Necessary and Neglected? An Empirical Study of Internal Documentation in Agile Software Development Teams

Christoph Johann Stettina
stettina@liacs.nl

Werner Heijstek
heijstek@liacs.nl

Leiden Institute of Advanced Computer Science, Leiden University
Niels Bohrweg 1, 2333 CA Leiden, the Netherlands
http://www.liacs.nl/home/{stettina,heijstek}

ABSTRACT

When compared to traditional development methods, agile development practices are associated with more direct communication and less documentation. However, few empirical studies exist that investigate the role of documentation in agile development teams. We thus employed a questionnaire to measure the perceptions of a group of agile practitioners with regard to the documentation in their projects. We obtained responses from 79 agile software development professionals and 8 teams in 13 different countries. Our findings include that over half of developers in our data set find documentation important or even very important but that too little documentation is available in their projects. Agile practitioners do not seem to agree with the agile principle that "The most efficient and effective method of conveying information to and within a development team is face-to-face conversation." We were able to validate this result for a set of dissimilar agile teams in various domains.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specifications—*Methodologies*; D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement—*Documentation*; K.6.1 [**Management of computing and Information Systems**]: Project and People Management—*Management techniques*

General Terms

Documentation, Management, Human Factors, Measurement

Keywords

Documentation, Agile software development, Empirical study

SIGDOC'11, October 3–5, 2011, Pisa, Italy.

1. INTRODUCTION

Scrum [18] is an often applied software development methodology [1, 6]. In the spirit of lean manufacturing, Scrum emphasizes a focus on working software and direct communication rather than written documentation [1, 3, 6, 17]. Based upon the mental attitude of value-oriented utilization of resources, lean manufacturing considers the expenditure of resources for any goal other than the creation of value for the end customer as wasteful. In agile development, these attitudes can be found in the agile manifesto [9] which considers heavyweight processes and comprehensive internal documentation of no direct use to the end customer. Internal documentation in software engineering may include requirement specifications, design specifications and technical documentation of program code. While such might be of no direct use to the end-user of consumer products, technical documentation in software projects such as embedded comments within the source code, descriptive and/or explanatory notes of APIs, interfaces and algorithms are thought to be beneficial to project initiators as well as to the engineers who are to maintain or expand the software in the future. Originating from practice [18] and reflected in theory [14], the advantages and limitations of Scrum as an agile development method and agile teamwork in particular [21] have been widely discussed in literature (e.g. [1, 6]). However, although companies using agile methods have been found to be more customer-centric and flexible than document-driven ones [20], little has been reported on use of documentation within agile teams. While the strictly defined Scrum method with its subsequent phases aids in reducing uncertainty, there is surprisingly little to no anchorage of documentation within the process. In this contribution we therefore aim to improve the understanding of internal documentation in agile software development teams in general, and how agile practitioners perceive this documentation in particular.

2. RELATED WORK

The term Scrum depicts cross-functional team characteristics and overlapping phases similarly to those in rugby. It is a lightweight, agile development method, aiming to strip the process of software development to its bare minimum and putting emphasis on direct communication and self-managing teams [6, 21]. As an adaptive rather than predictive model [17] it does not depend on heavy documentation written upfront. Relying on constant collaboration

among the team members and stakeholders, however, literature points at the dangers of general loss of undocumented knowledge (e.g. [1, 17]) when members leave the team or when the project is delivered. In distributed software development projects, which are increasingly common, software documentation is thought to play a more central role due to the absence of face-to-face communication. The common "transfer by development stage" [13] approach to global software development (GSD) where design and implementation activities take place at different geographical locations, complicates matters further. In this situation, offshore developers are often not able to directly contact a member of the architecture team due to geographical separation. Synchronous communication is often difficult due to time zone differences. In such a scenario documentation plays a more central role in intra-project knowledge sharing processes. In addition, complete and detailed documentation is essential to software maintenance as this typically involves different engineers from the ones who developed the system. Studies reporting on the use of software documentation during software development are few and deliver mixed results. In studies researching developers' preferences regarding documentation, Lethbridge et al [8,10] find a preference for simple and powerful documentation and conclude that documentation is an important tool for communication, even if it is not up to date. In their literature review on empirical studies addressing the scientific level of evidence behind agile software development methodologies, Dybå and Dingsøyr [6] address "documentation" in just one of the identified contributions [20]. In one of the few contributions to understanding of documentation in agile projects, Clear [3] points at the behavior of students and observes documentation being seen as something external. Instead of being produced in-line with the system as natural part of the development process, documentation was often hurriedly pieced together at the end of a project.

