software

MDPI

Article

User Story Quality in Practice: A Case Study

Mohammad Amin Kuhail [1,*] and Soren Lauesen [2]

[1] Zayed University, P.O. Box 144534, Abu Dhabi, United Arab Emirates
[2] IT University of Copenhagen, 2300 København, Denmark; slauesen@itu.dk
* Correspondence: Mohammad.kuhail@zu.ac.ae; Tel.: +971-2-599-3536

Abstract: (1) Background: User stories are widely used in Agile development as requirements. However, few studies have assessed the quality of user stories in practice. (2) Methods: What is the quality of user stories in practice? To answer the research question, we conducted a case study. We used an analysis report from a real-life project where an organization wanted to improve its existing hotline system or acquire a new one. We invited IT practitioners to write requirements for the new system based on the analysis report, user stories, and whatever else they considered necessary. The practitioners could ask the authors questions as they would ask a customer in a real setting. We evaluated the practitioners' replies using these IEEE 830 quality criteria: completeness, correctness, verifiability, and traceability. (3) Results: The replies covered only 33% of the needs and wishes in the analysis report. Further, the replies largely missed other requirements needed in most projects, such as learnability and maintainability. Incorrect or restrictive solutions were often proposed by the practitioners. Most replies included user stories that were hard to verify, or would have caused a cumbersome user interface if implemented independently. (4) Conclusion: In this project, relying on the user stories would have been a disaster. Although the user stories could have been improved, they wouldn't cover the necessary requirements in the project.

This version of the paper has corrected several misprints and bad layouts in the official version.

Keywords: user stories; problem-oriented requirements; case study; hotline support

Citation: Kuhail, M.A.; Lauesen, S. User Story Quality in Practice: A Case Study. *Software* **2022**, *1*, x. https://doi.org/10.3390/xxxxx

Academic Editor(s):

Received: 6 May 2022
Accepted: 23 June 2022
Published: date

1. Introduction

User stories have their origins in extreme programming (XP). Kent Beck, the founder of XP, stated that user stories were created to address the specific needs of software development, conducted by small teams in the face of changing and vague requirements [1].

In general, user stories consist of brief descriptions of a system feature written from the perspective of the customer who wants the system [2]. Different user story templates and practices have been proposed by Beck [1], Jeffries, et al. [3], Beck and Fowler [4], and Cohn [2,5].

According to a survey in 2014, user stories have become the most used requirements in an agile environment [6]. In fact, user stories are integrated into many agile techniques, such as release and iteration planning and tracking the progress of a project. Thus, user stories are increasingly a means of communication with end-users and customers and the basis for developing the related functionalities and building them into the system [7].

Despite their popularity, there is little evidence of the effectiveness of user stories [8]. According to some authors, only simple, customer-visible functional requirements can be expressed by user stories [9], while the consistency and verifiability of user stories are difficult to validate [10].

This case study aims to answer this question: What is the quality of user stories as requirements in practice? We define the quality of requirements as the extent to which they meet the IEEE 830 requirements criteria [11]. To answer the question, we conducted a case study. We used a real-life project where an organization wanted to improve its existing hotline support system or acquire a new one. On his initiative, the second author interviewed the hotline staff and wrote a four-page analysis report in free-text style. It mentions 30 stakeholder needs and wishes. We invited eight IT practitioners, who had professional experience with user stories, to specify the requirements of the hotline system based on this analysis report. They could use user stories and whatever else they found necessary. We analyzed the practitioners' replies and assessed the extent to which they met the IEEE 830 requirements criteria [11]: complete, correct, verifiable, and traceable.

Our key findings were as follows: seven out of the eight replies followed the Connextra template [8]: As X, I want to Y in order to Z. On average, the replies missed around two-thirds of the 30 stakeholder issues (needs and wishes). Also, more than two-thirds of the serious (business-related) issues were ignored, while some restrictive solutions that the text did not call for were added. The user stories were hard to verify and largely did not cover requirements needed in most projects but not explicitly mentioned in the analysis report, such as usability, data migration, phasing-out, security, and performance.

The remainder of this article is structured as follows. First, a review of the related work is discussed in Section 2. Section 3 describes the hotline project, as well as the profiles of the eight IT practitioners who participated in the study. Section 4 explains the design of the study. Section 5 reports the key findings, and Section 6 discusses the results.

2. Related Work

The related work can be divided into user stories as requirements and investigation of requirements in practice.

2.1. User Stories as Requirements

Many organizations prefer to use user stories as requirements, rather than traditional requirements [9]. User stories cover the essential parts of a requirement: who it is for, what is needed from the system, and, optionally, why it is important [12]. Further, user stories serve as conversation starters with customers, and they may change before, or during, implementation [2].

