An Exploratory Study on Handling Requirements and Acceptance Test Documentation in Industry

Sofija Hotomski, Eya Ben Charrada and Martin Glinz
Department of Informatics
University of Zurich
Switzerland
Email: {hotomski,charrada,glinz}@ifi.uzh.ch

***Abstract*—With the emergence and spread of agile processes, the practices of writing and maintaining documentation have drastically changed in the last decade. In this work, we performed a qualitative study to explore the current practices for managing two related types of software documentation: requirements and acceptance tests. We interviewed twenty practitioners from seventeen business units in fifteen companies to investigate the companies' practices for writing, maintaining and linking requirements and acceptance test documentation. The study yields interesting and partially unexpected results. For example, we had expected that tests would be more extensively documented than requirements, while we found a strong linear correlation between the number of requirements and tests in our sample. We also found that technical people are usually not involved in the requirements engineering activities, which often results in misunderstood or underestimated requirements. Acceptance tests are written, in many cases, based on requirements that are not necessarily detailed enough. Also, acceptance tests are not regularly maintained, which occasionally results in confusing features and bugs.**

I. Introduction

Requirements engineering and testing are related activities that show much synergy. Linking the testing and requirements activities can benefit both sides and save money and time [1]. For example, conceiving test scenarios concurrently with the elicitation and documentation of requirements helps uncover potential problems earlier, thus leading to better software quality [2].

Among the different types of tests, *acceptance tests* are those which are most related to requirements as they strive for *"comparing the program to its initial requirements and the current needs of its end users"* [3]. Due to this strong relation, several advocates of agile software development even suggest to write requirements in the form of acceptance tests [4][5].

In this work, we explore the current practices related to requirements engineering and acceptance testing.

Whenever we talk about tests, testing, or test documentation in the remainder of this paper, we always mean *acceptance* testing. The term "requirements" denotes *product* requirements in the context of this paper.

In contrast to previous studies on requirements and testing (e.g [6][7]) our study focuses specifically on the *documentation aspect* of the requirements and acceptance testing activities. Concretely, we investigated how the requirements documentation and the tests documentation are written, by whom they are written, in what format they are specified and how useful they are for the requirements engineering and acceptance testing activities.

We also examined how the documentation is evolved and maintained. For instance, we looked for the main triggers for change in requirements and tests, the current update procedures and the factors that support or hinder the documentation update. The results of this paper are based on semi-structured interviews with twenty practitioners.

The paper is organized as follows. In Section II we describe the research methodology, including study design, research questions and threats to validity. Section III presents the key findings of our study, which are then discussed in Section IV. Section V presents related work. Section VI concludes the paper with a summary and directions for future work.

II. Research Goal and Methodology

Our goal is to understand the current practice of requirements and acceptance test documentation management, as well as the challenges faced during this process. As a first step towards this goal, we conducted a qualitative exploratory study, using semi-structured interviews [8]. The interviews were based on a predefined interview instrument[1], composed of five parts. The first and the second part focus on the characterization of the interviewees and the companies, respectively. In the third part we investigate the company practices for writing requirements and test documentation. Next, we explore the practices related to the evolution and update of the documentation. Finally, we asked the participants about the characteristics of the perfect tool for supporting the documentation management. We also gave the participants the opportunity to freely talk about matters that they think could be related to the topic of the interview and which have not been addressed.

A. Research Questions

From our research goal, we derived the following two research questions:

RQ1. *How are requirements and test documents written in practice?* With this question, we aimed at exploring how

[1] http://www.ifi.uzh.ch/rerg/people/hotomski/InterviewInstrument.pdf

requirements and test documents are written, who writes them and in which format they are specified. Besides, we looked into the factors that influence the documentation practices, such as the company organization, the used process models and the available tools.

RQ2. *How are requirements and acceptance tests updated and what difficulties are faced?* With this question, we aimed at examining how the requirements documentation and the test documentation are maintained and evolved. For example, we investigated what are the main triggers for change in requirements and in tests. We also explored the current documentation update practices as well as the factors that support or hinder the documentation update.

B. Study design

1) Preparation: Based on a review of the existing literature about the management and use of software documentation in practice, we shaped our research questions and created the interview instrument to answer the research questions. The interview instrument was validated and improved in two rounds: first it was reviewed and discussed within our research group. Second, we conducted two pilot interviews, one with a researcher from our group and one with a practitioner who was not involved in the study design.

2) Participants: For the selection of participants, we used a purposeful sampling strategy [8]. The selected participants needed to satisfy the following two criteria: [C1] they work for a company in which requirements and test documentation exist in some form, [C2] they are knowledgeable about and/or directly involved in the requirements engineering and design processes and the test planning process.

We interviewed twenty practitioners from seventeen business units in fifteen companies, located in seven countries. The interviewees had between two and twelve years of experience in their respective fields. The geographic distribution and the size of the companies are shown in Figure 1 and Figure 2, respectively.

Companies C4 and C7 both consist of two units that have significantly different characteristics in terms of the software process model applied, the team structure and the type of the produced software. In both companies we interviewed one representative from each unit separately. Therefore we treated these units as separate companies, named C4.a, C4.b and C7.a, C7.b, respectively. Thus, for simplicity, we report about a total of seventeen companies in the remainder of this paper.

We interviewed the participants individually, except for companies C3, C8 and C12, where we interviewed participants in pairs. Since these interviewees had different roles in their respective companies, we could obtain two complementary views of the practices in these companies. An overview of the participants and their companies is shown in Table I.