3. OBJECTIVES

There is a commonly assumed antipathy of agile practitioners towards internal documentation. This assumption stems not in the last place from the agile manifesto, which states that *"[Software documentation is] important, but we need to understand that customers don't care about documents, UML diagrams or legacy integration."* [9]. With the increasing number of integration projects and changing team setups, however, the transfer of development knowledge becomes increasingly important. In this paper we are interested in understanding the role of documentation in agile development projects, and how the amount and importance of documentation is perceived from the perspective of an agile team. We therefore pose the two following research questions:

1. *How do team members in agile software development projects document their work?*
2. *How do they perceive the amount and importance of their internal documentation?*

4. METHOD

In order to be able to attain a level of external validity, data from a wide range of agile practitioners is needed. In order to uniformly obtain a sizable sample and to promote objective answers we developed an anonymous questionnaire.

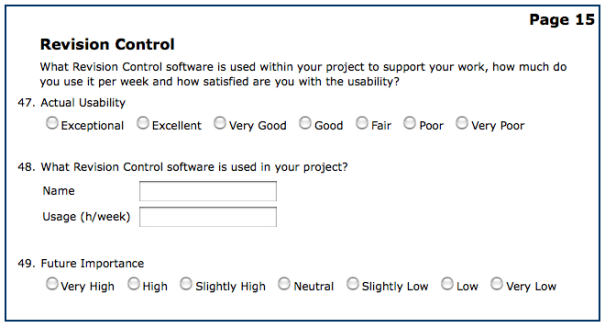


Figure 1: Online Questionnaire: Technology

Surveys are easy to deploy and when self-administered through a computer, their use minimizes the effect of socially desirable responding [15] due to the impersonality of the computer. To reduce bias we encouraged the respondents to provide their honest opinions by emphasizing the anonymous treatment of data. While we provided personalized links for each team member to ensure the consistency of input, no personal details were stored or used within the examination. An example of the format of the questionnaire is displayed in Fig. 1. Additionally, Miles and Huberman [12] propose linking quantitative and qualitative data to provide a deeper picture for the research. We thus asked each ScrumMaster (i.e. (agile) project leader) of an interested Scrum team a set of open-ended interview questions regarding the project environment at the end of the data collection. These qualitative questions (compare Tab.1) aimed to learn about the background of the project and details not covered by the quantitatively collected data.

4.1 Questionnaire Design

In line with our objective, we want to understand how agile teams produce their documentation and how satisfied they are with it. To reach this two-fold goal we divided the questionnaire into two parts. Part one was geared towards collecting the perceptions of team members regarding their documentation. Part two was designed to inquire about the software tools applied to manage that documentation. The questionnaire was part of a bigger study on agile teams and was developed with software developers including input from practitioners and fellow researchers working in the field of agile software development. To minimize possible flaws in the questionnaire design and to ensure appropriate scales applied, we reviewed the questions with colleagues from the faculty of psychology at the NTNU Trondheim which was the original site of this study.

4.1.1 Perceptions Regarding Documentation

The data collected from the teams was coded according to a five-point Likert scale. The questions have been administered in the following order:

1. *How much time do you spend on writing documentation daily?*
2. *How do you feel about documentation at work?*
3. *How effective do you consider finding internal documentation?*
4. *How important do you consider documentation for your project?*

Next to the questions regarding documentation, the following question was included to probe the usage of physical artifacts within the agile environment:

- *How important do you consider physical artifacts like story cards or "the wall" for your project?*

4.1.2 Supportive Software

In addition to developers' perceptions of documentation we also needed to inquire about the software tools supporting the development work. To cover a wide range of documentation tools within the survey we agreed upon including 6 categories of tools, namely: (1) *issue tracking*, (2) *revision control*, (3) *electronic discussion*, (4) *Scrum support*, (5) *document management* and (6) *calendar & scheduling*. As we needed to measure the usage of software packages applied, we first asked the participants if a certain tool category is being used within their project. We then asked for the perceived usability from the perspective of the developer. To optimally collect an objective measure of software usability, we applied a method adopted from the European Software Institute's (ESI) 1995 Software Excellence Survey [5] as discussed by Dybå and Moe [7]. We assessed the usability by placing two opposing subjective rating scales accompanying each question: the *actual usability* and the *believed future importance* as depicted in Fig. 1. The gap between believed importance and usability can be regarded as a measure for disaffection with a given solution. This gap is visualized in Fig. 7b. The perception of each team member was gathered for the respective categories as depicted in Fig. 1. To prevent inconsistency among the rating items we used two Likert scales consisting of 7 items: *Very High* = 7, *High* = 6, *Slightly High* = 5, *Neutral* = 4, *Slightly Low* = 3, *Low* = 2 and *Very Low* = 1.

