Identifying Conditions for Effective Communication with Just Enough Documentation in Continuous Software Development

Theo Theunissen

HAN University of Applied Sciences, Arnhem, the Netherlands
Theo.Theunissen@han.nl

Abstract. Modern development methods like Lean, Agile, and DevOps have characteristics in common concerning avoiding waste, delivering working software, and continuous delivery. We call these collective methods Continuous Software Development (CSD). Additionally, we take into account the full lifecycle of software development, including continuous architecting and operations. Typically, in CSD processes, one can observe a lower amount of documentation as well as a lower quality of process-internal information. Stakeholders start development with just enough, and informal, documentation through sketches. In combination with face-to-face communication, this is considered sufficient. Nevertheless, having no documentation at all is not an option. Developers and other stakeholders at least require a minimal amount of documentation which is of acceptable quality. In addition, information is scattered throughout tools in the software development ecosystem. An investigation is therefore conducted as to how this information can be organized into comprehensible documentation. The contribution of this research project primarily consists of an empirically validated and theoretically founded documentation framework that is streamlined for CSD.

Keywords: Agile · Continuous Software Development · DevOps · Documentation · Lean

1 Introduction

In recent years, we have seen widespread adoption of software development methods like Lean, Agile and DevOps. The term Continuous Software Development (CSD) is used to collectively refer to these software development methods, covering the entire development lifecycle. The main characteristics of CSD are:

1. It covers **values, principles, practices, tools, and processes** from Lean [15], Agile [5] and DevOps [9].
2. It covers **all phases** of the software development lifecycle, from concept to end-of-life. Lean [15] and Agile [5] typically focus on the project phases,

excluding operations and maintenance. With respect to DevOps [9], it includes continuous software architecting activities. We consider software-as-a-product, in view of which development, operations, and maintenance continue until retirement of the software-as-a-product.

3. Continuity refers to the **continuously changing state** of the architecture, software, operations, processes, and management because of progressive insights of the stakeholders, bug fixes, and continuously changing context: even keeping the software unchanged in a changing context results in additional requirements or other unforeseen factors.
4. **Information is distributed across tools** used in the software development ecosystem. There rarely is a single repository in which information is stored for a system. Furthermore, documentation includes a big variety of structured and unstructured data like texts, as in git commit messages, photos from whiteboard sketches, diagrams with Unified Modeling Language (UML) models or Entity-Relationship-Diagrams (ERDs), and executable scripts as used in Test Driven Development (TDD).
5. The **amount of documentation and quality of documentation is low**. In Lean, documentation is considered to be waste when it does not directly contribute to the end product [15]. In Agile [5] software development, working software is valued over comprehensive documentation, and face-to-face communication is considered much more effective than written documentation. In DevOps, CI/CD scripts are considered to be documentation with infrastructure-as-code [9].

Documentation is the central aspect we focus on here because on the one hand, we observe a fundamental mismatch between the amount and quality of documentation that is produced and consumed [20, 21]. On the other hand, having no documentation at all is not a viable possibility. Developers and other stakeholders require a minimal amount and quality of documentation because of the continuity of development until retirement of the software-as-a-product. Our research project contributes insights on what is just enough documentation for developers and other stakeholders throughout the beginning, continuation and finalization of the software-as-a-product lifecycle. Second, it contributes insights on how structured and unstructured information that is scattered across the software development ecosystem can be transformed into comprehensible documentation.

In this paper we present types of information (the *what*) in Section 2 and reasons for documenting (the *why*) in Section 3. Previously, after a few preliminary studies [20–22], a literature review was conducted to search for publications on documentation in CSD. Currently, we are in the process of conducting a case study on necessary and sufficient conditions for gaining insight with respect to control of effective communication through documentation in CSD, and on how information that is scattered throughout the tools in the software development ecosystem can be transformed into coherent and comprehensible documentation. After the completion of this case study, the objective is to define a theoretical

framework on documentation in CSD. The project will be concluded with an empirical validation of the theoretical framework.

The contribution of the project to the scientific community is the following. Although there are numerous studies on documentation on software development, not much research has been done in Lean, Agile, and DevOps software development methods (which have CSD characteristics). Our project at large primarily aims to contribute a documentation framework that is streamlined for CSD, creating the necessary and sufficient conditions for effective communication with just enough documentation in CSD.