Fig. 1. Geographical distribution of companies

TABLE I
OVERVIEW OF THE PARTICIPANTS AND THEIR COMPANIES

Comp.	ID & role	Domain	Type	SPM
C1	P1 - Test developer	Outsourcing	project	S
C2	P2 - Test engineer	Games and technology	product	S
C3	P3 - Test developer P4 - Prod. manager	IT consulting, outsourcing	project	S
C4.a C4.b	P5 - Prod. manager P6 - Test lead	Electricity distribution, automation management	product project	SAF W
C5	P7 - Test engineer	Technology services, custom application	product	S
C6	P8 - IT Consultant	ERP systems, ECM systems	project	W
C7.a	P9 - IT Consultant	Cloud storage services	product	S
C7.b	P10 - IT Consultant	Performance optimization services, mobile app development, outsourcing	project	K
C8	P11 - RE engineer, P12 - Software dev.	App for car sharing, business consulting	project	S
C9	P13 - Proj. manager	IT services in finance, insurance and media	project	S
C10	P14 - Business analyst	Assurance, consulting, financial, legal	project	S
C11	P15 - Test engineer	Storage platform and services, security	product	S
C12	P16 - Proj. manager, P17 - Test engineer	Medical and logistics software	project	S
C13	P18 - Business analyst	Health insurance	project	W
C14	P19 - QA/Test lead	Payment services	project	S
C15	P20 - Business analyst	Invoicing, marketing, customer care platforms	project	S

SPM - Software process model S - Scrum K - Kanban
SAF - Scaled Agile Framework W - Waterfall

TABLE II
DIFFERENCES BETWEEN PRODUCT AND PROJECT ORIENTED COMPANIES

Product-oriented company	Project-oriented company
The software is developed for and driven by the market	Bespoke software is developed for a specific customer
The company specifies the requirements based on a market analysis	Requirements are elicited from the stakeholders
Developing the software is a continuous and iterative process	The development ends when the stakeholders' requirements are met and the product is delivered
Developing the software is a long-term process	Developing the software is a short-term process
The company owns the software	The customer owns the software

We distinguish two types of companies: *product-oriented* companies develop market-driven software products, while *project-oriented* ones develop bespoke software for specific customers. The main differences between the two types are summarized in Table II.

Only three of the interviewed companies use a waterfall

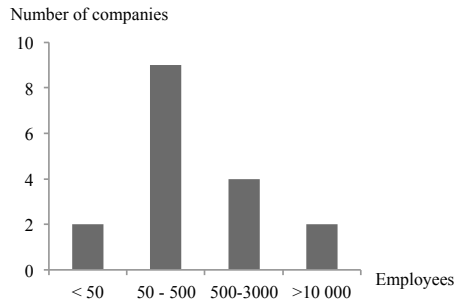

Fig. 2. Size of the companies (number of employees)

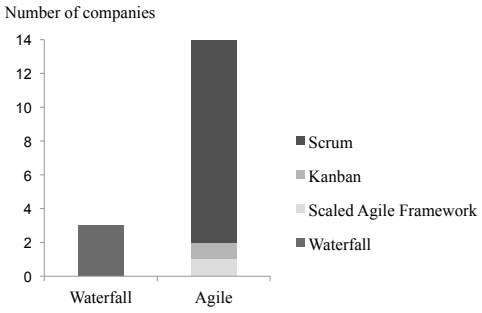

Fig. 3. Software process models used

software process model; all others employ different types of agile development models (see Figure 3).

3) Data collection and analysis: The interviews were carried out between July and November 2015. All interviews were conducted by the first author. Face-to-face interviews were used whenever possible. In total we had ten face-to-face interviews and seven interviews via Skype. The interviews were conducted in English, except for two interviews, which were in Serbian because the interviewees felt more comfortable to use their native language.

The duration of the interviews was between 55 and 120 minutes. The face-to-face interviews were generally longer, because participants showed us their documentation. All interviews were audio recorded (with permission from the interviewees). For analyzing the data, the interviews were transcribed and coded. Serbian interviews were translated to English during transcription. We created an initial list of codes based on the interview questions and complemented it with codes that emerged during data analysis. We used the codes to group related answers, compare them to each other and derive hypotheses.

4) Threats to validity: As for every qualitative study, the validity of the findings is subject to several threats [9].

Below we explain the main design decisions we made with the aim of limiting the potential threats.

Construct validity is threatened if the answers from the participants do not accurately reflect the real practice. This could be due to the interviewees not feeling comfortable talking about certain topics or to the interviewer influencing the discussion. To limit these threats, we avoided judgement and evaluations during the discussions, we assured the interviewees about the anonymity of the study and we abstained from communicating our hypotheses to the participants.

In order to collect reliable data about the companies' practices, we only selected participants who are knowledgeable about requirements and testing procedures on the company level and not only within their team (criterion C2). When we interviewed participants from different units of the same company, we asked them not to talk about the interview to others in order to avoid bias. Unclear questions and misunderstandings between the interviewee and interviewers are also possible threats that cannot be completely ruled out. To limit these threats, we discussed the interview questions with RE researchers from our research group before conducting the interviews and conducted two pilot interviews. Thus we could revise the questions that turned out to be unclear or generated misunderstandings. We also tried re-explaining the questions differently to the interviewees whenever we thought that the participant misunderstood us.

However, reliability threats that relate to researcher bias cannot be completely ruled out, because the interviews and analysis were conducted by only one researcher.

External validity issues are related to the inability to generalize the results of the study beyond the studied companies. In order to achieve reasonable generalizability, during the sampling, we selected companies that have different characteristics in terms of size, domain of activity, internal organization and location.

Internal validity issues appear if a causal relationship between treatment and outcome is wrongly established. Possible factors that could negatively impact the internal validity in our case are selection bias and interview instrument change. Although we started the selection of participants with our personal contacts, the network was soon spread with indirect contacts, thus decreasing the selection bias threat. All participants were contacted directly and none of them declined. This decreases the threat of having only participants who are interested in the topic. Regarding the interview instrument change, we evaluated the interview questions with two pilot interviews and revised them before starting the real data collection. So we avoided any changes to to the interview instrument during the actual interviews.

