-7717) 20 (4) (2018) 71–92, http://dx.doi.org/10.4018/JCIT.2018100105.

[S340] Shinobu Saito, Yukako Iimura, Aaron K Massey, Annie I. Anton, How much undocumented knowledge is there in agile software development?: Case study on industrial project using issue tracking system and version control system, in: Proceedings - 2017 IEEE 25th International Requirements Engineering Conference, RE 2017, 2017, pp. 194–203, http://dx.doi.org/10.1109/RE.2017.33.

[S341] Dina Salah, A framework for the integration of user centered design and agile software development processes, in: Richard Taylor (Ed.), ICSE'11 : 33rd International Conference on Software Engineering : Proceedings of the Conference, May 21-28, 2011, Waikiki, Honolulu, Hawaii, in: ICSE '11, ACM, Waikiki, Honolulu, HI, USA, 2011, pp. 1132–1133, http://dx.doi.org/10.1145/1985793.1986017.

[S344] Nuno Santos, Jaime Pereira, Francisco Morais, Júlio Barros, Nuno Ferreira, Ricardo J Machado, An agile modeling oriented process for logical architecture design, in: Enterprise, Business-Process and Information Systems Modeling, Springer, 2018, pp. 260–275, http://dx.doi.org/10.1007/978-3-319-91704-7_17.

[S346] T. Sauer, Using design rationales for agile documentation, in: Proceedings of the Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE, 2003-Janua, 2003, pp. 326–331, http://dx.doi.org/10.1109/ENABL.2003.1231431.

[S349] Vitali Schneider, Reinhard German, Integration of test-driven agile simulation approach in service-oriented tool environment, in: Proceedings of the 46th Annual Simulation Symposium, Society for Computer Simulation International, 2013, p. 11.

[S350] Timo Schröders, Virgilio Cruz-Machado, Sustainable lean implementation: An assessment tool, in: Proceedings of the Ninth International Conference on Management Science and Engineering Management, Springer, 2015, pp. 1249–1264, http://dx.doi.org/10.1007/978-3-662-47241-5_105.

[S352] Karl Scotland, Alexandre Boutin, Integrating scrum with the process framework at yahoo! europe, in: Proceedings of the Agile 2008, IEEE Computer Society, 2008, pp. 191–195, http://dx.doi.org/10.1109/Agile.2008.22.

[S356] Mohamed Shalaby, Sherif El-Kassas, Applying scrum framework in the IT service support domain, in: 2011 IEEE Asia-Pacific Services Computing Conference, IEEE, 2011, pp. 9–15, http://dx.doi.org/10.1109/APSCC.2011.76.

[S359] Seung Woo Shin, Haeng Kon Kim, A framework for SOA-based application on agile of small and medium enterprise, in: Computer and Information Science, Springer, 2008, pp. 107–120, http://dx.doi.org/10.1007/978-3-540-79187-4_10.

[S361] Suprika Vasudeva Shrivastava, Urvashi Rathod, A risk management framework for distributed agile projects, Inf. Softw. Technol. 85 (2017) 1–15, http://dx.doi.org/10.1016/j.infsof.2016.12.005.

[S364] Henrique F Soares, Nicolli S.R. Alves, Thiago S Mendes, Manoel Mendonca, Rodrigo O. Spinola, Investigating the link between user stories and documentation debt on software projects, in: Proceedings - 12th International Conference on Information Technology: New Generations, ITNG 2015, 2015, pp. 385–390, http://dx.doi.org/10.1109/ITNG.2015.68.

[S371] Adler Diniz de Souza, Rodrigo Duarte Seabra, Juliano Marinho Ribeiro, Lucas Eduardo da Silva Rodrigues, An experience of using a board serious virtual game for teaching the SCRUM framework, in: S. Latifi (Ed.), Information Technology-New Generations. Advances in Intelligent Systems and Computing, Springer, 2018, pp. 213–218, http://dx.doi.org/10.1007/978-3-319-77028-4_31.

[S372] Sergio Cozzetti B de Souza, Nicolas Anquetil, Káthia M de Oliveira, A study of the documentation essential to software maintenance, in: Proceedings of the 23rd Annual International Conference on Design of Communication Documenting & Designing for Pervasive Information - SIGDOC '05, in: SIGDOC '05, ACM, New York, NY, USA, 2005, p. 68, http://dx.doi.org/10.1145/1085313.1085331.

[S373] J. Staples, Agile Documentation: A Pattern Guide To Producing Lightweight Documents for Software Projects, Vol. 51, (4) John Wiley & Sons, 2004, pp. 560–561.

[S375] Christoph Johann Stettina, Werner Heijstek, Necessary and neglected? An empirical study of internal documentation in agile software development teams, in: Proceedings of the 29th ACM International Conference on Design of Communication - SIGDOC '11, in: SIGDOC '11, ACM, New York, NY, USA, 2011, p. 159, http://dx.doi.org/10.1145/2038476.2038509.

[S376] Christoph Johann Stettina, Werner Heijstek, Tor Erlend FÃegri, Documentation work in agile teams: The role of documentation formalism in achieving a sustainable practice, in: Proceedings - 2012 Agile Conference, Agile 2012, 2012, pp. 31–40, http://dx.doi.org/10.1109/Agile.2012.7.