The remainder of this paper is organized as follows. In Section 2, the types of information (the *what*) are described. Next, Section 3 provides a description of reasons for documentation (the *why*). In Section 4 the study design is explained, including research questions. The conclusion and ideas for future research are presented in Section 5.

2 Types of Information

From a previous literature review on documentation practices, documentation challenges, and tooling in CSD, we learned that the following relevant typology of information can be distilled from the literature: We adapted the Constant

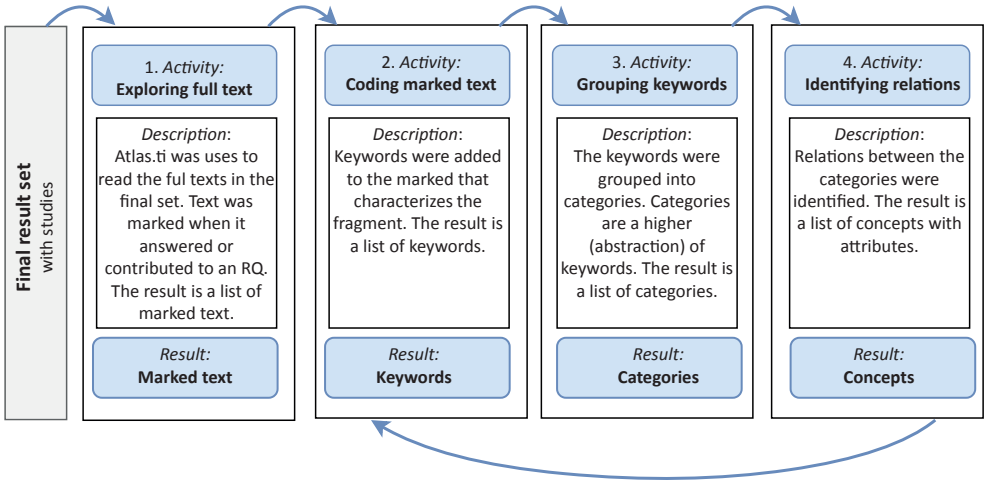

Fig. 1. Constant Comparative Method, adapted from Miles et al. [13]

Comparative Method from Miles et al. for the qualitative analysis of the literature review [13], as depicted in Figure 1. The list of information types was compiled by first reading and marking all the text fragments in the studies. The second step was to add keywords to the marked text. The third step was to group

the keywords into categories with similar meaning, for example *when* the documentation was created, or the intended *use* of the documentation. The last step was to add relations between the categories to create concepts of information types.

The types of information we identified, based on git documentation [1] and on studies from Beck et al. [5], Bass [3], ISO/IEC documents [8, 19], Christensen et al. [7], Bosch et al. [6], Poppendieck et al. [15], include:

1. Stakeholder concerns, risks, constraints, and context.
2. Requirements define *what* a system should do.
3. Specifications define *how* a system should work.
4. The source code itself.
5. Annotations in the source code.
6. Commit messages accompanied with source control management systems.
7. Playbooks, which are executable scripts for CI/CD pipelines.
8. Metrics list indicators of measurable features of processes, tasks, and systems.
9. Values and beliefs, principles, and practices in which values and beliefs can be observed in behavior. We define principles as explicit rationalizations of values and beliefs. Practices can be observed as recurrent actions.
10. Processes, procedures, and tools.
11. Knowledge, competences, and attitude.

Furthermore, the aspect of the **medium** of the information should be taken into account. The medium concerns the way information is communicated [17, 18]. Traditionally in software development, *written documentation* was communicated on paper and later replaced by digital documents. In CSD, *verbal communication* is prominently practiced, for instance in the daily stand-ups in Scrum. This type of information is not written down in any way. At the end of team meetings, knowledge literally walks out the door. Finally, since information is scattered throughout (tools in) the development ecosystem, we should consider *software development tools* also as a medium.

In this section, types of documentation (the *what*) were described. The items in this list are the result of a literature review. Both the types of information and the aspect of the medium were discussed. In the following section, we describe the purposes for documentation in CSD.

