A systematic review on the use of Definition of Done on agile software development projects

Conference Paper · June 2017

DOI: 10.1145/3084226.3084262

7 authors, including:

Mirko Perkusich
VIRTUS
120 PUBLICATIONS **1,039** CITATIONS
SEE PROFILE

Ana Silva
Federal University of Campina Grande
3 PUBLICATIONS **45** CITATIONS
SEE PROFILE

Thalles Araújo
Instituto Federal de Educação Ciência e Tecnologia da Paraíba
2 PUBLICATIONS **43** CITATIONS
SEE PROFILE

Ednaldo Dilorenzo
Instituto Federal de Educação Ciência e Tecnologia da Paraíba
10 PUBLICATIONS **238** CITATIONS
SEE PROFILE

A systematic review on the use of Definition of Done on agile software development projects

Ana Silva
Federal Institute of Paraíba
PB-264
Monteiro, Paraíba, Brazil 58500-000
anasilva.ifpb@gmail.com

Thalles Araújo
Federal Institute of Paraíba
PB-264
Monteiro, Paraíba, Brazil 58500-000
thalleshenrique.na@gmail.com

João Nunes
Federal Institute of Paraíba
PB-264
Monteiro, Paraíba, Brazil 58500-000
lockenunes@gmail.com

Mirko Perkusich
Federal University of Campina Grande
882 Aprigio Veloso street
Campina Grande, Paraíba, Brazil 58429-900
mirko.perkusich@embedded.ufcg.edu.br

Ednaldo Dilorenzo
Federal Institute of Paraíba
PB-264
Monteiro, Paraíba, Brazil 58500-000
ednaldo.dilorenzo@ifpb.edu.br

Hyggo Almeida
Federal University of Campina Grande
882 Aprigio Veloso street
Campina Grande, Paraíba, Brazil 58429-900
hyggo@embedded.ufcg.edu.br

Angelo Perkusich
Federal University of Campina Grande
882 Aprigio Veloso street
Campina Grande, Paraíba, Brazil 58429-900
perkusic@embedded.ufcg.edu.br

ABSTRACT

Background: Definition of Done (DoD) is a Scrum practice that consists of a simple list of criteria that adds verifiable or demonstrable value to the product. It is one of the most popular agile practices and assures a balance between short-term delivery of features and long-term product quality, but little is known of its actual use in Agile teams.

Objective: To identify possible gaps in the literature and define a starting point to define DoD for practitioners through the identification and synthesis of the DoD criteria used in agile projects as presented in the scientific literature.

Method: We applied a Systematic Literature Review of studies published up to (and including) 2016 through database search and backward and forward snowballing.

Results: In total, we evaluated 2326 papers, of which 8 included DoD criteria used in agile projects. We identified that some studies presented up to 4 levels of DoD, which include story, sprint, release or project. We identified 62 done criteria, which are related to software verification and validation, deploy, code inspection, test process quality, regulatory compliance, software architecture design, process management, configuration management and non-functional requirements.

Conclusion: The main implication for research is a need for more and better empirical studies documenting and evaluating the use of the DoD in agile software development. For the industry, the review provides a map of how DoD is currently being used in the industry and can be used as a starting point to define or compare with their own DoD definition.

CCS CONCEPTS

•Software and its engineering →Agile software development;

KEYWORDS

Agile Software Development, Definition of Done, Systematic Literature Review

ACM Reference format:
Ana Silva, Thalles Araújo, João Nunes, Mirko Perkusich, Ednaldo Dilorenzo, Hyggo Almeida, and Angelo Perkusich. 2017. A systematic review on the use of Definition of Done on agile software development projects. In *Proceedings of Evaluation and Assessment in Software Engineering conference, Kalskrona, Sweden, June 2017 (EASE'17),* 10 pages.
DOI: 10.475/123_4

EASE'17, Kalskrona, Sweden

DOI: 10.475/123_4

1 INTRODUCTION

Scrum is the most popular agile method [13]. It is a framework to manage agile projects and defines artifacts, practices and roles. To promote continuous assessment of the quality of the product and guarantee incremental delivery, Scrum presents the practice Definition of Done (DoD). It consists of a set of criteria to define

if a deliverable is done. In other words, it defines the minimal restrictions that must be fulfilled for a product to be released.

According to Sutherland and Schwaber [11], DoD promotes transparency between the stakeholders on the meaning of completing work and is a key concept in Scrum. It It can significantly vary between different teams, because it should be defined given their context. According to Williams [15], in which data from 326 practitioners were analyzed, DoD is the most popular agile practice along with short iterations and continuous integration.

On the other hand, Williams [15] concluded that the principles in the Agile Manifesto are not enough to emphasize the need to produce high quality systems and the importance of non-functional requirements. As a consequence, the focus on fast delivery of customer value in a short period of time causes technical debt, which will, as time progresses, slow the progress.

The problem was aggravated with the popularity of Scrum, which is not composed of quality management techniques such as test automation, continuous integration and pair programming. This is not a failure, but a characteristic: Scrum is a framework that must be complemented with technical and managerial processes.

The use of Scrum with focus only on managerial activities is, as described by Martin Fowler, an influential Software Engineering practitioner, *Flaccid Scrum*. In a study performed with 18 teams of 11 companies, Eloranta et al. [6] identified that one of the most common anti-pattern on Scrum projects is the lack of testing and test automation during the iteration in which the code was produced. As a consequence, defects that were introduced into the code are not detected and the code evolves on top of code with a high probability of containing defects, which improves technical debt. Furthermore, delaying activities related to code quality is an obstacle for decision making in agile software development [4]. These issues can be avoided by an appropriate use of the DoD.