[S377] Christoph Johann Stettina, Egbert Kroon, Is there an agile handover? An empirical study of documentation and project handover practices across agile software teams, in: 2013 International Conference on Engineering, Technology and Innovation, ICE 2013 and IEEE International Technology Management Conference, ITMC 2013, 2015, pp. 1–12, http://dx.doi.org/10.1109/ITMC.2013.7352703.

[S379] Daniel Ståhl, Jan Bosch, Cinders: The continuous integration and delivery architecture framework, Inf. Softw. Technol. (ISSN: 09505849) 83 (2017) 76–93, http://dx.doi.org/10.1016/j.infsof.2016.11.006.

[S381] Nachiappan Subramanian, Angappa Gunasekaran, Cleaner supply-chain management practices for twenty-first-century organizational competitiveness: Practice-performance framework and research propositions, Int. J. Prod. Econ. 164 (2015) 216–233, http://dx.doi.org/10.1016/j.ijpe.2014.12.002.

[S382] Yu Xia Sun, Huo Yan Chen, T.H. Tse, Lean implementations of software testing tools using xml representations of source codes, in: Proceedings of the 2008 International Conference on Computer Science and Software Engineering-Volume 02, IEEE Computer Society, 2008, pp. 708–711, http://dx.doi.org/10.1109/CSSE.2008.515.

[S385] Binish Tanveer, Liliana Guzmán, Ulf Martin Engel, Effort estimation in agile software development: Case study and improvement framework, J. Softw.: Evol. Process 29 (11) (2017) e1862, http://dx.doi.org/10.1002/smr.1862.

[S389] Theo Theunissen, Uwe Van Heesch, Specification in continuous software development, in: Proceedings of the 22nd European Conference on Pattern Languages of Programs, in: EuroPLoP '17, ACM, New York, NY, USA, 2017, p. 5, http://dx.doi.org/10.1145/3147704.3147709.

[S390] Sirin Thongsukh, Smitti Darakorn Na Ayuthaya, et al., Startup framework based on scrum framework, in: 2017 International Conference on Digital Arts, Media and Technology (ICDAMT), IEEE, Section, Thailand, 2017, pp. 458–463, http://dx.doi.org/10.1109/ICDAMT.2017.7905012.

[S395] Ömer Uludağ, Martin Kleehaus, Xian Xu, Florian Matthes, Investigating the role of architects in scaling agile frameworks, in: 2017 IEEE 21st International Enterprise Distributed Object Computing Conference (EDOC), IEEE, 2017, pp. 123–132, http://dx.doi.org/10.1109/EDOC.2017.25.

[S396] Juan Urrego, et al., Archinotes: a global agile architecture design approach, in: Agile Processes in Software Engineering and Extreme Programming, Springer-Verlag, 2014, pp. 302–311, http://dx.doi.org/10.1007/978-3-319-06862-6_24.

[S398] Edward Uy, Rene Rosendahl, Migrating from SharePoint to a better scrum tool, in: Proceedings of the Agile 2008, IEEE Computer Society, 2008, pp. 506–512, http://dx.doi.org/10.1109/Agile.2008.69.

[S400] U. Van Heesch, P. Avgeriou, R. Hilliard, A documentation framework for architecture decisions, J. Syst. Softw. (ISSN: 01641212) 85 (4) (2012) 795–820, http://dx.doi.org/10.1016/j.jss.2011.10.017.

[S402] Uwe Van Heesch, Theo Theunissen, Olaf Zimmermann, Uwe Zdun, Software specification and documentation in continuous software development: A focus group report, in: Proceedings of the 22nd European Conference on Pattern Languages of Programs, in: EuroPLoP '17, ACM, Hillside Europe (Organization), Association for Computing Machinery, Irsee, Germany, 2017, pp. 35:1–35:13, http://dx.doi.org/10.1145/3147704.3147742.

[S405] Sai Datta Vishnubhotla, Emilia Mendes, Lars Lundberg, Designing a capability-centric web tool to support agile team composition and task allocation : A work in progress, in: Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering, in: CHASE '18, ACM, IEEE, Gothenburg, Sweden, 2018, pp. 41–44, http://dx.doi.org/10.1145/3195836.3195855.

[S406] Stefan Voigt, Detlef Huttemann, Andreas Gohr, Sprintdoc: Concept for an agile documentation tool, in: Iberian Conference on Information Systems and Technologies, CISTI, 2016-July, 2016, pp. 1–6, http://dx.doi.org/10.1109/CISTI.2016.7521550.

[S407] Stefan Voigt, Jorg von Garrel, Julia Muller, Dominic Wirth, A study of documentation in agile software projects, in: Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '16, in: ESEM '16, ACM, New York, NY, USA, 2016, pp. 1–6, http://dx.doi.org/10.1145/2961111.2962616.

[S408] Jarno Vähäniitty, Kristian T. Rautiainen, Towards a conceptual framework and tool support for linking long-term product and business planning with agile software development, in: Proceedings of the 1st International Workshop on Software Development Governance - SDG '08, ACM, 2008, p. 25, http://dx.doi.org/10.1145/1370720.1370730.