3 Reasons for Documenting

In the literature review, we found that information about a system is scattered throughout *tools* comprising a software development ecosystem [10]. We found purposes for documentation in studies from Ambler [2], Laporte et al. [12], Visković et al. [23] and Praks et al. [16]. The purposes are:

1. Providing project stakeholders with required documentation.
2. Definition of a contract model, such as a codified description of interfaces.

3. Support of communication inside the team, especially larger teams.
4. Support of communication with external, geographically distributed, groups.
5. Support of organizational memory.
6. Auditing purposes.
7. Believing something to be true.
8. Reducing defects in working software.
9. Facilitating training.

In this section we listed reasons (the *why*) for documentation as described in primary studies. The types of information presented in the previous section combined with the reasons for documentation presented in this section form the basis for the research questions proposed in the next Section.

4 Study Design

4.1 Objectives and Research Questions

The *objective* of this study is defined in the main research question:

> *What are the necessary and sufficient conditions to gain insight in, and have control of, effective communication with just enough documentation in Continuous Software Development (CSD)?*

"Necessary conditions" refers to the minimal requirements for an event to occur. "Sufficient conditions" make the event to actually occur. A necessary condition alone is not sufficient. A simple example can make this clear. The necessary conditions for fire are 'air', 'fuel' and 'heat'. However, these necessary conditions become sufficient for fire only when air, fuel and heat are in a specific configuration. A simple example for a sufficient condition without being necessary is "you traveled to Amsterdam by plane", the plane being sufficient, but not necessary. *Insight* refers to knowledge and facts that someone is aware of. *Control* refers to the ability to change the course of events in certain directions. Insight is a necessary condition for control. *Effective communication* refers to results that could not have been achieved without sufficient information. *Documentation in CSD* refers to the domain (scope) for this research. Making the necessary and sufficient explicit helps in understanding which elements are required for concepts like 'insight', 'control', 'effective communication', and 'documentation in CSD' including the relations between these concepts. Additionally, the conditions make explicitly clear how, why and when the concepts occur.

RQ1 What are *reasons* for documenting in CSD?

1. How do types of documentation (the *what*) and reasons for documentation (the *why*) match?
   In Section 2 a list of *what* is documented was presented. When we extend these types of documentation with attributes, one of the attributes could be the reason for documenting. In Section 3, we presented reasons

for documentation. However, the type of documentation (the *what*) and reason for documenting (the *why*) have not yet been connected. In our research project, we want to investigate what reason(s), found in related literature, can be attributed to what type of documentation.

2. What are trade-offs for different *types of industry* and *why they weigh alternatives differently* for documentation?
   Bass et al. describe two trade-offs: fast time to-market and regulatory business [4]. Examples of industries for fast time-to-market include social media websites and web shops. Examples for regulatory industry include the Food and Drug Administration (FDA), and banking and tax administration. This can easily be extended with other trade-offs such as fast time-to-market and accountability, or agility and maturity. We are looking specifically to types of industry that have high demands for both alternatives.
3. Validation of the reasons that are found in the literature as presented in Section 3 on reasons for documentation through the questioning of experts in industry.

RQ2 What are necessary and sufficient conditions to organize information scattered throughout a CSD ecosystem into comprehensible documentation?

1. Which tools are used in the software development ecosystem?
   Kersten et al. present a wide range of tools and tool categories [10]. With this research question we investigate the tools and tool categories in the software development ecosystem. We also investigate how these tools are used (for communication) in CSD.
2. What is the variety of the information that is stored in tools?
   Variety refers to structured and unstructured information that is stored in the tools. The information is of many kinds like texts such as in git commit messages, photos from whiteboard sketches, diagrams with UML or ERDs and executable scripts such as TDD.
3. Which information is stored with what tool?
   In Section 2 on types of information, a list is presented of *what* is documented. With this research question, we investigate which tools or tool categories are used to store what types of information.
4. How can this scattered information be organized into comprehensible documentation?
   Since the information is scattered throughout the tools that comprise the software development ecosystem, the question is how this scattered information leads, or can lead, to comprehensible documentation.

4.2 Research Project

Our research project at large is built up from a number of studies. Below, we present the studies including objectives, and methods.