Even though DoD is a key concept, to the best of our knowledge, there is no study that explored which criteria are being used in the context of agile software development. According to Dybå and Dingsøyr [5], there is a need of research to assess which agile practices proposed by practitioners are really useful. In this context, the main goal and contribution of this paper is to report the state of the are in the field of DoD in agile software development by means of a systematic literature review. We followed the guidelines proposed by Kitchenham and Charters [8] and Wohlin [16]. In this paper, we detail our study and also point the gaps and future directions for research in DoD for agile software development.

This article is structured as follows. In Section 2, we present the protocol of our systematic review process. In Section 3, we present our findings. In Section 4, we discuss the results. In Section 5, we present our conclusions and recommendations for future research.

2 REVIEW METHOD

According to Kitchenham and Charters [8], systematic review is an evidence-based technique that uses a well-defined, unbiased and repeatable methodology to identify, analyze and interpret all the relevant papers related to a specific research question, subject area, or phenomenon of interest. A key component of this technique is the protocol, which is a plan that describes the conduct of the systematic review. It includes the research questions, search process, selection process and data analysis procedures. A summary of the protocol used in this review is given in the following sub-sections.

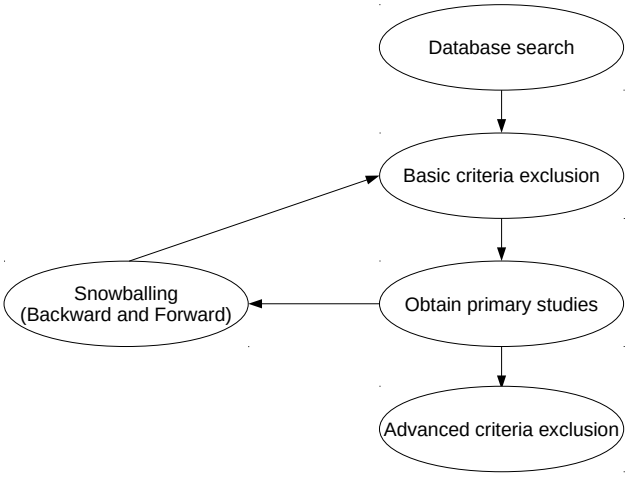

Figure 1: Overview of the search and selection process

2.1 Research questions

The aims of this research are to identify possible gaps in the literature and define a starting point to define DoD for practitioners through the identification and synthesis of the DoD criteria used in agile projects as presented in the scientific literature. Given this, we formulated the following research questions (RQs):

RQ1 What are the done criteria used in agile software development projects?

RQ2 What are the characteristics of the application domain of the papers that report the done criteria identified in RQ1?

RQ3 What types of studies are performed in the papers the report the done criteria identified in RQ1?

2.2 Search strategy

To minimize the probability of missing relevant papers, we used a hybrid search strategy, which is based on database search and snowballing (backward and forward). First, we defined a search string, which is presented in Section 2.2.1 and used it to search databases containing scientific papers in the context of Computer Science, as shown in Section 2.2.2. After applying the basic criteria exclusion, shown in Section 2.3, the resulting papers were defined as the starting (i.e., seed) set for the snowballing. After executing the snowballing iterations, we applied the advanced criteria exclusion, which is related to the actual data extraction and quality assessment. We show an overview of the search and selection process in Figure 1.

We decided to use this strategy to avoid missing papers due to limitations and inconsistencies of digital libraries. They have different formats to handle the Boolean expressions, as discussed in Brereton et al. [3], and we were not sure how reliable is their ability to handle searches with long strings. Finally, there are evidences in the literature of the risks of missing papers using only one approach, as discussed in Badampudi et al. [2] and Felizardo et al. [7].

2.2.1 Search terms. Starting with the research questions, suitable keywords were identified using synonyms and related terms. The following keywords were used to formulate the search string:

- **Population:** Agile software development. *Alternative keywords:* Scrum, Extreme Programming, Feature-Driven Development, Dynamics Systems Development Method, Kanban, Crystal, Lean software development and DevOps.
- **Intervention:** Definition of done. *Alternative keywords:* Done criteria, DOD and definition of ready.
- **Context:** Industry or academia. Our target population was papers performed in the industry or academy and we intended to capture papers in that context regardless of the type of research performed. Not used in the search string.

To define a first version of the search string, the keywords within a category were joined by using the Boolean operator 'OR' and the two categories were joined using the Boolean operator 'AND'. This was done to target only papers in the context of agile software development related to definition of done. To simplify the strings and include additional synonyms and Portuguese translations, we defined the following search string:

("definition of ready" OR "definition of done" OR "done criteria" OR "dod" OR "definição de preparado" OR "definição de pronto") AND (software AND (agile OR ágil) AND (scrum OR xp OR (crystal AND (clear OR orange OR red OR blue)) OR dsdm OR fdd OR "feature driven development" OR (lean AND (development OR desenvolvimento)) OR Kanban OR "extreme programming" OR "programação extrema" OR devops))

2.2.2 Data sources. Since our goal is to cover the literature published in Computer Science, we chose the following digital databases for data retrieval:

- ACM Digital Library
- Engineering Village
- Science Direct
- Scopus
- Springer
- Web of Science
- Wiley Online Library

We did not include IEEExplore because it could not handle our search string due to its size. On the other hand, Scopus, Engineering Village and Web of Science indexes IEEE papers.

2.3 Selection criteria

Before applying the selection criteria given the topic of the review, we defined a generic exclusion criteria:

- Published in non-peer reviewed publication channel such as books, thesis or dissertations, tutorials, keynotes, etc. OR
- Not available in English or Portuguese language OR
- A duplicate.

We implemented the first two criteria in the search strings that were executed in the digital libraries, wherever possible. For the third criteria, since there are intersections of the coverage of journal and conferences by the libraries, we implemented a script to automatically remove papers with the same title, authors and abstract.