III. Key Findings

In this section, we present the key findings of our study, grouped by research question. As our our findings are not statistically representative, we formulate them in terms of hypotheses, for which we have evidence from our study data.

A. Writing the requirements and test documentation in practice - RQ1

Hypothesis H1.1*: Technical people are not involved in the requirements engineering activities.*

Except for one company (C3), where the requirements are provided by the client directly, all companies have the requirements specified by a person in a managerial position, such as project manager, product owner, business analyst or

consultant. These managers may have a technical background, but their engagement is only managerial and they are not aware of technical details. Managers elicit, specify and maintain the requirements based on communication with stakeholders or their own, market-driven views. Engineers, who will be responsible for the development and testing of the product, are not involved in any of these activities. As managers do not always have the necessary technical knowledge about the project, this results in two major problems: (1) the requirements are often misunderstood and (2) their complexity is underestimated.

A few participants mentioned the amount of work as a possible reason for not involving technical people in the requirements engineering activities: *"One person cannot worry about what is needed and also how it will be implemented. That is just too much"* (P20). Additionally, in many companies, a strict separation of roles is applied, which limits the domains of responsibility of the practitioners to only one aspect such as design, implementation or testing. This is especially true for participants who are involved in many projects.

The organizational hierarchy could also have a negative impact here: *"People do not want to ruin the hierarchy. Business people are on a higher level of hierarchy, even if they are not capable sometimes to make the right decisions. Giving the opportunity to the developers or testers to decide about features would bring less power to the business people and they are not willing to accept that"* (P3). In one company, finances are the problem. Since their customers are spread over different continents, *"it is costly that the whole team attends design meetings and this is why only business people are involved"* (P19).

Hypothesis H1.2*: The format of requirements specifications depends on the software process applied.*

The companies following a waterfall process document requirements in a traditional format: In C4.b and C6, the requirements are written in prose and stored as text documents. In C6, these documents contain not only the requirements, but also test cases. In C13, the requirements are specified as structured use cases: *"It is much easier to keep requirements up to date if they are not in prose text, but in form of use cases, because it is easier to spot the part that needs to be changed"* (P18).

In the companies employing an agile process, user stories [10] are the most common way for documenting requirements. User stories contain acceptance criteria, which are further used for writing the acceptance tests. Eight companies use user stories only, four combine user stories with sticky notes or index cards and two companies augment the user stories with detailed explanations. Sometimes, requirements are written on index cards or sticky notes only, which are then arranged on walls or tables to facilitate planning and discussion (C2). However, in none of the companies this was a standard format for requirements.

Most of the companies store and manage user stories in JIRA or Confluence. An overview of the tools and repositories used for managing and storing requirements and test documentation is given in Figure 4. Some companies use more than one tool for managing acceptance tests.

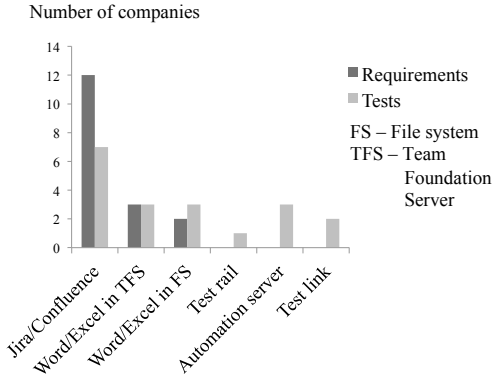

Fig. 4. Repositories and tools for storing and managing requirements and acceptance test documentation in the interviewed companies

None of the agile companies produce a vision document. Companies C8 and C9 produce requirements documents that contain a detailed explanation of every requirement, in addition to user stories. However, these requirements documents do not contain information regarding the strategic intent of the product to be built and thus, these are no vision documents [10]. In C9, more detailed requirements documentation is requested from the clients: *"Sometimes, clients are willing to apply agile methodologies in terms of continuous delivery of software and their involvement during all phases, but they insist on having more documentation in traditional form of prose text"* (P13). However, due to time limitations and difficulties to handle changes in an unstructured text, this document usually becomes outdated when the requirements change. In C8, well established procedures and habits result in having more detailed requirements documentation, which will be discussed later (see Hypothesis H2.5).

Hypothesis H1.3*: There is a strong linear correlation between the number of requirements and acceptance tests.*

In all companies, requirements are specified first and each test case is derived from the associated requirement. The companies aim at covering each requirement by at least one acceptance test case. However, the results show that the number of acceptance tests is only marginally larger than the number of requirements. In none of the companies, requirements are covered by more than two acceptance test cases. Moreover, in C3 and C6, the number of tests and requirements is reported to be equal: there is only one test case for each requirement. Figure 5 illustrates the strong linear correlation between the number of requirements and acceptance tests per project, with an average ratio of 1:1.27.

The estimates were given by the participants.

Acceptance test-driven development (ATDD), is reported as a good practice by practitioners. Therefore, except for one company (C4.a), where the tests are written at the end of every sprint, all companies following an agile process write the acceptance tests before the source code, whenever possible. Tests would be written in parallel with the source code only

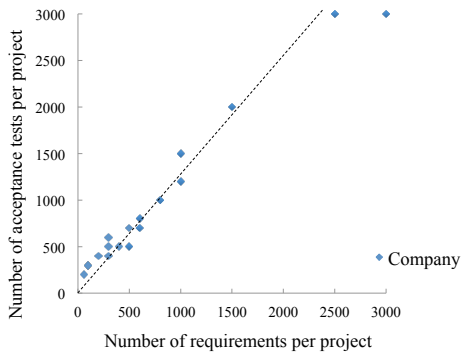

Fig. 5. Approximate number of requirements and tests per project.The dotted line shows the average ratio.

in the case of time constraints. Writing the acceptance tests before the code is the first principle of ATDD. However, the second principle of ATDD, which states that requirements documentation should be specified in the form of acceptance tests [11], is not applied among the interviewed companies, because every test case covers one, separately written requirement.