1. The first study is a **literature review**. The objective is to present an overview of literature related for documentation practices and documen-

tation challenges in CSD, and which tools are used in the software development ecosystem of CSD. The period starts with the introduction of the Agile Manifesto in 2001 and ends in 2019. The method for the literature review is a Systematic Mapping Study (SMS) with elements from a Systematic Literature Review (SLR). For the SMS methodology, we use Petersen et al. ([14]), and for the SLR, we use Kitchenham et al. ([11]). We want to present a broad overview of literature and to categorize this into dimensions. We address broader research questions regarding the trends in documentation practices, challenges, and tools in CSD. The topic area is specific with respect to documentation in CSD. The focus is on primary literature, and we exclude secondary studies such as other literature reviews. Furthermore, the topics of Lean, Agile and DevOps is very practitioner-oriented and at least part of the studies are empirical, and therefore it is complicated to evaluate the quality of primary studies. The result is an overview of literature in a bubble chart, based on Wieringa et al., with dimensions for contribution facets, context facets (related to research questions), and research facets [24].

2. The objective of the **case study** is to present an overview of observations in the industry community for documentation practices and documentation challenges in CSD, and which tools are used in the software development ecosystem of CSD. The *cases* are, according to Yin et al., contemporary phenomena in their real-life context [26]. The cases in this study concern examples from industrial sectors that have high demands on agility and reliability, e.g. fast time-to-market, because of competition or continuously changing regulations such as web shops and tax administration. Some industries, however, have a high demand for accountability and transparency [4].
3. Based on the literature and case studies, we construct a **theoretical framework** that is streamlined for documentation in CSD. With this framework we provide the context, i.c. documentation in CSD, develop hypothesis to explore the research, provide a framework with categories for observations, define a concept, provide a research design, provide a model for interpretations, and give generalizations. The framework serves as a guide to systematically identifying logical en causal relations among concepts. We will use Wieringa's Design Science [25] framework to construct the theoretical framework.
4. The research project ends with a **validation of the conceptual framework** in the industry. We will use both quantitative and qualitative methods to evaluate the framework.

5 Conclusions and Future Research

We refer to CSD as a set of common characteristics from Lean, Agile, and DevOps. CSD includes activities from the complete lifecycle of software-as-a-product: continuous architecting, operations, and maintenance. In CSD, there is less documentation and in general this documentation is of low quality. We consider a mismatch between the production and consumption of documentation on

the one hand, while on the other hand there is a need for a minimal amount of documentation for developers and other stakeholders to start, continue, and end the software-as-a-product lifecycle. In this paper we presented types of information (the *what*) and reasons for documenting (the *why*), together with research questions for further research. In future research we will set out to investigate how types of information and reasons for documenting match. We also want to find out to what extent the requirements differ for the various types of industry, such as fast time-to-market and heavily regulated organizations.

We observed that information in CSD is scattered throughout the software development ecosystem. We are therefore interested in how this scattered information can be transformed into comprehensible documentation for developers and other stakeholders.

Future research includes the construction of a theoretical documentation framework that is streamlined for CSD. The final objective is to validate the documentation framework in the industry. This framework provides the necessary and sufficient conditions for effective communication with just enough documentation in CSD.

Acknowlegements I would like to thank the supervisors of this Ph.D. project, which are: Prof. Dr. Sjaak Brinkkemper, Dr. Stijn Hoppenbrouwers, and Dr. Sietse Overbeek for their reviews and support of this paper.