Despite their popularity, there are many concerns with user stories. For instance, they are often imprecise, require significant implementation effort [13], and are unsuited for expressing requirements for large or complex software systems. In such a case, separate system and subsystem requirements are needed [13,14].

Another major concern with user stories is the lack of requirements traceability, creating problems when requirements, code, and tests change over time [15,16].

Non-functional requirements (NFRs) are often ill-defined or ignored by user stories, as stakeholders focus on core functionality and ignore scalability, maintainability, security, portability, and performance [9,10].

The requirements side of the engineering community has made several attempts to address some of these concerns and improve the quality of user stories [12]. For instance, some authors [17] suggested guidelines for documenting NFRs, where the diversity, scope, and detail level of the NFRs are taken into account. Other authors [18] suggested using tools for visually modeling NFRs to help to reason out NFRs.

Some approaches to improving user stories utilize qualitative metrics, such as the heuristics of the INVEST framework (Independent-Negotiable-Valuable-Estimable-Scalable-Testable) [19], and the general guidelines for ensuring quality in agile requirements engineering, proposed by Heck and Zaidman [20]. Lucassen et al. [12] proposed the Quality User Story Framework, containing syntactic, semantic, and

pragmatic criteria that user stories should conform to. The proposed frameworks do not advocate IEEE criteria, such as verifiability, traceability, and modifiability, but commend other criteria that overlap with the IEEE criteria, such as completeness and unambiguity.

A notable contribution to enhancing the quality of user stories in practice is a unified model that covers the essential elements of user stories: who it is for, why it is important, and what it needs [21], as well as visually representing user stories using Rational Trees (RTs) [22]. RTs help analysts view the interdependencies of user stories and could potentially help them identify inconsistencies or missing requirements. However, despite their benefits, it can be challenging for analysts to build RTs [23]. Further, it was found that using a visual representation (Rational Tree) did not help analysts determine whether some user stories were missing [24].

2.2. Requirements in Practice

Fernandez et al. [25] conducted a mapping study on empirical evaluation of software requirement specification techniques and found that most authors conducted experiments in academic environments. A recent example of such experiments can be found in [26]. The authors performed a controlled experiment with 118 undergraduate students to assess the benefits of user stories versus use cases as part of a course. The authors concluded that participants could derive a more complete conceptual model with user stories because of the conciseness and focus of user stories and the repetitions of entities in the user stories.

A few articles examined how requirement engineering methods are used in practice. For instance, a recent study [27] assessed the quality of derived conceptual models from two notations: user stories and use cases. The study found that the requirement notation has little effect on the quality of the derived conceptual models. Another empirical study explores how practitioners use user stories and perceive their effectiveness (defined as the extent to which user stories improve productivity and quality of work deliverables). Lucassen et al. conducted a survey with 182 practitioners and 21 follow-up interviews. The results show that practitioners agree that using a user story template and quality guidelines, such as INVEST, improve the quality of the user story requirements [8]. Related studies found in [8,26] assessed the perceived benefits and effectiveness of user stories from the perspective of students and practitioners. Since perceived effectiveness is subjective, there is a need to assess the effectiveness of user stories in a more objective way. Another study [6] found that most practitioners do not use requirements standards and prefer to use their own personal style.

The aforementioned studies undoubtedly contributed to the body of knowledge. Nonetheless, none of these studies evaluated the quality of the requirements of a real-life project expressed with user stories. A recent systematic literature review suggests that agile requirements engineering as a research context needs more empirical investigations to better understand the impact of agile requirements engineering in practice [28].

An attempt to evaluate the quality of non-agile requirements was embodied in an experiment comparing use cases with task descriptions by asking 15 professionals to specify requirements for a hotline system with use cases or task descriptions [29]. The result of the experiment was that traditional use cases covered stakeholder needs poorly in areas where improvement was important but difficult, and restricted the solution space severely.

We could not find studies that assessed the quality of user stories in practice with objective requirements criteria. This case study attempts to fill the gap by assessing the quality of user stories according to the IEEE 830 criteria [11]: complete, correct, verifiable, and traceable.

3. The Study

Our main objective was to assess the quality of user stories as requirements in a real-life project. As mentioned above, we defined quality as how well the user stories met four IEEE 830 criteria [11]: complete, correct, verifiable, and traceable. While these criteria were not written with user stories in mind, we argue that they are valid for user stories too. For instance, completeness and correctness are also quality criteria specified in the Agile Requirements Verification Framework [20]. Further, completeness is a criterion specified by the Quality User Story Framework [12], while verifiability is a criterion in the INVEST framework [19], known as testability. We included traceability as it is often needed in real-world projects [30], such as the one discussed in our study.