Hypothesis H1.4*: Test engineers write acceptance tests based on the requirements that are not necessarily complete.*

According to Humble and Farley [12], testing is a cross-functional activity that involves the whole team and should be continuously done from the beginning of the project. However, our study shows that tests are usually specified only by a dedicated, technical role, such as test engineer or, sometimes, a developer (C7.a). The rest of the team is not included in this process. Only in companies C6 and C13 tests were specified by the same person who specified requirements. That person has a managerial role. In C10 the acceptance test were specified by the client.

When writing the test documentation, test engineers first refer to the requirements specification. Only if something is not clear, they communicate with managers or developers. However, participants admit that this additional communication is often needed: the test cases cannot be derived from the requirements documentation alone, because it is not detailed enough. *"We have requirements in form of user stories with acceptance criteria defined. Sometimes, this is not enough for me to write the test cases, so I need to communicate with developers or the product owner. They are not always easy to catch, since we are distributed and people are busy. Then, I just wait for the next weekly meeting"* (P7). Participants also said that sometimes this results in postponing deadlines.

Oral communication is not always the optimal solution. In fact, participants stated that having a more detailed requirements document would also help them in writing tests.

Hypothesis H1.5*: Acceptance tests are mostly written manually, using the typical format.*

Typical manual tests are written in natural language and contain the usual elements: *name*, *prerequisites*, *ID*, *description*, *steps (actions)* and *expected results*. Except for one company (C1), which uses more automated tests than manual ones, all companies either use manual tests only (11 companies) or manual tests combined with automated tests (C2, C4, C10).

The automated tests contain the scenario and the code related to this scenario via annotations. A scenario is written in natural language using the *Given-When-Then* template [13]. Therefore, the test specification for automated tests does not differ much from the manual test specification in terms of structure and format:

- *Given* describes the state of the world before starting the behavior. It corresponds to *prerequisites* in manual tests.
- *When* describes the key action the user performs. It corresponds to *steps (actions)* in manual tests.
- *Then* describes the expected outcome of the behavior. It corresponds to *expected result* in manual tests.

Cucumber and Selenium are the main tools that companies use for creating and executing the automated tests. None of the companies uses tools for automatically generating tests.

Hypothesis H1.6*: Mixed and centralized teams produce less documentation than distributed ones.*

We found a relation between the amount of documentation and the team organization and structure. The structure of the teams differed much from one company to another. While in some companies teams are mixed, in others they are separate. By separate we mean that each team has a specific responsibility for the software, such as development, testing and QA, deployment or technical writing. Mixed teams include people responsible for various roles, or a representative for each of the roles. In distributed teams, the members are located in different cities or countries. In centralized teams, the members are in the same office or building.

In agile companies, teams are mostly centralized. In fact, in some companies, testers and developers were co-located, while the product owner and scrum master were distributed. In distributed teams, daily or weekly meetings are regularly held via tools for web conferencing, but both managers and engineers rely more on the existing documentation when they write or execute tests, create tasks or implement features. This usually results in these teams having more detailed documentation than co-located ones.

In mixed and centralized teams, people rely more on (informal) communication than in distributed or separate teams. In two agile companies, participants said that requirements are documented as user stories, but in some cases, the requirements documentation exists only in the form of sketches, drawings on whiteboards or sticky notes: *"Sometimes we do not even have requirements in form of user stories. We capture only what is important in form of notes and drawings and discuss them during meetings, so we know how to test the features and developers know how to implement them. User stories are only a call for discussion. This is possible because we are all here; we communicate"* (P2). By "call for discussion", P2 meant that the user stories only trigger further communication and are used as a reminder of current sprint elements that need to be discussed during the meeting.

B. Updating the requirements and acceptance test documentation in practice - RQ2

Hypothesis H2.1*: Changes of requirements are mainly triggered by the inability to implement the specified requirements.*

When discussing the triggers for changing requirements, we found that the majority of the change requests that impact requirements did not come from external stakeholders (e.g., clients), but rather from the internal ones. In fact, as technical engineers were usually not involved in the elicitation of the requirements (see Hypothesis H1.1), the complexity of the requirements was frequently underestimated: *"Business people often see only where we should get, but not what it takes to get us there. They do not know if something is possible to implement and how."* (P12). Underestimated complexity often results in the inability to implement the specified requirements. In nine companies, this inability was reported as the cause for more than 70% of the changes in requirements.

Internal change requests can also be triggered by changes in the used platforms, frameworks and tools. Sometimes, such changes in the infrastructure require a redesign of the software architecture and also the requirements, as reported in two companies.

Changes in requirements can also be requested by external stakeholders if they previously were not sure about what they need or if they changed their mind later on: *"At the beginning, clients sometimes cannot imagine how something will really look like or how it will really work and fit to the rest of the system."* (P19). Furthermore, changes may happen due to misunderstandings between the people eliciting the requirements and the external stakeholders.

The main triggers for changing requirements in the investigated companies are presented in Table III. In these companies, the trigger contributes to more than 70% of the requirements changes. In C1, C3 and C15, internal and external triggers impact requirements equally. Therefore, these companies are not included in Table III.