4.2 Team Agreement

Variance (σ^2) is a measure of how far each value in a set of responses is from the mean. We calculated the variance of the collected values on a team basis to obtain insight in inter and intra-team agreement. A lower variance corresponds with a greater level of agreement within a team. The maximum variance in a team is the variance of the maximum and minimum values that can be given in response to an answer. The minimum variance is 0, denoting complete agreement. On a Likert scale ranging from -2 – 2, the maximum variance is

$$\sigma^2 = \frac{\sum(X-\mu)^2}{N} = \frac{\sum X^2}{N} - \mu^2$$

$$\text{MAX}(\sigma^2\{1,5\}) = \frac{1^2+5^2}{2} - \left(\frac{1+5}{2}\right)^2 = 4.$$

4.3 Data Collection

To collect data, the questionnaire has been presented as a set of online survey questions to a group of international project teams, practitioners and experts applying the Scrum methodology. The participants had to be actively involved in a Scrum development team. Only completely filled questionnaire were considered for analysis. To look for international Scrum teams interested in the study Scrum and agile oriented groups within business related networks were targeted. After identifying related individuals from different online community platforms, user groups as well as agile practitioners originating from direct and indirect contacts, invitations to participate were sent to 150 potential participants. Those who expressed their interest in participation, received an anonymized link allowing identification of teams within the online survey system. In addition, each ScrumMaster then received the set of open-ended questions regarding the project environment (Tab. 1).

5. RESULTS

After data collection all obtained answers were collected into a *global sample*. In addition to this sample, where at least four agile practitioners of a single team provided a valid survey response, data was also divided into *team samples*. These team samples provide a second perspective on the collected data set by enabling analysis on a team level. The total number of valid data responses comprises 79 software engineers from eight teams located in 13 different countries. Most of the engineers are software developers (47%) and ScrumMasters (18%). Other groups, however, emerged within the data collection phase. These groups are: *Product Owner* (8%), *Quality Assurance* (6%), *Agile Coach* (6%), *Consultant* (9%), *Interaction Designer* (1%) and *CTO* (5%). Their data has been taken into evaluation as long as the individuals were committed to a *"Pig role"* within the Scrum project, thus, being directly involved in the production in a formal and frequent way while referring to internal documentation. The gross amount of relevant working experience among the participants is situated around a work record of 1–5 (38%) and 6–10 (29%) years. Eight teams were analysed in detail. The teams consisted of at least four members and at least two-thirds of all teams have answered the survey. We therefore feel that a consistent representation of a group image was attained for all eight teams. We first present our results from the global sample and then switch to the team perspective.

5.1 Global Sample

5.1.1 Documentation

In Fig. 2, an overview can be found of the time the participants spend on writing documentation in their project daily. We find that the majority of participants spend less than 15 minutes on writing documentation daily. In line with this finding, we observe that almost half of the participating software experts perceive the existing documentation in their project as *too little* (Fig. 3). Fig. 4 presents an overview of developers' perceived effectiveness of finding existing documentation in their project. The diagram depicts a balanced distribution. This balanced distribution implies that a substantial group of participants does not find availability in line with agile practices in which documentation must support the development process. Fig. 5 depicts an overview of participants' responses with regards to their perception of the importance of documentation. Here, we observe that majority of the participants define documentation as being *important* or *very important*. This observation is remarkable as we would have expected that agile practitioners would rate documentation as not so important. According to Sharp et al., while relying heavily on direct, face-to-face communication, agile teams use simple physical artifacts to support interaction [19]. This is confirmed by our findings as the majority of participants think physical artifacts are important in their project (Fig. 6).

Table 1: ScrumMaster Interview Questions

• Please describe your project shortly. What is the product? What is its size (e.g. man months, lines of code)?
• Please describe your team shortly.
• What is the team's spatial distribution? Is it physically separated?
• What changes did you had to adapt to while switching to Scrum?
• Which aspects of communication do you consider the most salient for the project?
• Does the communication differ to the projects you had before Scrum? How?
• How do you report to your manager? Do you have separate reports to Project Owner and to "Chickens"?
• How do you document the work you do? What is the structure of the project documentation?
• How do you prioritize documentation tasks within the project?
• Are you happy with the current process to handle communication in general (Documentation, Requirements, Meetings etc.)?
• Are you using additional tools to those shared by the rest of the team to improve your productivity?