As a qualitative study, our objective was not to establish a cause-effect relationship but to shed light on a largely unexplored area, the quality of user stories in practice.

Years before the case study, the second author had, on his own initiative, studied a real project, the acquisition of a hotline support system, where he could interview stakeholders, have them comment on requirements and other documents, and observe the system in use.

He had written a four-page analysis report about the situation. The case-study idea was to let professional analysts write requirements based on this analysis report. We gave the report a short introduction and sent it to professional analysts. It can be found in [31] and is shown in table format in Table 3.

3.1. The Hotline System

The hotline (help desk) system was a real-life project. The customer was a thousand-user department of the Danish government. They had an IT department that, among other things, operated a hotline. Hotline staff wanted to improve their existing open-source hotline system or acquire a new one.

In summary, the hotline staff received requests from IT users. First, first-line supporters handled the request. In 80% of the cases, they could solve the problem and close the case. The remaining requests were passed on to second-line supporters, who could solve the problem themselves, ask for more information, transfer it to an expert, order parts from a vendor, or put the request on hold. The hotline supporters might change roles or attend to other work duties.

The first part of the analysis report sent to the professionals is shown in Box 1.

Box 1.

Hotline support system

[A1] The Z-Department is a department of the Danish government. It has around 1000 IT users. It has its own hotline (help desk). They are unhappy with their present open-source system for hotline support and want to get a better one. They do not know whether to modify the system they have or buy a new one in a tender process.

An analyst has interviewed the stakeholders and observed what actually goes on. You find his report below. Based on this, your job is to specify the requirements for the new system, using your favorite user-stories approach and what you otherwise find needed.

Please send your requirements to [Author1] or [Author 2]. They will compare the replies and give you feedback. Feel free to ask for more information.

IT users

[A2] The users encounter problems of many kinds. For instance, they may have forgotten their password, so they cannot start their work; or the printer lacks toner, or they cannot remember how to make Word write in two columns. The problem may also be to repair something, for instance, a printer, or to order a program the user needs.

[A3] The easiest solution is to phone the hotline or walk into their office. In many cases, this solves the problem right away. However, the hotline prefers to receive the problem request by e-mail to hotline@ ... Sometimes, this is impossible, for instance, if the problem is that the user has forgotten his password.

[A4] …

This text contains three stakeholder issues, shown as paragraphs A1, A2, and A3 above. For ease of comparing the replies, we have shown all the 30 stakeholder issues as a table (Table 3). The analysis report that the participants received, did not have the identifiers A1, A2, etc.

The issues A1–A30 in the report were in a form that stakeholders could read and revise, but they were not sufficient as requirements.

3.2. Analyst Recruitment and Profiles

We invited IT practitioners (analysts) to write requirement specifications based on the hotline analysis report. The invitation mentioned that user stories are widely praised in industry and academia, but there is no agreement on how to use them and how they cover requirements. Analysts could use user stories and what they otherwise found necessary. They could ask questions for more information, but nobody did.

To attract professionals, we announced the case study as an experiment. We announced it in June 2020 on our personal websites and professional platforms, such as LinkedIn and personal contacts. We received several comments saying: *This is an interesting idea, but I do not have the time to reply*. We also offered financial compensation to some of the professionals, half of which accepted, and the other volunteered to do it as a challenge.

We received eight replies. The full replies are available in [31]. The profiles of the analysts are shown in Table 1. We have included reply S, which is the actual requirements for the hotline. It was made by the second author, based on the analysis report, years before the case study. It does not use user stories but problem-oriented requirements, an approach where you do not specify what the system shall do but what it will be used for.

Table 1. Analyst Profiles.

Analyst	Position/Country	Education	Experience
A	A team of two consultants (Denmark).	MS in computer science.	The team has extensive industry experience and has taught their own version of user stories for 7 years.
B	A researcher (Sweden).	Ph.D. in computer science.	Modest industry experience.
C	A software engineer (USA).	BS in computer science.	Industrial experience in a corporation for 4 years. She used user stories extensively in her work as part of specifying requirements.
D	A software engineering consultant (USA).	MS in computer science.	More than 4 years of professional experience in a corporation. He has written user stories using Jira Software as part of his work.
E	A business analyst (UAE).	BS in Management Information Systems (MIS).	She has experience in managing and analyzing projects for more than a year. She writes user story requirements as a daily part of her job.
F	A software engineer (USA).	BS in Computer Science.	He has industry experience at various corporations for more than 3 years. He has used user stories as part of his job.
G	A software engineer (USA).	BS in Computer Science.	He has industry experience at a corporation for more than 2 years. He has working experience with user

			stories as his employer uses them to define requirements.
H	A researcher (USA).	MS in Computer Science.	He has industry experience at various US-based companies working with data science, quality assurance, and software development.
S	Soren Lauesen	MS in math and physics.	This was not a reply, but the second author's problem-oriented requirements for the hotline project.