[S409] Gerard Wagenaar, Sietse Overbeek, Remko Helms, Describing criteria for selecting a scrum tool using the technology acceptance model, in: Asian Conference on Intelligent Information and Database Systems, Springer, 2017, pp. 811–821, http://dx.doi.org/10.1007/978-3-319-54430-4_77.

[S410] Gerard Wagenaar, Sietse Overbeek, Garm Lucassen, Sjaak Brinkkemper, Kurt Schneider, Working software over comprehensive documentation–rationales of agile teams for artefacts usage, J. Softw. Eng. Res. Dev. 6 (1) (2018) 7, http://dx.doi.org/10.1186/s40411-018-0051-7.

[S411] Leslie J. Waguespack, William T. Schiano, Scrum project architecture and thriving systems theory, in: Proceedings of the 2012 45th Hawaii International Conference on System Sciences, in: HICSS '12, IEEE Computer Society, Washington, DC, USA, 2012, pp. 4943–4951, http://dx.doi.org/10.1109/HICSS.2012.513.

[S413] Todd Waits, Joseph Yankel, Continuous system and user documentation integration, in: IEEE International Professional Communication Conference, 2015-Janua, 2015, pp. 1–5, http://dx.doi.org/10.1109/IPCC.2014.7020385.

[S414] Xiaofeng Wang, Kieran Conboy, Oisin Cawley, "Leagile" software development: An experience report analysis of the application of lean approaches in agile software development, J. Syst. Softw. (ISSN: 01641212) 85 (6) (2012) 1287–1299, http://dx.doi.org/10.1016/j.jss.2012.01.061.

[S415] Yang Wang, Ivan Bogicevic, Stefan Wagner, A study of safety documentation in a scrum development process, in: Proceedings of the XP2017 Scientific Workshops, ACM, 2017, p. 22, http://dx.doi.org/10.1145/3120459.3120482.

[S416] Muhammad Waseem, Peng Liang, Microservices architecture in DevOps, in: 2017 24th Asia-Pacific Software Engineering Conference Workshops (APSECW), IEEE, 2017, pp. 13–14, http://dx.doi.org/10.1109/APSECW.2017.18.

[S417] Michael Waterman, Agility, risk, and uncertainty, part 1: designing an agile architecture, IEEE Softw. 35 (2) (2018) 99–101, http://dx.doi.org/10.1109/MS.2018.1661335.

[S418] Michael Waterman, James Noble, George Allan, The effect of complexity and value on architecture planning in agile software development, in: International Conference on Agile Software Development, Springer, 2013, pp. 238–252, http://dx.doi.org/10.1007/978-3-642-38314-4_17.

[S419] Gero Wedemann, Scrum as a method of teaching software architecture, in: Jürgen Mottok (Ed.), Proceedings of the 3rd ECSEE, European Conference Software Engineering Education : Seeon Monastery, Germany, 15 and 16 June 2018, in: ECSEE'18, ACM, Seeon/ Bavaria, Germany, 2018, pp. 108–112, http://dx.doi.org/10.1145/3209087.3209096.

[S420] Jan Werewka, Krzysztof Jamr??z, Dariusz Pitulej, Developing lean architecture governance at a software developing company applying ArchiMate motivation and business layers, in: Communications in Computer and Information Science, Vol. 424, Springer, 2014, pp. 492–503, http://dx.doi.org/10.1007/978-3-319-06932-6_48.

[S421] Jan Werewka, Anna Spiechowicz, Enterprise architecture approach to SCRUM processes, sprint retrospective example, in: 2017 Federated Conference on Computer Science and Information Systems (FedCSIS), Staff, IEEE, 2017, pp. 1221–1228, http://dx.doi.org/10.15439/2017F96.

[S422] Johannes Wettinger, Uwe Breitenbücher, Oliver Kopp, Frank Leymann, Streamlining DevOps automation for cloud applications using TOSCA as standardized metamodel, Future Gener. Comput. Syst. (ISSN: 0167739X) 56 (2016) 317–332, http://dx.doi.org/10.1016/j.future.2015.07.017.

[S424] Werner Wild, Barbara Weber, Hubert Baumeister, Aosta: Agile open source tools academy, in: International Conference on Agile Processes and Extreme Programming in Software Engineering, Springer, 2008, pp. 248–249, http://dx.doi.org/10.1007/978-3-540-68255-4_42.

[S425] Laurie Williams, Agile software development methodologies and practices, in: Advances in Computers, Vol. 80, Elsevier, 2010, pp. 1–44, http://dx.doi.org/10.1016/S0065-2458(10)80001-4.

[S439] Olaf Zimmermann, Designed and delivered today, eroded tomorrow?: Towards an open and lean architecting framework balancing agility and sustainability, in: Rami Bahsoon (Ed.), Proceedings of the 10th European Conference on Software Architecture Workshops, in: ECSAW '16, ACM, Copenhagen, Denmark, 2016, p. 7:1, http://dx.doi.org/10.1145/2993412.3014339.