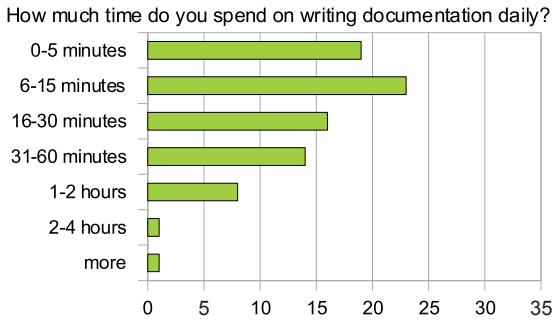


Figure 2: Time spent on writing documentation

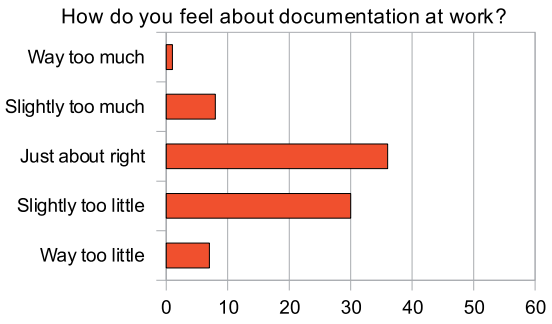


Figure 3: Perceived amount of documentation

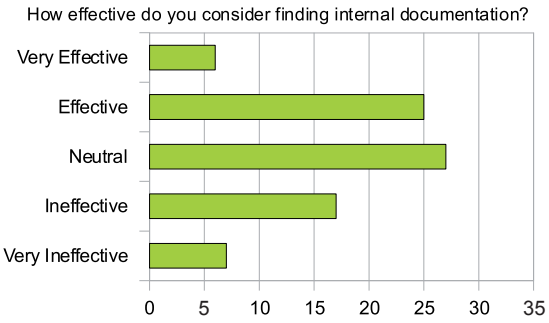


Figure 4: Perceived effectiveness in finding documentation

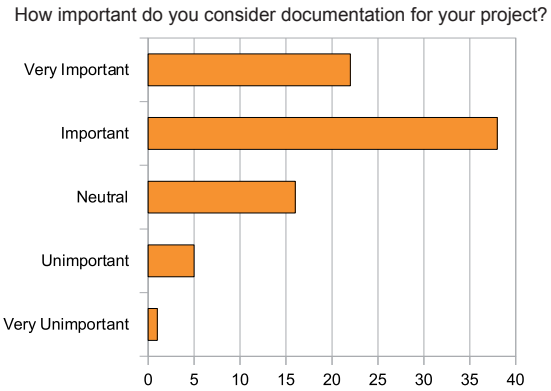
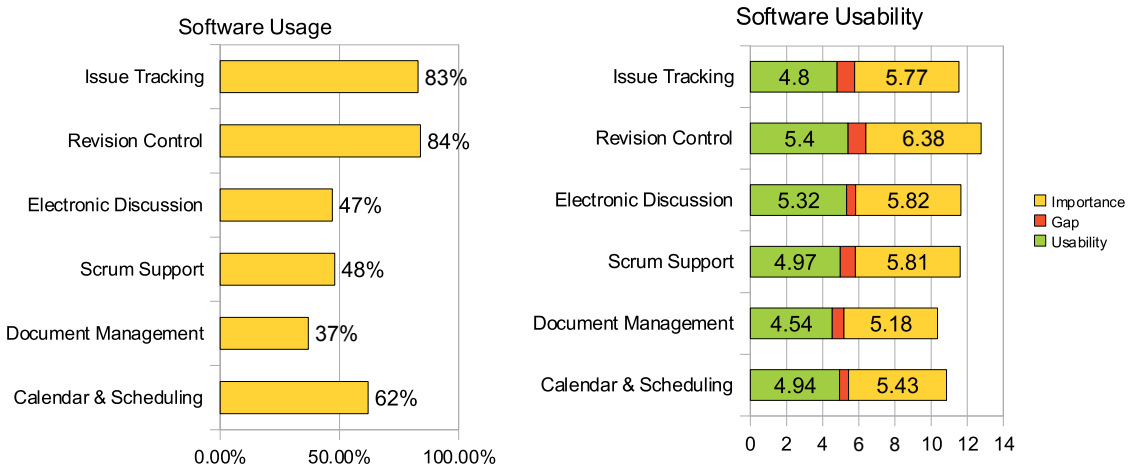


Figure 5: Perceived importance of documentation

5.1.2 *Supportive Software*

In Fig. 7, we present an overview of the software categories applied by the agile practitioners in their projects. We observe that issue tracking and revision control are used by most participants. The common use of this technology might well be explained by the notion that, next to their employees, source code and knowledge are among the most valuable assets of a software company. Calendar and scheduling tools are used by 62% of the participants. Software supporting electronic discussions and Scrum are similarly distributed with 47% and 48%. Document management is the least applied tool category. Interestingly, document management is reported to be used least by the respondents. In the survey, *document management* was explained to entail the storage of requirements, design papers, used interfaces, created functions and sub-functions.