4. Evaluation

We evaluated the analysts' replies according to these criteria from the IEEE 830 recommended practice for software requirements specification.

Completeness: Are all stakeholder issues (needs and wishes) specified in the requirements?

Correctness: Do the requirements represent stakeholder issues? Some requirements may be incorrect because they restrict the solution space by enforcing an inconvenient solution. Requirements can also be incorrect because they specify something that the customer does not need.

Verifiability: Is there a cost-effective method for checking that the requirements are met?

Traceability: Can we trace business issues to requirements and test scripts to requirements?

Procedure

We received ethical clearance from the research ethics committee at Zayed University to conduct this study. Figure 1 shows the stages of the study together with the used and resulting artifacts. The research procedure was aimed at achieving standard-based assessment of the quality of user stories, while minimizing subjectivity.

We gave the analysts the four-page analysis report explained in Section 3.1. The analysts were invited to ask questions, but none of them did.

We received 8 replies (replies A–H). After receiving the replies, we noticed that they covered the stakeholder issues badly. So, we asked the analysts whether they thought their replies covered all the issues mentioned in the analysis report. To our surprise, they all said "Yes."

The two authors assessed all replies independently, based on an agreed assessment protocol illustrated below. Each author spent two to three hours on each reply. Further, for each reply, we had agreed on making a tracking version of the analysis report, in which we indicated for each of the 30 issues where the reply dealt with it—if at all. Further, we made an annotated copy of the reply, in which we indicated for each user story or requirement, which issue or issues it dealt with, and whether it was a possible solution, a partial solution, a wrong solution, or that it did not cover the issue. The replies and our assessments are available in [31].

We compared our independent assessments and settled all disagreements by scrutinizing the reply.

We sent our joint evaluation to the analysts, asking them for comments. Some authors pointed out a few mistakes in our evaluation, for instance, that we had missed requirements in their reply that dealt with some issue. We had follow-up discussions with them and easily agreed on these points.

Assessment Protocol: The authors had an agreement on how to assess completeness, correctness, verifiability, and traceability. We used this rating for completeness: 1 if the issue was well covered, ½ if it was mentioned or had a partial solution, and 0 otherwise. Issue A11 (Table 3) provides an example: *There are 10–15 employees who occasionally or full time serve as supporters. They know each other and know who is an expert in what. The supporters frequently change between first and second line, for instance, to get variation. It happens,*

unfortunately, that a supporter moves to second line without realizing that nobody remains in first line.

Reply A (User Story 3) handled it with this user story: *As a supporter, I wish to be able to change between working in first- and second-line support to ensure variation in my work. System warns if the last first-line supporter attempts to switch group*.

In this example, the analyst wrote a plausible solution (the system warning) to the problem mentioned in the assignment (nobody realizes that no one remains in the first line). While there may be other equally valid solutions, we gave the analyst a rating of 1 (the issue was well covered).

Here is another example:

Issue A9: *Half of the second-line requests are in principle easy, but cannot be dealt with immediately. The supporter may have to move out of the office, for instance, to change the toner in the printer or help the user at his own PC. Usually, this ends the request, but it may also turn into a long request because a specialist or a spare part is needed. Often the supporter visits several locations when he moves out of the office*.

Reply C (Lines 90, 91) handled it in this way: *As a Supporter, I want to be able to mark the request as Closed from a mobile device, so I do not need to remember to update the request if I complete the request away from the office*.