Afterwards, the remaining papers were evaluated through two sets of selection criteria: basic and advanced.

2.3.1 Basic criteria. We applied the basic criteria to evaluate if papers are relevant to the aims of our paper by reading the titles and abstract. This criteria was applied on papers that passed the generic exclusion criteria and were identified through database search or snowballing. In this context, we included papers that:

- Are related to agile software development AND
- Are related to Definition of Done.

To evaluate if a paper was related to definition of done, we were not rigorous. Even if the term was not included in the title or abstract, we still included the paper if it presented a case (e.g., company or project) in which an agile process was applied due to the chance of the description present examples of done criteria.

We excluded papers that:

- Are not related to agile software development OR
- Are not related to Definition of Done.

Following the procedure presented in Ali et al. [1], we decided that papers are classified as: Relevant, Irrelevant or Uncertain (in the case the available information on the title and abstract is inconclusive). Each paper was evaluated by two reviewers, which were chosen randomly. Given this, there are six possibilities of agreement or disagreement between the reviewers, as shown in Figure 2.

		Reviewer 2		
		Relevant	Uncertain	Irrelevant
Reviewer 1	Relevant	A	B	D
	Uncertain	B	C	E
	Irrelevant	D	E	F

Figure 2: Paper selection possibilities.

Papers evaluated as A or B are selected for inclusion in the next step of the paper, because at least one reviewer evaluated them as relevant. Therefore, since we used a "when in doubt, include" approach, we include them. Papers evaluated as F are excluded, as both reviewers agreed on their irrelevance.

Papers evaluated as C were further reviewed independently by both reviewers using the steps of adaptive reading, as described below. With this practice, we collect more details from the paper to assist on decision making.

Papers evaluated as D and E show disagreement, in which category D is the worst, because one author considers the paper irrelevant and the other considers it relevant. Papers in these two categories were discussed by the reviewers. As a result, through a consensus, the papers were classified as category A, C or F.

The adaptive reading process is composed of three steps:

(1) read the Introduction,
(2) if not having agreement in 1, read conclusion,
(3) if not having agreement in 2, use the keywords to evaluate their usage to describe the context of the paper.

2.3.2 Advanced criteria. The advanced criteria is related to the actual data extraction, in which the full-text of the papers was read by two reviewers independently. We used the same criteria used in the previous steps (see Section 2.3.1), but reading the full-text. We excluded studies published in multiple papers, only including the

extended version of the study. Additionally, we excluded papers that did not contain relevant information for RQ1. In other words, a paper was only included if it contained examples of done criteria.

Following the advice presented in Brereton et al. [3] and Staples and Niazi [10], each paper was evaluated by one data extractor and one data checker. The data extractor completed the extraction form and the data checker confirmed that the data on the extraction form were correct. Any disagreement were discussed and an agreed final data value recorded. Details regarding the quality assessment and data extraction process are presented, respectively, in Section 2.5 and Section 2.6.

2.4 Snowballing

The snowballing approach was, first, performed on the set of papers identified through the database search and included using the basic criteria. For each paper in the set, we applied the backward and forward snowballing.

To execute the forward snowballing, we used Scopus and Google Scholar to identify the title and abstract of the papers citing our set of selected papers. We applied the basic criteria, shown in Section 2.3.1, to include these papers, in which each paper was evaluated by two reviewers.

To execute the backward snowballing, first, we distributed the papers to be evaluated by only one reviewer. The reviewer was responsible to apply the generic exclusion criteria shown in Section 2.3. This was done by evaluating the title in the reference list and, if necessary, the place of reference in the text. Afterwards, the included studies were evaluated using the basic criteria, in which each paper was assessed by two reviewers.

We decided not to follow the guidelines for snowballing presented in Wohlin [16], in which it is recommended that the final inclusion of a paper should be done based on the full paper, because our research questions are too specific. Therefore, it is possible that papers that do not present done criteria are cited by or reference papers by relevant papers. Instead, we used the basic criteria (shown in Section 2.3).

2.5 Quality assessment

The quality assessment of this research followed eleven criteria established by Dybå and Dingsøyr [5]. These criteria are listed below:

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

To evaluate each of the criteria listed above, we used a dichotomous scale, receiving a score of "1" for "yes" and "0" for "no". As discussed in Section 2.3.2, each paper was evaluated by a data extractor and a data checker.

2.6 Data extraction

We used a spreadsheet editor to record relevant information. With the spreadsheet, we were able to map each datum extracted with its source. From each paper, we extracted general information such as year and name of the publication channel and data related to the RQs. The following data were extracted from the papers:

(i) type of article (journal, conference),
(ii) name of the publication channel,
(iii) year of publication,
(iv) agile method,
(v) product domain,
(vi) application domain,
(vii) team size,
(viii) team distribution,
(ix) number of cases,
(x) research type,
(xi) research question type,
(xii) empirical research type,
(xiii) research validation,
(xiv) definition of done levels,
(xv) done criteria.

For question (x), we used the classification presented by Wieringa et al. [14]: validation research, evaluation research, solution proposal, philosophical papers, opinion papers or experience papers. For (xi), we used the classification presented by Shaw [9]: method or means of development; method for analysis or evaluation; design, evaluation, or analysis of a particular instance; generalization or characterization; or feasibility study or exploration. For (xii), we used the classification presented by Tonella et al. [12]: experiment, observational study, experience report, case study or systematic review. For (xiii), we used the classification scheme presented by Shaw [9]: analysis, evaluation, experience, example, persuasion or blatant assertion. The data collected for questions (xiv) and (xv) were synthesized to answer RQ1.