While looking at the deviation of usability and importance we observe that revision control, issue tracking and Scrum support have the biggest gap. This means that the participants perceive the highest disaffection for these three categories. The use of Scrum support tools seems surprising. Although one might think that the need for applications supporting a rather direct and physical development method would be rare, there indeed seems to be a demand for supplementary software tools. The perceived usability of solutions for electronic discussion shows the smallest gap among all categories. The most applied software packages



(a) Usage (b) Usability

Figure 7: Software usage, perceived usability and believed importance

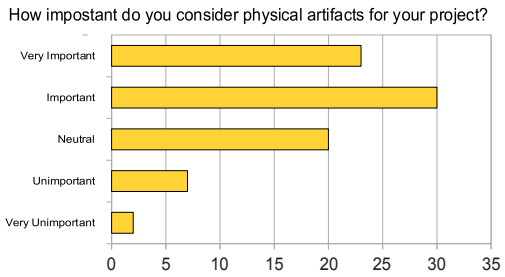


Figure 6: Perceived importance of physical artifacts

for the respective categories were: (1) issue tracking: *JIRA*, (2) revision control: *Subversion*, (3) electronic discussion: *Skype*, (4) Scrum support: *JIRA and Microsoft Team Foundation Server (Scrum plug-in)*, (5) document management: *Wikis and Google Docs* and (6) calendar & scheduling: *Microsoft Outlook*.

5.1.3 Correlation Analysis

Due to non-normality, we resort to non-parametric tests. Using a Mann-Whitney test on the global sample of 79 participants, we find, not surprisingly, that those who identify as software engineers are significantly younger and have significantly less experience than people in managerial roles such a project leader, director or chief technology officer. The latter group notes that they spend significantly more time on conversations at work and writing documentation. This contrasts with the claim by software engineers that they find that too little documentation is available. However, when we compare software engineers to those in senior technical roles such as architects, we find that this latter group also spends significantly more time writing documentation. So in the teams in our data set, top down dissemination of documentation seems prevalent from the perspective of who writes the documentation.

Interestingly, we find no difference between the answers given by the ScrumMasters and those that identify themselves as 'normal' project leaders. When compared to 'traditional' technical leads, ScrumMasters note that they write significantly less documentation but that they spend significantly more time in conversations. Again, this does not seem to correspond with agile developer's requirements regarding the availability of documentation in their projects.

We find that the more experienced agile team members are, the more efficient they find documentation in their project ($\tau = -0.294$ at $p = 0.018$). A possible explanation could be that developers only learn to understand the importance of documentation as they progress in their careers.

We find that those that spend much time creating documentation are significantly more positive about the amount of documentation available in their jobs ($\tau = 0.555$ at $p = 0$). This could simply mean that these people justify their labor or that this group is simply more aware of the documentation that is available. It might, however, also indicate that this group is given the time to write documentation and are therefore happier. We are unsure whether teams in which these engineers work were more efficient as a result of this, though.

5.2 Team Sample

Data sets consisting of sufficient interrelated team members have been aggregated into team samples and are presented in Tab. 2. Team T1 (UK) provided a back-end software for a major Massive Multiplayer Online (MMO) game publisher. Team T2 (US) developed a collaborative solution to support building design and construction-related tasks for a major American software company. Team T3 (UK) developed software for digital media and mobile platforms concentrating on software development, interaction and visual design. Team T4 (Norway) was employed by a company providing smart card based public key solutions for security transactions, consisting of developers from 2 separate locations running several parallel projects. T5 (The Netherlands) was in charge of corporate websites and web shops. Team T6 (Sweden) developed and maintained a Swedish news guide and community website. Team T7 (India) worked on an e-commerce solution. Team T8 (New Zealand) con-

sisted of a business analyst, a quality assurance specialist and two developers working for a state insurance agency.

Between the teams, the team members agree most on the fact that too little documentation is available. This consistency between dissimilar teams strengthens our earlier finding that agile practitioners find that too little documentation is available.

5.2.1 Documentation

As presented in Tab. 2, the perceptions on the amount of documentation, effectiveness in finding internal documentation and importance of artifacts are accumulated from the data. According to the respective Likert scale, the values are distributed on a scale from "−2" to "+2". A "0" thus corresponds to *"just about right"*, a "+1" to *"slightly too much"* and a "+2" to *"way too much"* while a "−1" and a "−2" correspond to *"slightly too little"* and *"way too little"*, respectively. The value of "1" therefore means that the team perceives documentation as *"slightly too much"*, while the value of "−2" would represent a team perception of documentation as *"way too little"*. Consistently with the global sample Tab. 2 reveals that more than half of the researched teams reported perceived a deficit in the amount of documentation. Note that none of the teams perceive their documentation as *"too much"*.