TABLE III
MAIN TRIGGERS FOR CHANGING REQUIREMENTS DOCUMENTS IN COMPANIES

Trigger type	Reason	Companies
Internal	Underestimated complexity of requirements	C2, C4.b, C5, C6, C7.a, C7.b, C8, C9, C11
	Infrastructure/architectural change	C4.a, C13
External	Clients are not sure about their needs at the beginning	C12, C10
	Clients change their mind about what they require or requirements are misunderstood	C14

Hypothesis H2.2*: Existing requirements are updated in waterfall-style projects while new user stories are created in agile ones.*

In all three waterfall projects that we covered in the interviews, the requirements are usually kept up-to-date. In two of them, the participants claimed that requirements are always changed before the code is changed: *"We update the documentation first, and later the code. Otherwise there is no point to have any documentation if nobody can trust it. We put a lot of effort sometimes to track changes, but that is all part of the job"* (P6). In the third company, the requirements are updated after the code is changed, and mainly because the external stakeholders require it.

Agile companies handle the requirements differently. In fact, instead of updating existing requirements, they usually create new user stories whenever a change is required. Even if the change occurs in the middle of a sprint, the procedure is the same: the work on the old user story is stopped, and a new user story is created, prioritized and put in the backlog. The new user story might then be assigned to the next sprint or not, based on its priority: *"After change and prioritization, that new user story is sometimes not developed for months, or even years. By then we will for sure forget what was the original requirement, but do we even care?"* (P5).

An exception is made in the case of minor changes, as no new user story is created: *"We will make new user stories only if the change is not, for example, changing the label from 'Email' to 'E-mail' or a similar cosmetic change. Otherwise, although we know this is not by the book, we will change the existing user story and continue the sprint, because it would take much more time to create a new user story in this case."* (P16). Similarly, P9 explained: *"We will create a new user story instead of updating the existing one only if the change is greater than 20%"* (P9).

Although most of the companies keep the old user stories in the backlog, a few companies do not. In fact, in three companies, the user stories are cut from the backlog into the sprint instead of being copied. When the story needs to be changed, a new one is created, placed in the backlog, and the one from the sprint is deleted. In one company (C12), user stories are copied from the backlog into the sprint, but deleted from both, backlog and sprint, when they are replaced by new ones.

Only in company C8 new users stories are linked to the old ones in the backlog, which is also the original requirement. The others reported no need for linking the old and the new user stories: *"We do not need to know what was the requirement in the past. The only relevant reference point is the source code. It [the source code] is the requirement from the past and then we start from that reference point and make changes"* (P11).

Hypothesis H2.3*: Acceptance test documentation is less frequently updated in teams with good communication.*

The practices regarding the update of acceptance tests vary considerably among the companies. Some companies keep their acceptance tests regularly up-to-date by updating them either before the code is changed or in parallel with the code. However, about half of the participants mentioned that in their projects, the acceptance tests are only updated after the code is changed. For these projects, it also happens that the acceptance

tests do not get updated. This is mainly because the testers rely much on the communication with the engineers and developers to run the tests. This is especially true for small and centralized teams where the testers gather information about the changes in meetings with the developers and managers, manually test the changed features, and often leave the test documentation unchanged.

Although relying on informal communication works rather well with small and centralized teams, sometimes this also leads to problems. For instance, when the change is not well communicated, a feature could be mistaken for a bug: *"Sometimes we need a lot of time to realize that it is not a bug, but a feature. We need to communicate with developers and check with the product owner in order to be sure that the requirement has changed, which can take quite some time" (P19)*.

Another cause for outdated test documentation is the use of exploratory testing [14], where testers mainly rely on their own experience and intuition in order to find weak areas and, therefore, do not use the test documentation much. Our study confirms the usage of exploratory testing, especially in centralized teams with good communication.

Hypothesis H2.4*: The update of documents is hindered by missing traceability links and limited tool support.*

The majority of participants agreed that without the support of traceability links, updating the requirements is a cumbersome manual task: *"One requirement is usually spread over many user stories. Then, if requirements change, we need to go through all user stories and figure out which of them cover that requirement. Moreover, every user story is covered by one or sometimes even more tests. Finding the appropriate tests when the requirement is changed is sometimes an art"* (P10).

Although modern management tools like Jira, TFS or Confluence offer features for creating traceability links between artifacts, these features are not used in more than half of the companies. In one company, the interviewees were even not aware of the existence of such features. Other participants who knew about them would still not use them due to the lack of time: *"We know about the option, but sometimes documents are not linked because of a lack of time. It depends on how much time the consultant has"* (P19).

These tools provide limited or even non-existent support for change propagation and updates. For example, in Jira and Confluence, there is a change notification option. In more than half of the companies that use these tools, the notification option is not used, but practitioners rather choose to rely on communication to find out that some documentation artifacts need to be updated: *"Even if we get an email automatically when some artifacts change, we can see in that email only what has been changed and not why, so we do not know what to do with that change without additional communication. Also, we do not know what else is impacted. We anyway need to check the changed artifact, to see which artifacts are impacted and to manually search one by one in order to change them. This is why the email option does not help much and we decided not to use it, because it will only spam us"* (P20).

Only one participant mentioned that considering the traceability links is unimportant and unnecessary: *"We have small projects and small teams. Everything is based on communication, so we discuss everything in our daily meetings. This is how we know which tests cover which requirements and we do not need to spend additional time to link them [requirements and tests] in tools"* (P17).

Participants stated that a perfect tool for handling requirements and test documentation "keeps all documents at the same place" (P4), "is able to automatically detect impacted test cases when requirements change" (P4, P7), "has a notification system providing a decent level of information" (P20, P6), "has an integrated chat to support better communication" (P4), and "is able to automatically generate test cases from code and requirements" (P3, P7, P4).