3 RESULTS

In this section, we present the results for the systematic review process and for the research questions as well. In Figure 3, we present an overview of the number of studies passing through the different stages of the study. We show details of the results of the database search in Figure 4 and of the results of the snowballing process, in which we iterated twice, in Figure 5 and Figure 6. We present the list of the included studies in Table 1.

In Figure 7, we show the number of papers per year. In Figure 8, we show the distribution of papers per type of publication channel. In Figure 9, we show the aggregated results of the quality assessment. By analyzing Figure 9, we concluded that the quality of the studies from an evidence-based perspective is low. On

Table 1: Overview of the selected studies.

Paper Number	Authors	Year	Title	Publication channel
P1	Eloranta, V., Koskimies, K., Mikkonen, T.	2016	Exploring ScrumBut—An empirical paper of Scrum anti-patterns	Information and Software Technology
P2	Vlietland, J., Solingen R. V., Vilet, H. V.	2016	Aligning codependent Scrum teams to enable fast business value delivery: A governance framework and set of intervention actions	The Journal of Systems and Software
P3	Rindell, K., Hyrynsalmi, S., Leppänen, V.	2015	Securing Scrum for VAHTI	Symposium on Programming Languages and Software Tools
P4	Davis, N.	2013	Driving quality improvement and reducing technical debt with the definition of done	Agile Conference
P5	O'Connor, C. P.	2010	Letters from the edge of an agile transition	ACM International Conference Companion on Object Orientad Programming Languages and Applications Companion
P6	Igaki, H., Fukuyasu, N., Saiki, S., Matsumoto, S., Kusumoto, S.	2014	Quantitative assessment with using ticket driven development for teaching scrum framework	International Conference on Software Engineering Companion
P7	Gupta, R. K., Manikreddy, P. Naik, S., Arya, K.	2016	Pragmatic Approach for Managing Technical Debt in Legacy Software Project	India Software Engineering Conference
P8	Saddington, P.	2012	Scaling product ownership through team alignment and optimization	Agile Conference

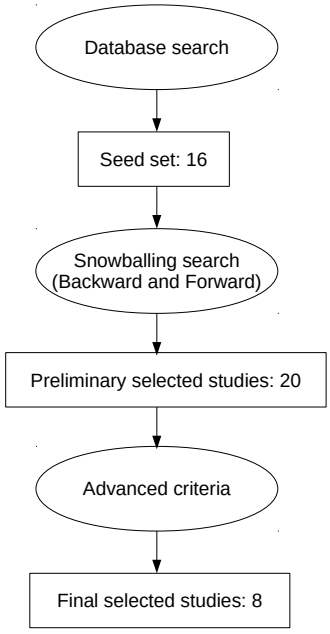

Figure 3: Number of papers in study selection.

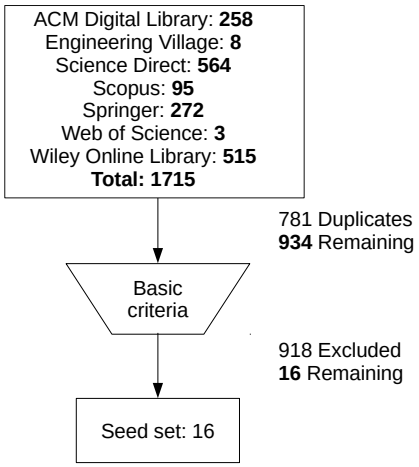

Figure 4: Overview of the database search.

the other hand, due to the topic of the study, it is acceptable to include industrial experience reports with the risk of less reliable conclusions.

3.1 RQ1: Done criteria

In this section, we show the done criteria identified in the included papers. A total of 62 criteria were identified. Three papers, out of the eight included papers, applied multilevel DoD, as shown in Table 2. In Table 2, P4 is presented twice, because it presents two cases with different done criteria. Furthermore, in Table 2, we present the done criteria exactly as they are presented in the papers.

By analyzing Table 2, it is possible to conclude that there are different types of criteria such as activity (e.g., peer code review), metrics (e.g., localization defect density), targets (e.g., product committed to VCS), standards (e.g. coding standards) and checklist (e.g.,

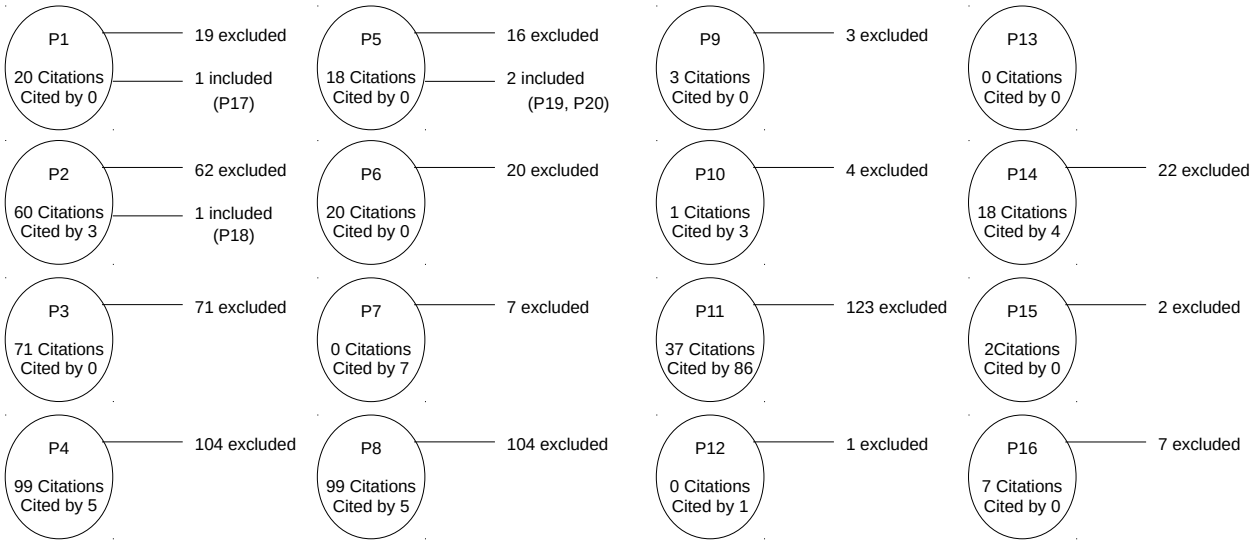

Figure 5: Results of the first snowballing iteration.