5.2.2 Supportive Software

We observe that Wiki solutions are prominent within the researched teams: five of the eight groups specifying to use a document management solution name a Wiki system as the tool used. The two Wiki solutions in brackets, mentioned in row "documentation tool" in Tab. 2, mark the two cases where the team ScrumMaster mentioned the particular application used by the team in the open questions. Team members of those same teams, however, did not specify the use of these specific tools. This might imply that these Wiki solutions were introduced but not adapted in practice by the team: *"[W]e have a wiki that we are supposed to use,"* comments the ScrumMaster of Team T6.

After the Wiki, the most commonly used artifact seems to be the "user stories". A user story is the agile equivalent of a use case. The distinction between the Wiki and user stories seems to be fluid, as one ScrumMaster (Team 1) notes, *"Work is documented on the company wiki where required. If it is a small addition then it will be part of a task, but if it is a larger addition then it may have it's own user story."*. Software tracker JIRA[1] was mentioned several times, *"We use JIRA with the Greenhopper plug-in. This provides virtual equivalents to for a wall, User Story cards, burndown,"* says a ScrumMaster (Team 2). Also Team seven was positive about JIRA, *"Jira is our "source of truth". Confluence [A Wiki] is the repository for collaborative documentation. We also required sensible comments in commits. We ran Clover for test coverage, Crucible for collaborative code reviews, CruiseControl for Continuous Integration. Most of the project documentation was divided between the product backlog (in Jira) and shared documents in Confluence,"* noted its ScrumMaster.

Team three makes use of a combination of Google Docs and a whiteboard, *"We have a sprint backlog for each team as a Google docs, editable by the whole team. Our release plan is on a whiteboard, and we have per-client product backlogs for individual products - again, in Google Spreadsheets. [...] Beyond that the main documentation we produce is user experience design docs. I think we're a bit heavyweight here right now but we're working for some clients who demand this stuff for their internal sign-off procedures,"* according to the ScrumMaster.

Team five works with a combination of an advanced change and configuration management system and post-it notes: *"We have a Story Board with progress using post-its next to the team. I add the burndown-chart there every morning. The team keeps the StoryBoard up to date. For estimating we use Planning Poker (with cards). [...] As the ScrumMaster I don't have a lot of documentation to do. The Product Owner will provide the documentation for the team, creates User Stories for them in Microsoft Team Foundation Server (TFS) and decides the prioritization,"* says the ScrumMaster.

[1]http://www.atlassian.com/software/jira/

6. DISCUSSION

In this section we discuss our findings in line with our research questions. There seems a natural antipathy of software engineers towards activities not directly linked to the creative process of writing code, many engineers indeed prefer writing code instead of drafting their ideas in text [3]. Agile software development methods concentrate on direct communication emphasizing that comprehensive internal documentation is of no direct use to the end customer [9]. In line with existing concerns [1, 17], however, the findings of this study point out that direct communication alone seems insufficient.

6.1 Documentation Amount and Importance

Agile practitioners in our multinational sample perceive documentation as important for their projects and find there is too little internal documentation available in their projects. Documentation seems neglected by original Scrum literature. It is mentioned but not anchored in the original process [18] which relies heavily on verbal communication. But verbal communication is susceptible to lapses of memory and after some time it get progressively harder to recall design rationale. This problem is compounded by team turnaround. As a member of T6 comments,*"Code comments are used in an effective manner. During project development any needed documentation is generally available. However, finding documentation for older projects is not always easy, and sometimes this documentation is missing."*

The correlation analysis confirms a predominant top-down dissemination in agile settings where practitioners in senior technical roles spend significantly more time on writing documentation. As mentioned in the interviews, product owners usually write requirements and user stories. From there on it is up to the team, it takes care of the implementation and controls what is documented.

Some agile scholars propose documentation to be *"light but sufficient"* [4]. Sufficient, however, it doesn't seem to be, as none of the researched teams noted that they perceive documentation as such. Writing less than 15 minutes (Fig. 2) of documentation daily seems not enough for the produced software and is surely not enough to sustain a co-developed artifact, produced in-line with emerging code. Surprisingly, the vast majority of respondents perceives documentation as important or very important.