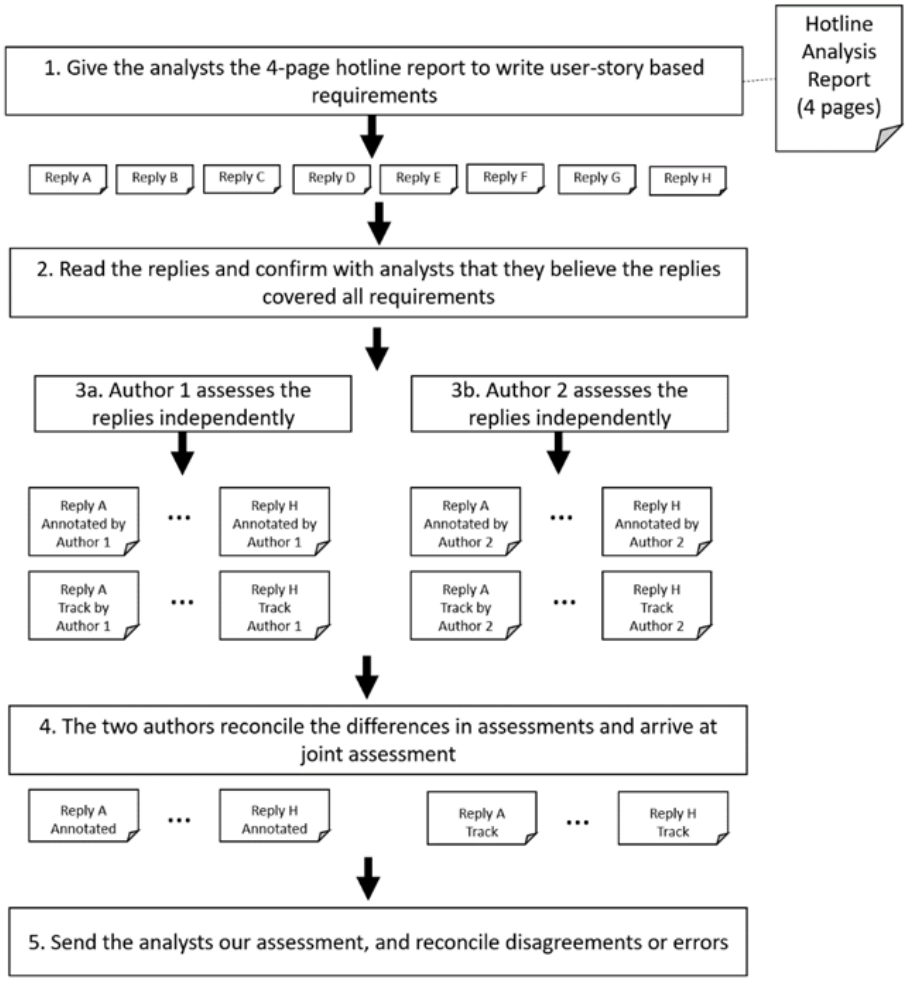

Figure 1. The stages of the case study.

Our assessment is that this solution partially deals with the issue of being able to close requests away from the office, but leaves out other details, including pending requests due to external issues. Accordingly, we gave this story a rating of ½. (We consider this a mild assessment).

The procedure described above also assessed correctness (step 3). The two authors checked independently whether the requirements expressed with user stories reflected a need. Thereafter, the authors discussed their assessment and came to an agreement.

To assess verifiability, the two authors checked independently whether there was a cost-effective way to check that the user stories were met. Thereafter, the authors discussed their assessment and came to an agreement. For instance, the authors assessed if test cases could be created to verify that the user stories were fulfilled.

To assess traceability, the two authors checked independently whether each user story had some IDs that allowed references to the story from business goals or test scripts.

5. Findings

One reply (A) specified acceptance criteria for the user stories. All replies, except reply D, followed the Connextra user story template in which a user story follows this pattern: "as X, I want to Y, in order to Z", or a similar pattern "as X, I want to Y, so that Z". The style of requirements in reply D resembled the traditional shall requirements, so we asked the analyst if he thought he deviated from the common user story style. He explained that he wrote the requirements in this style because he used a software tool named Jira [32], where he could write requirements in the 'shall' style, then let Jira convert them to user stories. We treated this reply as if each shall requirement was part of a user story.

Most of the replies organized the user stories into groups (sometimes called epics). However, there was no agreement on how to do it.

Reply A had 8 epics, called "backlog item#1#", "backlog item #2#", etc. Each of them had one user story (As a … I want …) with several acceptance criteria, each corresponding to a system functionality (roughly: the system shall …).

Reply B had 12 high-level user stories (As a …), each comprising several sub-user stories (As a …).

Reply C had around 28 user stories grouped into 3 user groups (IT user, supporter, and manager). The term "epic" was not mentioned.

Reply D had 15 system-shall requirements, which we decided to treat as the "I-want-to" part of 15 user stories.

Reply E had 77 user stories, organized in a group for each user type (IT-user, supporter, …).

Reply F had 5 user stories, all starting with "As a supporter." Further, there were two epics (New system and Statistical Analysis) with around 5 user stories.

Reply G had 5 epics, e.g., Notifications and Features. Each epic had around 4 stories.

Reply H had 2 epics (From the user's and the supporter's perspectives). Each had around 30 user stories organized in two levels.

5.1. Completeness

Requirements were considered complete when they covered all stakeholder issues. Table 3 shows how replies A-H covered the 30 stakeholder issues, A1 to A30. A "1" in the table indicates that the issue was well covered, and a ½ indicates that it was partly covered, e.g., because it mentioned a (partial) solution, but not the problem to be solved. Otherwise, it was rated "0".

Column S shows the coverage of the actual hotline requirements specification, written by the second author. It did not use user stories, but problem-oriented requirements. This approach described what the system was to be used for, and the problems to be resolved. It was up to the suppliers to show how their solutions supported

the tasks and handled the problems. We compare the problem-oriented approach with the user story approach in more detail in Section 6.4.