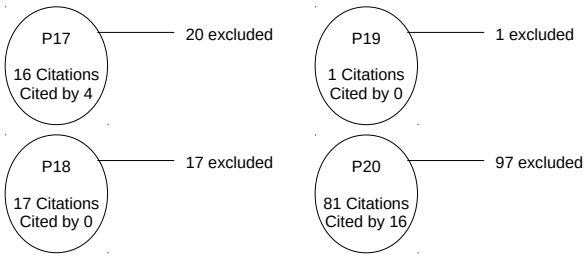

Figure 6: Results of the second snowballing iteration.

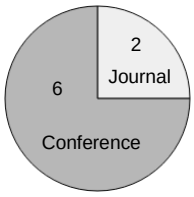

Figure 8: Distribution of papers per type of publication channel.

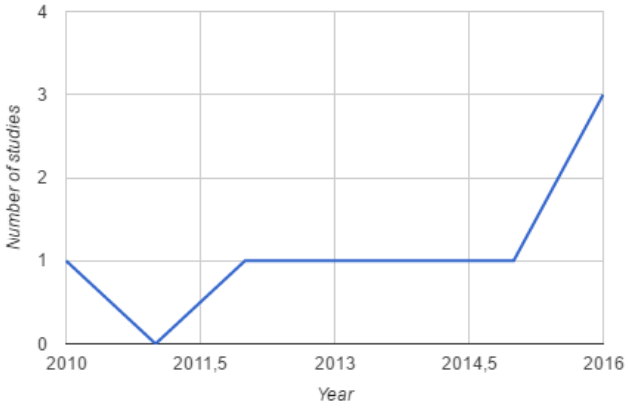

Figure 7: Number of papers per yer.

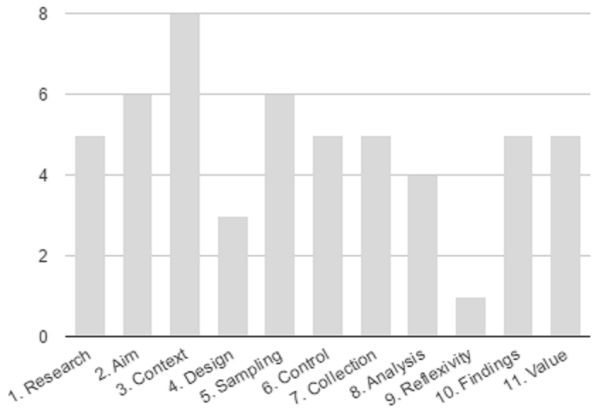

Figure 9: Results of the quality assessment.

design review checklist). To categorize the identified criteria, we normalized them by relating them to activities. For instance, in study P7, the criteria *No open static analysis errors*, was associated with the activity *Static code analysis*. Afterwards, we categorized

Table 2: Done criteria identified.

Paper Number	Number of levels	Done criteria
P1	1	Unit Tested, Integration Tested, Acceptance tested, Delivered and installed.
P2	1	Functionality of a story worked in accordance with the feature stories, including the integration with the application connectivity. System tested, including interfacing and middleware testing
P3	3 (basic, heightened and high)	**Basic**: Security training, Additional security training, Application risk analysis, Test plan review, Threat modeling, threat modeling updates. **Heightened**: Goal and criticality definition, Business impact analysis, Documentation of security solutions, Architecture security requirement analysis, Security auditing, Security testing, Application security setting definition. **High**: Architectural and development guidelines, External interfaces review, Attack surface recognition and reduction, Architectural security requirements, Internal communication security, Security test cases review, Test phase code review, Use of automated testing tools, Security mechanism review, Development time auditing.
P4	3 (story, sprint and release)	**Story**: Design (according to design standards), Design peer review (using design review checklist), Code and unit Test (according to coding standards), Code peer review (using code review checklist), Acceptance test written (goal of 100% automation), Acceptance test peer review (using test review checklist). **Sprint and Release**: Localization Defect Density, Performance Test, System Test, Static Analysis (using organizational rules), Security Review
P4	4 (story, sprint, release and project)	**Story**: Development planning, Requirements analysis, Architectural design, Detailed design, Verify detailed design, Unit test implementation and verification, Code verification, Integration and integration testing and Systems testing. **Sprint**: Development planning, Integration and integration testing, System testing and regression testing **Release**: Development planning, Integration and integration testing, System testing and regression testing and release. **Project**: Development planning, Risk management planning, Requirements analysis, Risk control measures, Re-evaluate medical device risk analysis, Verify software requirements, Architectural design, Identify segregation for risk control, Verify software architecture.
P5	1	Acceptance tests
P6	1	Product (source code, test code or review reports) committed to VCS, Source code is reviewed by a different person, source code in VCS must be able to be built normally after committing by developers, the number of assigned task for a developer should be within +-20% of the average of all the assigned task of the given type (source code, unit code, review), unit tests completed.
P7	1	No open static analysis errors, No open memory leak violation, No degradation in performance compared to baseline, Reviewed by experts.
P8	2 (sprint and release)	Functionality complete, has been through technical design following a code quality checklist (not available), new service APIs defined, services integrated, meets all acceptance criteria for user story, UI conforms to approved wireframes and Visual Comps/Red Lines, committed to SVN against the Jira ID for the User Story, Build is successful after code committed, all errors generated within Flex PMD are removed, code is run through Flex Formatter to correct any formatting issues, Tech Design Checklist is reviewed and completed, Unit Testes written and verified, Code conforms to Code Standards Guide (not available), Code Peer Reviewed following Code Review guidelines (not available), Functional testes written, Functional testes reviewed by PO, PO accepts story as complete, Jira is updated for story, Confluence is updated, All DB scripts for story must be documented, packaged and reviewed by DBA.