Table 2: Descriptive variables, team results (x) and agreement (σ^2)

		T1	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**avg. agr.**
country		UK	US	UK	NO	NL	SE	IN	NZ	
*team size (*pers.*)*		4	9	5	12	6	4	8	6	
collected answers		4	6	5	6	5	3	8	4	
*avg. exp. (*yrs.*)*		7.75	13.7	6.6	12.7	2.6	10	7	3.5	
spacial distribution		co-loc.	co-loc.	co-loc.	distrib.	co-loc.	co-loc.	distrib.	co-loc.	
documentation tool		Wiki	Confluence Wiki	Google Docs	-	-	Wiki	Confluence Wiki	-	
perceived doc. amount	x	**-0.25**	**-0.50**	**-0.40**	**-1.30**	**-1.00**	**-0.75**	**-0.13**	**0**	
	σ^2	*(0.19)*	*(0.25)*	*(1.44)*	*(0.89)*	*(0.40)*	*(0.67)*	*(0.61)*	*(0)*	*(.56)*
perceived eff.. finding doc	x	**0.65**	**0.76**	**0.44**	**0.60**	**0.52**	**0.50**	**0.75**	**0.45**	
	σ^2	*(0.69)*	*(0.47)*	*(0.16)*	*(1.33)*	*(1.44)*	*(0.89)*	*(0.69)*	*(0.69)*	*(.80)*
perceived importance artif.	x	**1.00**	**0.70**	**0.76**	**0.57**	**0.72**	**0.75**	**0.7**	**0.85**	
	σ^2	*(0)*	*(2.25)*	*(0.16)*	*(0.47)*	*(1.04)*	*(0.67)*	*(0.50)*	*(0.69)*	*(.72)*
average agreement	σ^2	*(.29)*	*(.99)*	*(.59)*	*(.90)*	*(.96)*	*(.74)*	*(.60)*	*(.46)*	*(.69)*

6.2 Software, More Than a Backchannel

The teams in our study predominantly adopt collaboration tools to document and share agile artifacts such as user stories or sprint backlogs. An interesting finding is the perceived importance and application of software that directly aim to support Scrum. This is surprising to that extend that one could expect the sufficiency of direct communication and physical artifacts. Convenient handling of agile artifacts and distributed settings seems to be one reason here. *"We have good experience using physical artifacts for local projects, but most of our projects are multi location and require an electronic solution."*, says the ScrumMaster of Team T4.

The perceived usability of solutions for electronic discussion showed the smallest gap among all categories, meaning that the participants find the current solutions very usable. According to comments from team members, the solutions surpass the expectations of a pure *"backchannel"* (Team T3). The growing instant messaging culture seems to make a contribution here and Skype has been the most named tool in this category. A quote from the ScrumMaster of Team T7: *"Communication with the team and our client worked very well when we decided to move away from "voice" conversations (accents, network latencies, time wasted setting up conference phones) to text chats. Even though they can take substantially longer, logs are permanent and we found it easier to share documentation, make decisions and stick to them."*.

7. CONCLUSIONS

In this paper we present the results of an empirical study on documentation in agile Scrum software development teams. Executed in a multidimensional manner we analyze the collected data sets consisting of quantitative team and global samples as well as qualitative interview questions. Our findings stem from a representative data set of agile practitioners and international teams and warrant further study.

One of our main findings is that documentation alone is insufficient. While agile methods recommend to make "lean" documentation, suggesting that documentation should only include information that is used, we found that agile software development practitioners perceive their internal documentation as important but that they feel that too little documentation is available. Analogously to the observations of Clear [3], we found that documentation is rather seen as a burden than a co-created (core) artifact and found support to the perceptions in literature that without ensuring a proper documentation process agile methods can cause major knowledge loss during or after system development [1,17].

We found that agile teams adopt collaboration tools to share and work on agile artifacts and that Scrum dedicated software is perceived as important and helpful to support the method. We found that instant messaging is perceived as a helpful "backchannel" and supports documenting decisions. We can conclude that integration of software tools into the process is crucial. Lightweight solutions such as Wikis or Google Docs are prominent, their adoption however needs further research (compare Tab. 2).

When interviewing practitioners the authors often found abroad interest for agile methods. Discussing their tools and routines it was the first time developers would address a software development process with passion. This, however, seems not yet true for agile documentation, and future research needs to address an appropriate incorporation within the process to make knowledge transfer truly agile *and* sufficient.

7.1 Validity Considerations

Due to the low amount of data sets containing 79 individuals conclusions were drawn carefully. As we base our evidence on small team data sets, we have improved the transparency of data by adding the variance of given answers among the team members. Throughout the whole process of data collection, we encouraged participants to give realistic answers and emphasized the anonymous treatment of data to establish a reasonable level of trust and to reduce bias. No results other than the processed outcome for the whole team would be distributed or given to their superiors.