On average, the user story replies covered 33% of the stakeholder issues, with reply B having the highest coverage (13 out of 30) and reply F having the lowest coverage (5 out of 30). The problem-oriented reply (S) covered 29 of the 30 stakeholder issues. It could be argued that the coverage of requirements was poor due to the lack of experience with user stories. However, the coverage was low for all the analysts, regardless of their experience. To our surprise, the analyst with the best requirement coverage was a researcher with little professional experience (Analyst B) with 43.3% coverage.

Some stakeholder issues completely disappeared from the user story replies. For instance, all replies missed the issue in A1: The organization does not know whether to modify their existing system or buy a new one. Reply S dealt with it by not specifying what the system should do, but what it would be used for. This allowed the analyst to mention an issue without specifying a solution. The actual solution might differ from one supplier to another.

Most analysts considered only stakeholder issues mentioned in the analysis report, and largely missed other types of requirements needed in most software acquisitions, for instance, usability, response time, system maintenance, and phasing-out (e.g., transferring your data to another product). This confirmed findings from previous studies such as [9] and [10].

Conversation starters: User stories are negotiable and meant to start a conversation between stakeholders and developers [2]. As such, user stories are not complete until the conversation is made. The analysts were invited to ask questions to the authors. While the authors themselves were not the customers, they knew the customers and could convey the questions to them. Nonetheless, none of the analysts asked any questions.

It was also dubious that the conversation would bring up the many issues not mentioned in any user story.

5.2. Correctness

Requirements are correct if they reflect a customer need. We observed several incorrect requirements that were either too restrictive solutions or simply wrong requirements that the stakeholders did not want. As an example of a wrong requirement, analyst F wrote the following solution to issue A4 (Table 3):

A4: *User: . . . When can I expect a reply? . . . The issue has been solved, but the user does not know*.
Reply: *As the requester, I should be able to see what the expected completion time of my request is, based on prior statistics of a similar problem* (Reply F, Lines 56, 57)

While it may be useful to analyze statistics of similar requests, completion time also depends on other factors, such as how busy the hotline team is currently. So, the solution was wrong. It also excluded a simple solution: The system sends a mail to the requester when the supporter closes the request.

Even analyst C, who received 1 on completing issue A4, specified this restrictive solution to the problem:

As an IT User, I want to receive an e-mail of approximate time to complete my request, so I know what to expect (Reply C, Lines 18, 19).

While the specified solution was reasonable, it remained restrictive. Perhaps some users prefer a different method for informing them of the approximate time, such as an SMS or a phone notification. Further, an otherwise good hotline system may not estimate completion time but send an e-mail when a request is closed. Such a system might be discarded for not supporting the user story.

As another example of dubious requirements, analyst E wrote this as a solution to issue A24:

A24: . . . Supporter: . . . *difficult to specify a cause initially. Cannot be changed later.*

Management: . . . *No list of potential causes, and hard to make one*.
Reply: *As a line 1 supporter, I want to be able to request an additional cause to be added to the root causes list without delaying the ticket closure so that the list is updated properly* (Reply E, Lines 75, 76).

This may sound plausible, but experience shows that it causes a wildly growing list of causes. New causes must somehow be managed, and a system supplier may know it and provide a better solution.

Here is an example of a restrictive requirement in the reply written by analyst B to issue A4:

A4: User: . . . *When can I expect a reply? The issue has been solved, but the user does not know.*
Reply: *As an IT user, I want to be able to look up the status of a request I have made, so I can track its progress in detail and plan my work* (Reply B, 3c).

While this may be a possible solution to issue A4, it is an inconvenient one, according to the analysis report (A5). Similarly, analyst H mentioned this solution to issue A11:

A11: Supporter: . . . *nobody left on 1st line*.
Reply: *As a supporter, I should be notified if my team members move from first line to second line or vice versa so that I know the current team lists.* (Reply H, d-ix).

If such a solution is implemented, it may be very inconvenient, especially if supporter transfers are frequent.

5.3. Verifiability

According to IEEE 830 and the INVEST quality criteria, a well-written user story should be testable [19], or in other words, verifiable. However, our observation was the contrary for most replies. Particularly, it was hard to verify the "in-order-to" or "so-that" part of many written user stories. For instance, analyst B wrote this user story as a solution to issue A25:

A25: Supporter: . . . *Today it is hard to record additional comments*.
Reply: *As a hotline supporter, I want to be able to add additional information to any request, so that I can help the request be resolved sooner.* (Reply B, 9f).