Hypothesis H2.5*: Organizational culture has a strong impact on the quality and quantity of documentation.*

In the previous paragraphs, we have discussed factors that influence the quality and quantity of documentation in companies, such as the distribution of the teams and the software development model. However, we also found that in one company (C8), organizational culture has a stronger impact on the quality and quantity of documentation than the other factors discussed above.

In C8, a product-oriented, agile company, the amount of documentation is much higher than in the other companies. The requirements are first specified in the form of text documents that contain descriptions of the features. These descriptions are then transformed into user stories with defined acceptance criteria. The descriptions and the user stories are then used to write acceptance tests. In this company, all the documents are linked to each other and stored in their document management tool set. If the requirements change, then the original user stories are kept in the backlog and new user stories are created and linked to the original or previous version. The process is part of the organizational culture of the company and is therefore followed by the employees: *"This is how we worked from the beginning"* (P12), *"I joined this company three months ago. The first thing they showed me is this book [the book written by CEO of the company, which contains a set of process patterns that worked for them in the past, with an aim to make all workers familiar with those patterns]. Then you just follow the good practices and you realize quickly that it works"* (P11).

IV. Discussion

The main findings of this study are summarized in Table IV. In this section, we relate our findings to each other as well as to similar earlier studies.

A. Engineering could benefit from involving testers.

Several researchers and practitioners (e.g., [1][15][16]) have stressed the importance of bringing requirements and testing activities closer together and suggested that this would be beneficial for both sides. Despite the importance of such a practice, it seems not to be widely applied. In fact, for the majority

TABLE IV
SUMMARY OF THE KEY FINDINGS

1	**Requirements and tests writing practices - RQ1**
H1.1	Technical people are not involved in the requirements engineering activities.
H1.2	The format of requirements specification depends on the software process applied.
H1.3	There is a strong linear correlation between the number of requirements and tests.
H1.4	Test engineers write acceptance tests based on the requirements that are not necessarily complete.
H1.5	Acceptance tests are mostly written manually, using the typical format.
H1.6	Mixed and centralized teams produce less documentation than distributed ones.
2	**Requirements and tests updating practices - RQ2**
H2.1	Changes of requirements are mainly triggered by the inability to implement the specified requirements.
H2.2	Existing requirements are updated in waterfall-style projects while new user stories are created in agile ones.
H2.3	Acceptance test documentation is less frequently updated in teams with good communication.
H2.4	The update of documents is hindered by missing traceability links and limited tool support.
H2.5	Organizational culture has a strong impact on the quality and quantity of documentation.

of the interviewed companies, writing the requirements and writing the acceptance tests are separate activities performed by different people. The roles of the people involved in writing the requirements and the tests are summarized in Figure 6.

Not only testers are not involved, but also other internal stakeholders with technical background, such as developers, do not participate in the requirements engineering activities (H1.1). We found that this results in two major problems. First, the complexity of the requirements is underestimated, which then results in a need to change the requirements later (H2.1). Second, the specified requirements are frequently incomplete, which hinders the writing of tests (H1.4). Similar challenges have also been reported by [17][6][7]. Uusitalo et al. [7] identified the unavailability of the testers as one of the reasons for not involving testers in the requirements process. In addition to that, we also identified that the strict separation of roles and the imposed hierarchical structure negatively impacts the practice of including testers in the requirements engineering process.

B. There is a linear correlation between the number of acceptance tests and the number of requirements in all companies – tests have not replaced requirements.

In an attempt to further consolidate the requirements and testing activities, some agile advocates suggest specifying the requirements in the form of acceptance tests [11][5]. When evaluating this practice with a series of experiments, Ricca et al. [18] found it to increase the understanding of the requirements among different roles. We found that this practice

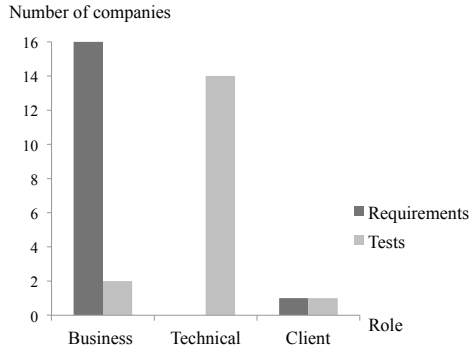

Fig. 6. Responsible roles for writing the requirements and acceptance test documentation in the companies

is not applied by any of the companies we interviewed. Although many companies are using acceptance test-driven development (H1.3), the acceptance tests are created based on the requirements (H1.4) and do not replace them (H1.3). The limited use of acceptance tests as requirements could be explained by the non-technical background of the stakeholders who perform the requirements activities. This relates to the results from the study of Bjarnson et al. [19] who found that customer involvement was one of the challenges that a company faced when using tests as requirements.

We were surprised to find that in the companies we studied, the ratio between the number of requirements and acceptance tests lies between 1:1 and 1:3.3 only, with an average of 1:1.27 (Fig. 5). This indicates that acceptance testing is not done properly, since a requirement cannot be fully covered by only one test case. Furthermore, this implies that a similar amount of effort is invested in acceptance testing and in requirements engineering.

C. The vision document is missing.

In theory, agile approaches propose to create a vision document that (1) represents the strategic intent of product that is built, (2) contains a description of the features to be implemented, (3) clearly defines the problem to be solved and the stakeholders who will benefit from the developed solution, and (4) provides a description of the software in terms of performance, platforms supported, scalability and reliability [10].

In practice, we found that such a document does not exist and even when a similar document exists, it is not kept up-to-date when the requirements change (H1.2). This indicates that in agile development, the focus is on the current goals, while the big picture of the product is missing.