While we provided personalized links for each team member to ensure the consistency of input, no personal details were stored or used within the examination. We address the bias of participants towards positively perceived answers (socially desirable responding (SDR) [15]) by anonymous self-administration of questions through a computer, meaning that when the subjects' personal details are not required the person does not feel directly and personally involved in the answers he or she is going to give. This effect is further neutralized through the impersonality of the machine. To validate the perceptions collected via quantitative questionnaire we collected open-ended interview responses from ScrumMasters as proposed by Miles and Huberman [12]. We must be careful to make conclusions regarding the reported time spent documenting as we are unsure what the quality of the resulting documentation was. Furter conclusions can only be drawn when we review the documentation itself.

8. FUTURE WORK

The results of this study point to a number of directions to pursue future research. First, our study in a reality-check manner collects how agile practitioners perceive internal documentation in their projects, further in-depth studies thus are necessary to address the quality of created documentation along with the process and team routines. Second, it has been argued that the lack of architectural focus leads to suboptimal design-decisions [11] in agile methods. UML has been found useful during the design phase but is considered as heavyweight by many practitioners. Further research on incorporation of lightweight modeling methods such as the ICONIX process [16] or the Active Documentation Software Design (ADSD) [17] is recommended. Physical artifacts are widely accepted in the agile community, perhaps there are also possibilities for a more physical extension of UML. Third, previous studies [2] found that Wiki systems have been successfully applied to improve internal documentation efficiency, and have been found widely applied in this study. Further research on the adoption of lightweight documentation systems such as Google Docs or a Wiki is necessary.

9. REFERENCES

[1] P. Abrahamsson, J. Warsta, M. T. Siponen, and J. Ronkainen. New directions on agile methods: a comparative analysis. In *Proceedings of the 25th International Conference on Software Engineering*, ICSE 2003, pages 244–254, Washington, DC, USA, 2003. IEEE Computer Society.

[2] A. Aguiar and G. David. Wikiwiki weaving heterogeneous software artifacts. In *Proceedings of the 2005 international symposium on Wikis*, WikiSym 2005, pages 67–74, New York, NY, USA, 2005. ACM.

[3] T. Clear. Documentation and agile methods: striking a balance. *SIGCSE Bulletin*, 35(2):12–13, 2003.

[4] A. Cockburn and J. Highsmith. Agile software development: The people factor. *Computer*, 34:131–133, November 2001.

[5] S. Dutta and L. N. van Wassenhove. Report on the 1995/1996 software excellence survey. Working Paper 96/52/TM, INSEAD, Boulevard de Constance, 77305 Fontainebleau, France, 1996.

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

[7] T. Dybå and N. B. Moe. Rethinking the concept of software process assessment. In *Proceedings of European Software Process Improvement Conference (EuroSPI 1999)*, Pori, Finland, 1999.

[8] A. Forward and T. Lethbridge. The relevance of software documentation, tools and technologies: a survey. In *ACM Symposium on Document Engineering*, pages 26–33, 2002.

[9] J. Highsmith and M. Fowler. The agile manifesto. *Software Development Magazine*, 9(8):29–30, 2001.

[10] T. Lethbridge, J. Singer, and A. Forward. How software engineers use documentation: The state of the practice. *IEEE Software*, 20(6):35–39, 2003.

[11] P. McBreen. *Questioning Extreme Programming*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[12] M. Miles and A. Huberman. *Qualitative Data Analysis : An Expanded Sourcebook*. Sage, Thousand Oaks, 2. edition, 1994.

[13] A. Mockus and D. M. Weiss. Globalization by chunking: A quantitative approach. *IEEE Software*, 18(2), 2001.

[14] S. Nerur and V. Balijepally. Theoretical reflections on agile development methodologies. *Communications of the ACM*, 50:79–83, March 2007.

[15] D. L. Paulhus. *The role of constructs in psychological and educational measurement*, chapter Socially desirable responding: the evolution of a construct, pages 46–69. Mahwah NJ: Lawrence Erlbaum, 2002.

[16] D. Rosenberg and K. Scott. *Use case driven object modeling with UML: a practical approach*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[17] E. Rubin and H. Rubin. Supporting agile software development through active documentation. *Requirements Engineering*, 16:117–132, 2011.

[18] K. Schwaber and M. Beedle. *Agile Software Development with Scrum*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2001.

[19] H. Sharp, H. Robinson, and M. Petre. The role of physical artefacts in agile software development: Two complementary perspectives. *Interacting with Computers*, 21:108–116, January 2009.

[20] A. Sillitti, M. Ceschi, B. Russo, and G. Succi. Managing uncertainty in requirements: A survey in documentation-driven and agile companies. In *Proceedings of the 11th IEEE International Software Metrics Symposium*, Washington, DC, USA, 2005. IEEE Computer Society.

[21] C. J. Stettina and W. Heijstek. Five agile factors: Helping self-management to self-reflect. In *Proceedings of European Software Process Improvement Conference (EuroSPI 2011)*, Roskilde, Denmark, 2011.