each activity given its end goal. For instance, *Unit test* was associated with *Software verification and validation* and *Static code analysis* with *Code inspection*. We defined a separate category for *Non-functional requirements*-related activities, because they require specialized testing. A special case is for standards compliance. In P3, the criteria were defined to comply with VAHTI, a Finish governmental security standard collection. In P4, case 2, the criteria were defined to comply with FDA IEC 62304 and TIR45, which are in the context of medical device software industry. In both cases, we associated their unique activities to *External standards/regulatory compliance*. In Figure 10, we present the distribution of the criteria given the identified activities (i.e., categories). Notice that each criteria was only counted once, even if it was presented by several papers. In Figure 11, we present the frequency in which a criteria was presented in the papers. Notice that since P4 presents two cases, it is possible that a criteria was counted twice for it. Furthermore, only criteria that were presented in at least two studies are presented in Figure 11.

3.2 RQ2: Characteristics of application domain of DoD

Since the application of DoD depends on the context of the project, to analyze the done criteria reported in a given study, it is important to analyze its context. Therefore, we collected data regarding the agile method used, product domain, application domain (i.e., industry or academic), team size, team distribution and the number

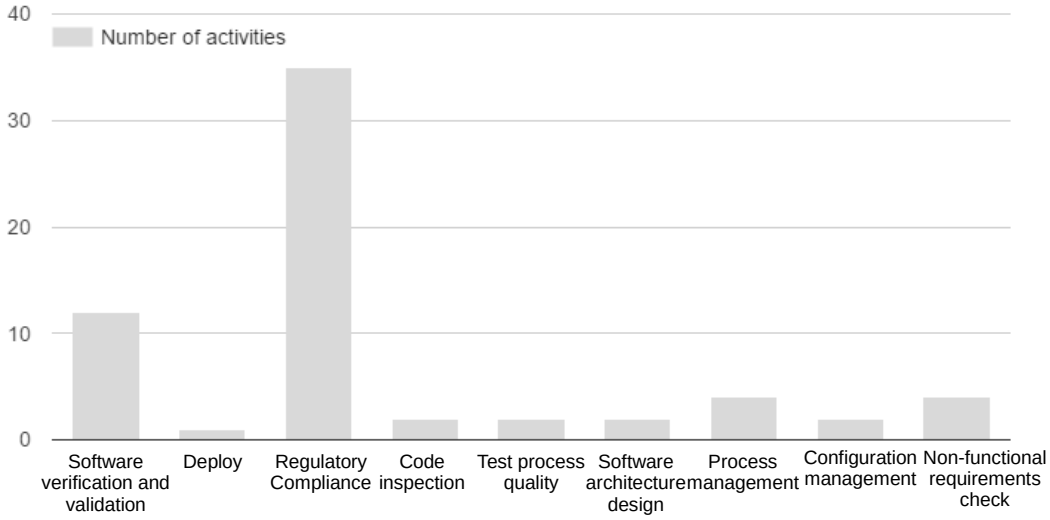

Figure 10: Distribution of criteria per category.

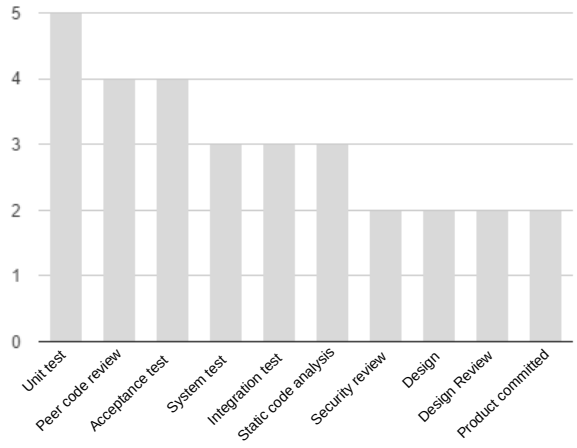

Figure 11: Frequency of criteria.

of cases. In Table 3, we show the collected data for each primary study.

3.3 RQ3: Characteristics of the studies performed

This RQ is designed to identify the characteristics of the studies that reported done criteria. The design of the study along with the quality assessment data can be used to infer the scientific relevance of the findings. In Table 4, we show the collected data for each primary study, following the criteria presented in Section 2.6.

4 DISCUSSION

The results of this study address three research questions (see Section 2.1), which explore the done criteria used in agile software development projects. We observed that the reporting of done criteria is recent. The first paper was published in 2010.

As shown in Figure 9, the average quality of the studies, in the perspective of evidence-based science, is low. This is because most papers were written, apparently, by practitioners presenting an experience of applying the Definition of Done, along with other practices, on agile software development projects.

On the other hand, even though we only identified 8 papers, we identified 62 criteria, which are applied with different goals, as shown in Figure 10. Since seven, out of eight, of the papers are in the context of Scrum, we will discuss the different use of done criteria in this context.