References

1. Gitcommitmessages - openstack. https://wiki.openstack.org/wiki/GitCommitMessages
2. Ambler, S.W.: Agile/lean documentation: Strategies for agile software development. http://agilemodeling.com/essays/agileDocumentation.htm
3. Bass, L., Clements, P., Kazman, R.: Software architecture in practice. Addison-Wesley Professional (2003)
4. Bass, L., Weber, I., Zhu, L.: DevOps: A software architect's perspective. Addison-Wesley Professional, 1st edn. (2015)
5. Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D.: Manifesto for Agile Software Development Twelve Principles of Agile Software. Tech. rep. (2001), http://www.agilemanifesto.org
6. Bosch, J.: Continuous software engineering: An introduction, vol. 9783319112. Springer (2014). https://doi.org/10.1007/978-3-319-11283-1-1, http://link.springer.com/chapter/10.1007/978-3-319-11283-1_1
7. Christensen, H.B., Hansen, K.M.: Towards architectural information in implementation. In: Proceeding of the 33rd international conference on Software engineering - ICSE '11. p. 928 (2011). https://doi.org/10.1145/1985793.1985948
8. Engineering, S., Committee, S.: IEEE Recommended Practice for Software Requirements Specifications, vol. 1998 (1998), http://www.math.uaa.alaska.edu/ afkjm/cs401/IEEE830.pdf
9. Hüttermann, M.: Infrastructure as code. In: DevOps for Developers, pp. 135–156. Springer (2012)

10. Kersten, M.: A cambrian explosion of DevOps tools. IEEE Software **35**(2), 14–17 (2018). https://doi.org/10.1109/MS.2018.1661330
11. Kitchenham, B., Charters, S.: Guidelines for performing Systematic Literature Reviews in Software Engineering. Engineering **2**(4ve), 1051 (2007)
12. Laporte, C.Y., April, A.: Policies, Processes, and Procedures (2018)
13. Miles, M.B., Huberman, A.M.: Qualitative data analysis: An expanded sourcebook, 2nd ed. Sage Publications, Inc, Thousand Oaks, CA, US (1994)
14. Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M.: Systematic Mapping Studies in Software Engineering. 12Th International Conference on Evaluation and Assessment in Software Engineering **17**, 10 (2008)
15. Poppendieck, M., Poppendieck, T.: Lean software development: an agile toolkit. Computer **36**(8), 89–89 (2003). https://doi.org/10.1109/MC.2003.1220585
16. Praks, J., Tikka, T., Kestilä, A., Hieta, M.: Online documentation approach for assisted system engineering and assessment in student projects. In: 2015 IEEE Global Engineering Education Conference (EDUCON). pp. 608–611. IEEE (2015)
17. Rodríguez-Elias, O.M., Martínez-García, A.I., Vizcaíno, A., Favela, J., Piattini, M.: A framework to analyze information systems as knowledge flow facilitators. Information and Software Technology **50**(6), 481–498 (2008)
18. Shannon, C.E., Weaver, W.: The mathematical theory of communication, 117 pp. Urbana: University of Illinois Press (1949)
19. Standards Committee: IEEE Std 1016-2009 (Revision of IEEE Std 1016-1998), IEEE Standard for Information Technology—Systems Design—Software Design Descriptions, vol. 2009 (2009). https://doi.org/10.1109/IEEESTD.2009.5167255
20. Theunissen, T., van Heesch, U.: 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, pp. 265–273. Springer, Cham (nov 2016)
21. Theunissen, T., Van Heesch, U.: Specification in Continuous Software Development. In: Proceedings of the 22nd European Conference on Pattern Languages of Programs. p. 5. EuroPLoP '17, ACM, ACM, New York, NY, USA (2017)
22. Van Heesch, U., Theunissen, T., Zimmermann, O., Zdun, U.: Software Specification and Documentation in Continuous Software Development. Proceedings of the 22nd European Conference on Pattern Languages of Programs - EuroPLoP '17 pp. 1–13 (2017). https://doi.org/10.1145/3147704.3147742, http://dl.acm.org/citation.cfm?doid=3147704.3147742
23. Visković, D., Varga, M., Ćurko, K.: Bad practices in complex IT projects. Proceedings of the International Conference on Information Technology Interfaces, ITI pp. 301–306 (2008). https://doi.org/10.1109/ITI.2008.4588425
24. Wieringa, R., Maiden, N., Mead, N., Rolland, C.: Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. Requirements Engineering **11**(1), 102–107 (2006). https://doi.org/10.1007/s00766-005-0021-6
25. Wieringa, R.J.: Design science methodology for information systems and software engineering. Springer (2014)
26. Yin, R.: Case Study Research: Design and Methods, 3rd Edition (Applied Social Research Methods, Vol. 5). Sage Publications, Inc., third edit edn. (2002). https://doi.org/10.1086/421629