Assuming the "I-want-to" part is sound, the "so-that" part is hard to verify. For instance, how "sooner" will the request be resolved with additional information? Also, should we compare with a request without information to verify that the one with additional information is resolved sooner?

We also observed analysts writing unverifiable solutions to problems expressed in the list of issues. For example, analyst E wrote the following solution to issue A14:

A14: Supporter: . . . *hard to spot important requests among other requests*.
Reply: *As a line 1 supporter, I want to sort all open issues by different methods so that I can easily locate an issue I want to work on.* (Reply E, Lines 84, 85).

Not only is the solution vague, but also hard to verify. Which methods would be used to sort requests? It could be argued that the solution's vagueness was due to natural language or lack of experience. However, in this case, issue A14 was itself somewhat vague (for instance, the meaning of an important request was not defined). As such, the solution the analyst came up with was vague too. In our experience, it is often hard to write a clear and verifiable solution until the problem has been analyzed further as part of the solution design.

As with other examples, the "so-that" part was hard to verify. How can we verify that the supporter can easily locate a request? We lack information about the problem—what the supporter is trying to achieve.

As stated previously, analysts largely ignored non-functional requirements, such as ease of learning and response time. As an exception, analyst B wrote the following user story about usability:

Reply: *As a hotline supporter, I want a support system interface that is incredibly usable, so that I do not waste my time, get frustrated, or avoid entering data.* (Reply B, 10).

This is a nice user story, but the "want-part" and the "so-that" part are unverifiable.

5.4. Traceability

In theory, a requirement is traceable if it can be tracked from its origin (business goals), through its specification and development, to its deployment (Gotel et al. [33]). When you trace something to a requirement, you write a reference to the requirement in some document, e.g., in a list of business goals or a test script. However, 3 out of 8 replies had no IDs (replies B, C, and F). The rest of the replies had IDs, but in different ways. Some replies used serial numbers only (i.e., 1, 2, 3), others used two-level numbering (i.e., 1a, 1b, 1c) where the number is the epic number, and the letter is the user story.

Tracing from business goals to requirements is important to ensure that the goals are met. However, none of the replies could trace business goals to user stories. In fact, none of the replies mentioned any business goals. We did not ask the analysts to deal with business goals, but a careful analyst should have done it—or asked the "customer" (the authors).

Tracing between test cases and requirements is important too. Otherwise, you cannot know whether the requirement has been tested. Here too, you need requirement IDs.

The issue of not assigning IDs to requirements may not be unique to user stories, but the lack of a numbering scheme leads to untraceable requirements, as was the case here.

6. Discussion

This section expands upon the findings to shed light on aspects of user story quality in practice. First, we discuss how the original idea of user stories is misapplied in practice, causing incomplete or incorrect requirements. Second, we discuss the context in which user stories are developed, and how this may have an impact on requirement quality. We also illustrate that a quality principle of user stories (atomicity) could result in a cumbersome user interface. Finally, we show that it is possible to cover the requirements with an alternative technique (problem-based requirements).

6.1. How a Good Idea Is Corrupted

The basic idea in user stories is good: We want to know who the users are, what users want, and why. This is covered nicely with the pattern *as X, I want to Y, in order to Z*.

The problem appears when the analyst believes that all requirements should be expressed as user stories. Many issues are not something that some user wants. So, what does the analyst do with these issues? In the replies, we see three approaches:

1. The analyst invents a user. As an example, look at issue A11 (Table 3): *It happens that a supporter moves to second line without realizing that nobody is left on first-line*. Analyst B dealt with this by inventing a manager (Reply B, 12a): *As a manager, I want the support system to always have a person available for the users, so that the users are not angry*. Actually, this small hotline does not have a manager. The hotline is part of the IT department, which has many other duties. Supporters find out themselves who does what, so the requirement was wrong. The customer did not want it. As another example, Analyst F made the "system" a user (Reply F, Line 25): *As the new system, I should poll all devices.*
2. The analyst invents a solution and claims that a user wants it. As an example, reply C (Lines 29, 30) said: *As an IT User, I want a reference number for my request, so that if I need to call support again, they will be able to easily look up my request*. This user story was not related to a specific issue. Further, in our experience, IT users hate such numbers. They much prefer that hotline staff identify the request by the IT-user's name. So, the

requirement was wrong. It was a bad solution. The reference number was convenient for other reasons, but not because the user wanted it.

3. The analyst ignores the issue. This happened often, and as a result, we saw a low hit rate: On average, each reply covered only 33% of the issues.