Since Scrum is a framework to manage agile projects, it requires to be complemented with technical and managerial practices. P6 and P8 define done criteria to complement Scrum with configuration management practices. They define a criteria to assess if the code is committed to VCS. P6 also verifies if, in the case of continuous integration, a successful build is generated by a developer's commit. P8 presents criteria to assess the management process: update project management tools and documentation. P6 presents criteria to assist on task management and avoid unequal distribution of effort. This shows the potential of using DoD to, given the context of the project (e.g., team experience), define a checklist of good process practices to be followed such as committing, branching and issue tracking.

P4 and P7 define criteria to check non-functional requirements. In the context of P4, criteria to check localization, performance and security were defined. In P7, they considered performance and memory leaks. In both cases, they present metrics and targets for each non-functional requirement. The targets are used to assist on deciding if a given story, sprint or release is done. In other contexts, we assume that it could be valuable to use DoD to assess other non-functional requirements such as battery consumption, for embedded systems, and interoperability, for web applications that must be accessed by others via APIs.

Table 3: Overview of the selected studies.

Paper Number	Agile method	Product domain	Application domain	Team size	Team distribution	Number of cases
P1	Scrum	Not presented	Industry	Various	Not defined	Multiple
P2	Scrum	Banking	Industry	Multiple codependent teams	Not defined	1
P3	Scrum	Not presented	None	None	None	None
P4	Scrum	Not presented	Industry	8	Collocated	2
P5	Non specific	Not presented	Industry	62	Collocated	1
P6	Scrum	Web application	Academic	9 teams of 5-6	Collocated	1
P7	Scrum	Web and mobile	Industry	Teams of 7-8	Distributed	1
P8	Scrum	Not presented	Industry	Multiple	Collocated	1

Table 4: Characteristics of the studies performed.

Paper Number	Research type	Research question type	Empirical Research type	Research validation
P1	Experience papers	Generalization or characterization	Observational study	Evaluation
P2	Solution proposal	Method or means of development	Case study	Analysis
P3	Experience papers	None	None	Experience
P4	Experience papers	None	None	Blatant assertion
P5	Solution proposal	Method or means of development	Case study	Experience
P6	Solution proposal	Method or means of development	None	Experience
P7	Solution proposal	Method or means of development	None	Experience P8
Experience papers	None	None	Experience	

P4 shows criteria for software architectural design. In their case, they noticed that they had to redo a lot of work due to poor design. Therefore, they defined a project design standard and a design review checklist to decrease the chance of wasting effort redoing work. Depending on the experience of the team and the project complexity, this might be necessary. In any case, it is an example of how DoD can complement an agile process to be more proactive.

P1, P2, P4, P5, P6, P7 and P8 present criteria to assist on quality management. P4, P6, P7 and P8 use peer code review as a gate that all code must pass before being considered done. They recommend using a checklist to provide a systematic way of reviewing code. Having a coding standard is also recommended. P4, P7 and P8 use targets regarding the number of open static analysis violations. P1, P2, P6 and P8 defines unit tests as an activity that must be fulfilled to consider a story as done. P1, P2, P4 and P5 considers that acceptance test must be passed to consider a story as done. Finally, P4 presents criteria to assess the quality of the test process, in which they measure the percentage of automated test, with a goal of 100% of automation. Therefore, there are several examples of how DoD can be used to complement the acceptance criteria of user stories and be used as a factor to consider them as done.

P3 and P4 show how Scrum can be complemented to comply with external standards that were defined for traditional waterfall models. In P3, Scrum is complemented with 23 security-related activities to comply with VAHTI, which is a document-driven Finish software security standard. In P4, Scrum is complemented to comply with FDA IEC 62304 and TIR45, which are in the context of medical device software industry. In this case, we can conclude that, given the context of the product being developed, agile processes can use the DoD to complement them and comply with external requirements such as for ISO 9001 and CMMI audits, for instance.

Furthermore, P3, P4 and P8 show cases in which multiple levels of done were used. In P3, the levels were defined given the VAHTI documentation. In P4 and P8, the levels were defined given the main output of the project. A project is composed of releases. Within a release, sprints are executed. As a result of sprints, stories are delivered. The number of levels is a function of the complexity of the process required to deliver the project. For instance, in P4, they initially used three levels of done: story, sprint and release. On the other hand, to adapt their process to be compliant with external standards, they added a new level: project.

As shown in Figure 11, out of the 62 criteria, only one was present in 5 studies: unit test. Only two were present in 4 studies:

peer code review and acceptance test. Only three was present in 3 studies: system test, integration test and static code review. This disagreement is expected, because, as claimed by the Scrum Guide, the DoD should be defined by the team according to its context. On the other hand, the data shows motivations to apply DoD.

Regarding the consequences of using DoD, in P8, the authors claim that using the presented DoD, along with other interventions, improved the collaboration between the teams and they reached a 130% decrease in defects. P7 documents that the constraints in the process promoted by the DoD also promoted collaboration, as the members were forced to rotate between different types of tasks (source code, unit tests and code review). P6 presents data on how the increased productivity and reduced technical debt. P4 reduced from 278 to 12 defects deferred, reduced the percentage of reopened defects from 20% to less than 1% and improved the team net promoter score from 63% to 70%. Even though the scientific rigor of some of the studies is not high, we believe that these data can be used as an indicator that the correct use of DoD can help agile teams to work better.

5 THREATS TO VALIDITY

A potential threat to validity is, as for any systematic literature review, if we were able to cover all primary studies. We aimed to construct a comprehensive string, covering key terms and their synonyms and applied the search strings to multiple databases that cover major Software Engineering conferences and journals. Furthermore, we complemented the database search with forward and backward snowballing. In regard to the quality of the selection of the study and data extraction, we used a systematic approach in which each paper was evaluated by at least two reviewers, to avoid reviewer bias and human errors.