D. User stories are not sufficient to support tasks dependent on the requirements specification.

With the spread of agile processes, the format used to specify requirements has evolved (H1.2). In agile companies we interviewed, user stories are now the standard way to document requirements. Although user stories have a predefined form, we found evidence that their content is not complete and detailed enough to support the development and testing

tasks. This is partially due to not involving technical people in the requirements phase (H1.1). Incomplete requirements hinder the testing process (H1.4) and result in delays. A similar problem has been reported by previous studies who found that requirements documents and test plans do not always provide enough information to perform the testing accurately and thoroughly [7] and that, due to incomplete requirements, *"testers need to guess and make up the missing information"* [6]. Of course, the quality and usefulness of the requirements does not only depend on their completeness, but also on their clarity and level of abstraction as reported by [6].

E. Agile development not only changed the format of requirements; it also brought new documentation update practices.

A survey study about the use of documentation reported that requirements documents are mostly not kept up-to-date while tests are usually updated [20]. Things seem to have changed over time. In fact, the agile practices did not only result in changing the requirements format (H1.2) but also changed the update practices for requirements and tests. For instance, we found that user stories are not updated when the requirements change, unless the change is minor. Instead of updating stories, practitioners create new ones (H2.2). Creating new user stories when changes occur, results in having requirements that are up-to-date. However, as old user stories are usually kept, the new requirements are then combined with obsolete ones. In some cases the old user stories are deleted, which means that the history of the change is lost.

The management of tests is different though as they stay outdated in some cases (H2.4). One of the reasons for not updating tests is to the use of exploratory-based testing and the reliance on informal communication. A few companies mentioned creating new tests instead of updating the existing ones in case the change is non-trivial. This could, however, be costlier than updating the old scripts as shown by [21] [22].

F. Traceability links between documents are not necessarily created.

Traceability links between requirements and test documentation support several activities in software development, testing and evolution [7] [23]. Nevertheless, such links are rarely created. Bjarnason et al. [6] report that traces between requirements and tests can be lacking (even in thought), although they are considered by practitioners as the most basic kind of traceability. This finding is confirmed in our study, as the companies we interviewed also did not have traceability links between requirements and tests. We also found that the lack of traces is an important factor that negatively influences the documentation update practice (H2.4). We identified two main reasons for missing traceability: the lack of time and the limited tool support. Although current tools provide functionalities for trace management, the creation of the links is still a manual task, which requires time and effort. Tools that researchers developed for generating candidate traces seem not to be used in practice yet.

G. Communication over documentation.

Documentation is in many cases not sufficient to support the software development activities. For example, incomplete requirements (H1.4) force the developers and the testers to recurrently communicate with the requirements engineers in order to specify the tests. Communication is also needed when the documentation is not kept up-to-date (H2.4). We also found that the quality of the communication impacts the documentation practices in companies. In fact, teams that are mixed and centralized usually produce less documentation than distributed ones (H1.6). Furthermore, test documentation seems to be less frequently updated in teams with good communication (H2.3). However, although these factors play an important role in the documentation practice, their influence does not overrule the companies' established procedures (H2.5). In fact, we found that well established procedures and habits in terms of documentation management have the greatest influence on the amount and quality of documentation within a company. In project-oriented companies, clients dictate the amount of documentation and the level of details within documents (H1.2), but our results showed that product-oriented companies can also produce high-quality documentation if the right procedures are established. Interestingly, this is the only difference we found between the practices in product-oriented vs. project-oriented companies.

V. Related Work

Over the past two decades, many researchers tried to assess the state of practice in the industry, regarding the management of software documentation. Singer [24] and Lethbridge et al. [20] investigated how software engineers use software documentation. Our study covers similar aspects, but uses a qualitative method that allows gaining a more in-depth knowledge on the documentation practice.

Several studies (e.g. [25][26][27]) have analysed the management of requirements change. These studies analysed changes in terms of their cause, type, implementation and verification. The results of our study, regarding the causes for a change in requirements, are different and surprising as we found that changes are mostly triggered by the inability to implement the specified requirements.

Stettina and Heijsteck [28] studied the time and effort needed for software documentation management in Scrum development teams. In our study, we also investigate documentation management in agile projects, but with a different focus, which is identifying the factors that influence the documentation management process.

Several researchers (e.g. [7][11][4][23]) studied the alignment between requirements engineering and testing. In this work, we also explored the alignment of requirements and tests during software evolution, with the focus on the link between requirements and acceptance test documentation.

Although the question of "how much documentation is enough", has drawn the attention of researchers since more than ten years [29], the knowledge in this field is still limited. Our study expands this knowledge by providing insight into

the documentation practices for requirements and acceptance tests in a sample of companies.

VI. Conclusion and Future Work

This paper reports the results of an exploratory study about the handling of requirements and acceptance test documentation in seventeen business units from fifteen companies. Our study shows that the processes for managing documentation have changed over the past years. Therefore, new challenges are faced, which requires further research. In this section, we discuss three of these challenges.

First, we found that writing requirements and acceptance tests are performed as two separate tasks by different people. This could result in underestimating the complexity of the requirements and also having incomplete specifications that are not sufficient for performing the testing. Therefore, practitioners need to rely much on oral or informal communication. When communication is hindered or when misunderstandings occur, it often happens that the project gets delayed. Exploring ways to bridge this communication gap is an interesting research area.

Second, in agile companies, the requirements are usually documented in the form of user stories only, while a vision document, which provides a long-term perspective, is usually missing. A deeper understanding of the risks and challenges caused by this practice is worth exploring in the future.

Third, modern documentation management tools do not provide enough support for documentation update. Although they allow tracing artifacts to each other, linking documents is still a manual task that is usually not done due to time constraints. The lack of links hinders analyzing and propagating changes among artifacts. Our study also shows that keeping the history of a change is widely seen as an unnecessary or secondary task, although it improves the quality of requirements documentation and could be used in retrospective analysis, in order to prevent repetition of mistakes.