6 CONCLUSION AND FUTURE WORK

This study presents a systematic literature review on definition of done in agile software development. We used a hybrid approach composed of database search and snowballing. The primary search fetched 934 unique results, from which 16 papers were selected as a seed set for snowballing. In the snowballing, which was composed of 2 iterations, we evaluated an additional 611 papers, in which 4 were added to our set of selected papers. After quality assessment and data extraction, only 8 papers were included in the study. Data from these papers were analyzed to answer each research question.

There is a variety of done criteria used in agile software development. Furthermore, some papers used a multilevel approach to manage DoD, which include story, sprint, release or project. We categorized the criteria into: software verification and validation, deploy, code inspection, test process quality, regulatory compliance, software architecture design, process management, configuration management and non-functional requirements. There was not a high rate of agreement of the done criteria, which is expected since they should be defined given the context of the project.

All papers presented activities that must be fulfilled to consider a story, sprint, release or project done. On the other hand, only three papers presented metrics and targets to assess that the activities are in conformance to the expected criteria. Finally, seven, out of the eight papers, are in the context of Scrum.

Practitioners can use the results of this study as a guide for them to apply DoD on their projects or compare with their own DoD definition. Moreover, based on the results of this study, we recommend that there is strong need to publish more papers presenting how DoD is applied in agile projects and to conduct empirical studies to assess the results of applying this practice. For future work, we intend to execute a survey with practitioners and compare the collected data with our results in this study. Furthermore, we will broaden the research aims by exploring the effectiveness of the claimed benefits, challenges on using DoD, factors that influence on defining the criteria and compare the state-of-practice with the authoritative guidelines and textbooks.

REFERENCES

[1] Nauman Bin Ali, Kai Petersen, and Claes Wohlin. 2014. A Systematic Literature Review on the Industrial Use of Software Process Simulation. *J. Syst. Softw.* 97, C (Oct. 2014), 65–85. DOI: http://dx.doi.org/10.1016/j.jss.2014.06.059

[2] Deepika Badampudi, Claes Wohlin, and Kai Petersen. 2015. Experiences from Using Snowballing and Database Searches in Systematic Literature Studies. In *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering (EASE '15)*. ACM, New York, NY, USA, Article 17, 10 pages. DOI: http://dx.doi.org/10.1145/2745802.2745818

[3] Pearl Brereton, Barbara A. Kitchenham, David Budgen, Mark Turner, and Mohamed Khalil. 2007. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software* 80, 4 (2007), 571 – 583. DOI: http://dx.doi.org/10.1016/j.jss.2006.07.009

[4] Meghann Drury, Kieran Conboy, and Ken Power. 2012. Obstacles to Decision Making in Agile Software Development Teams. *J. Syst. Softw.* 85, 6 (June 2012), 1239–1254. DOI: http://dx.doi.org/10.1016/j.jss.2012.01.058

[5] Tore Dybå and Torgeir Dingsøyr. 2008. Empirical studies of agile software development: A systematic review. *Information and Software Technology* 50, 9–10 (2008), 833 – 859. DOI: http://dx.doi.org/10.1016/j.infsof.2008.01.006

[6] Veli-Pekka Eloranta, Kai Koskimies, and Tommi Mikkonen. 2016. Exploring ScrumBut—An empirical study of Scrum anti-patterns. *Information and Software Technology* 74 (2016), 194 – 203. DOI: http://dx.doi.org/10.1016/j.infsof.2015.12.003

[7] Katia Romero Felizardo, Emilia Mendes, Marcos Kalinowski, Érica Ferreira Souza, and Nandamudi L. Vijaykumar. 2016. Using Forward Snowballing to Update Systematic Reviews in Software Engineering. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '16)*. ACM, New York, NY, USA, Article 53, 6 pages. DOI: http://dx.doi.org/10.1145/2961111.2962630

[8] B. Kitchenham and S. Charters. 2007. *Guidelines for Performing Systematic Literature Reviews in Software Engineering.* Technical Report EBSE-2007-01. Schoolf of Computer Science and Mathematics, Keele University.

[9] M. Shaw. 2003. Writing good software engineering research papers. In *25th International Conference on Software Engineering, 2003. Proceedings.* 726–736. DOI: http://dx.doi.org/10.1109/ICSE.2003.1201262

[10] Mark Staples and Mahmood Niazi. 2007. Experiences Using Systematic Review Guidelines. *J. Syst. Softw.* 80, 9 (Sept. 2007), 1425–1437. DOI: http://dx.doi.org/10.1016/j.jss.2006.09.046

[11] Jeff Sutherland and Ken Schwaber. 2013. The Scrum Guide. http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-US.pdf. (2013). Accessed: 2016-03-02.

[12] Paolo Tonella, Marco Torchiano, Bart Du Bois, and Tarja Systä. 2007. Empirical studies in reverse engineering: state of the art and future trends. *Empirical Software Engineering* 12, 5 (2007), 551–571. DOI: http://dx.doi.org/10.1007/s10664-007-9037-5

[13] VersionOne. 2016. 10th Annual State of Agile Development Survey Results. (2016).

[14] Roel Wieringa, Neil Maiden, Nancy Mead, and Colette Rolland. 2006. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. *Requirements Engineering* 11, 1 (2006), 102–107. DOI: http://dx.doi.org/10.1007/s00766-005-0021-6

[15] Laurie Williams. 2012. What Agile Teams Think of Agile Principles. *Commun. ACM* 55, 4 (April 2012), 71–76. DOI: http://dx.doi.org/10.1145/2133806.2133823

[16] Claes Wohlin. 2014. 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)*. ACM, New York, NY, USA, Article 38, 10 pages. DOI: http://dx.doi.org/10.1145/2601248.2601268