Acknowledgment

The authors would like to thank the companies and their representatives for making this research possible. This work was partially funded by the Swiss National Science Foundation under grant 200021-157004/1.

References

[1] D. Graham, "Requirements and testing: Seven missing-link myths," *IEEE Software*, vol. 19, no. 5, pp. 15–17, 2002.

[2] D. Damian and J. Chisan, "An empirical study of the complex relationships between requirements engineering processes and other processes that lead to payoffs in productivity, quality, and risk management," *IEEE Transactions on Software Engineering*, vol. 32, no. 7, pp. 433–453, 2006.

[3] G. J. Myers, C. Sandler, and T. Badgett, *The art of software testing*, 3rd ed. John Wiley & Sons, 2011.

[4] R. C. Martin and G. Melnik, "Tests and requirements, requirements and tests: A Möbius strip", *IEEE Software*, vol.25, no.1, pp. 54-59, 2008.

[5] K. Pugh, *Lean-Agile acceptance test-driven-development*. Pearson Education, 2010.

[6] E. Bjarnason, P. Runeson, M. Borg, M. Unterkalmsteiner, E. Engström, B. Regnell, G. Sabaliauskaite, A. Loconsole, T. Gorschek, and R. Feldt, "Challenges and practices in aligning requirements with verification and validation: a case study of six companies," *Empirical Software Engineering*, vol. 19, no. 6, pp. 1809–1855, 2014.

[7] E. J. Uusitalo, M. Komssi, M. Kauppinen, and A. M. Davis, "Linking requirements and testing in practice," in *16th IEEE International Requirements Engineering Conference (RE'08)*, pp. 265–270, 2008.

[8] M. Q. Patton, *Qualitative evaluation and research methods*. SAGE Publications, 1990.

[9] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[10] D. Leffingwell, *Agile software requirements: lean requirements practices for teams, programs, and the enterprise*. Addison-Wesley Professional, 2010.

[11] G. Melnik and F. Maurer, "Multiple perspectives on executable acceptance test-driven development," in *Agile Processes in Software Engineering and Extreme Programming*, pp. 245–249, 2007.

[12] J. Humble and D. Farley, *Continuous delivery: Reliable software releases through build, test, and deployment automation*. Pearson Education, 2010.

[13] G. Adzic, "How to get the most out of Given-When-Then," 2015. [Online]. Available: https://gojko.net/2015/02/25/how-to-get-the-most-out-of-given-when-then/, Last visit: 13.3.2016

[14] J. Itkonen, M. V. Mantyla, and C. Lassenius, "How do testers do it? An exploratory study on manual testing practices," in *Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement*. IEEE Computer Society, pp. 494–497, 2009.

[15] D. C. Gause and G. M. Weinberg, *Exploring requirements: quality before design*. Dorset House, New York, 1989.

[16] G. Adzic, *Bridging the communication gap: specification by example and agile acceptance testing*. Neuri, London, 2009.

[17] J. Larsson and M. Borg, "Revisiting the challenges in aligning RE and V&V: Experiences from the public sector," in *1st IEEE International Workshop on Requirements Engineering and Testing*, pp. 4–11, 2014.

[18] F. Ricca, M. Torchiano, M. Di Penta, M. Ceccato, and P. Tonella, "Using acceptance tests as a support for clarifying requirements: A series of experiments," *Information and Software Technology*, vol. 51, no. 2, pp. 270–283, 2009.

[19] E. Bjarnason, M. Unterkalmsteiner, E. Engström, and M. Borg, "An industrial case study on test cases as requirements," in *Agile Processes, in Software Engineering, and Extreme Programming*, pp. 27–39, Springer, 2015.

[20] T. C. Lethbridge, J. Singer, and A. Forward, "How software engineers use documentation: The state of the practice," *IEEE Software*, vol. 20, no. 6, pp. 35–39, 2003.

[21] E. Alégroth, R. Feldt, and P. Kolström, "Maintenance of automated test suites in industry: An empirical study on visual GUI testing," *Information and Software Technology*, vol. 73, pp. 66–80, 2016.

[22] B. Haugset and G. K. Hanssen, "Automated acceptance testing: A literature review and an industrial case study," in *IEEE AGILE'08. Conference*, pp. 27–38, 2008.

[23] J. Kukkanen, K. Väkeväinen, M. Kauppinen, and E. Uusitalo, "Applying a systematic approach to link requirements and testing: A case study," in *Proceedings of the 16th Asia-Pacific Software Engineering Conference*. IEEE Computer Society, 2009, pp. 482–488.

[24] J. Singer, "Practices of software maintenance," *Proceedings International Conference on Software Maintenance*, pp. 1–7, 1998.

[25] L. Mathiassen, O. K. Ngwenyama, and I. Aaen, "Managing change in software process improvement," *IEEE Software*, vol. 22, no. 6, pp. 84–91, 2005.

[26] N. Nurmuliani, D. Zowghi, S. Fowell, and P. O. B. Broadway, "Analysis of requirements volatility during software development life cycle," *IEEE Australian Software Engineering Conference*, pp. 1–10, 2004.

[27] L. Cao and B. Ramesh, "Agile requirements engineering practices: An empirical study," *IEEE Software*, vol. 25, no. 1, pp. 60–67, 2008.

[28] C. J. Stettina and W. Heijstek, "Necessary and neglected ? An empirical study of internal documentation in agile software development teams," *Proceedings of the 29th ACM International Conference on Design of Communication*, 2011.

[29] L. C. Briand, "Software documentation: how much is enough?" in *Seventh European Conference on Software Maintenance and Reengineering*. IEEE, 2003, pp. 13–15.