The "in-order-to" clause should relate the "want" to a higher purpose or to the problem to solve. However, it usually ended up as a trivial statement. Here are some examples:

1. *As a user I wish to report an IT-request to service desk, in order to get it solved as soon as possible so I can continue my work* (Reply A, User Story 1, in relation to issue A3 in Table 3).
2. *As an IT user, I want to report printer problems so that I can print* (reply B, User Story 1b, in relation to issue A2 in Table 3).
3. *As a supporter, I want a simple template to create requests, so it is fast and easy to create simple requests* (reply C, Lines 36,37, in relation to issue A7 in Table 3).
4. *As a Supporter, I want to be able to view requests from a mobile device, so if I need information away from the office it is available* (reply C, Lines 93, 94, concerning issue A18 in Table 3).
5. *As a Line supporter, I want to view all open issues so that I can action them* (reply E, Line 20, in relation to issues A15 and A19)

A higher purpose or a business problem often needs support from several requirements. For example, issue A14 (Table 3) states, *hard to spot important requests among other requests* is a business problem. Whether a request is important and needs attention now, depends on much data, such as request priorities, other requests in the pipeline, deadlines, experience data on time to deal with this kind of issue, and availability of experts. This does not fit into the atomic user story pattern. At best, it needs several atomic user stories.

6.2. Development Contexts

There are several types of IT projects, for instance: A system to be developed in-house, a product to be developed for a market, buying a COTS (commercial off-the-shelf) system (possibly with some additions), a maintenance project (modifying a system already owned).

In the hotline case, stakeholders did not know whether to improve the system they had (a maintenance project) or buy an existing one (a COTS project). This is a common situation, and requirements should deal with it.

In principle, user stories should be suited for small maintenance projects because they specify a few issues some stakeholders have. According to the analysis report, the hotline case could become a maintenance project. However, in the replies, most of the stakeholder issues were not covered by the user stories, as explained in Section 5.1 (completeness).

The alternative, according to the analysis report, is a COTS system. Could the user stories be used here? No, most of them expressed issues as solutions, and the COTS system might provide other solutions, thus not meeting the written requirements.

Apparently, the analysts were not concerned about the development context and assumed that it was a development from scratch. If we were going to develop a hotline system for a market or our own company, the user stories would be the same. They would specify bad solutions and cover few of the needs.

6.3. Implementing a User Story

A well-written user story should be independent of other user stories [19], allowing them to be implemented independently [34]. How does this principle influence the implementation of user stories, particularly the user interface design? Reply A was written

by a team of two consultants. We implemented one of their user stories: *As a supporter I wish to assign a request to the right supporter or support group in order to get someone qualified, working on finding a solution.* We wanted to implement it in a context-independent way, aware that the solution would be cumbersome to the supporter.

Our suggestion was a two-step dialog (Figure 2). First, the supporter selects what to do, in this case "Assign supporter." This brings up the dialog box. Next, the supporter uses the "Request ID" dropdown to search for the request to be assigned. In this case, request 6241. Next, the supporter uses the "Assign-to-person" dropdown to select the proper supporter. Finally, the supporter closes the dialog box by clicking OK.

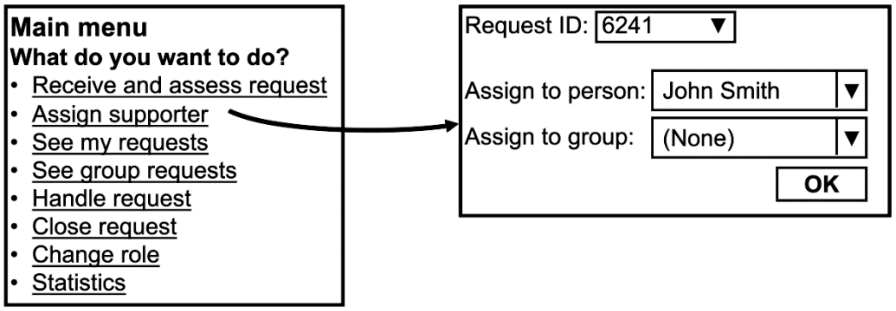

Figure 2. A cumbersome user interface derived from analyst A's user stories.

We presented the solution to the analysts, and they found it great. They said they would do it much the same way. Such a user interface supports other user stories in a similar dialog-based fashion. The interface is based on selecting the action, then the object to act on. However, as demonstrated, this leads to many steps and windows.

The existing system is different. Figure 3 shows the main screen. It shows the list of requests that the supporter works on at present. To assign a request to someone else, the supporter right-clicks the request on the list and selects *Assign Supporter*. This shows a list of supporters. The supporter clicks the new supporter on the list. The system closes the lists and updates the list of requests to show the new "owner" of the request. That is all. There is no search, only clicking twice.

The supporter will often carry out several actions on the request (A19 in Table 3), but we cannot see this in the atomic user stories. This is because a user story does not reflect a real work context. The real work context is this task: Look at the list of requests, find one to handle, look at the details and do whatever can be done to it now.

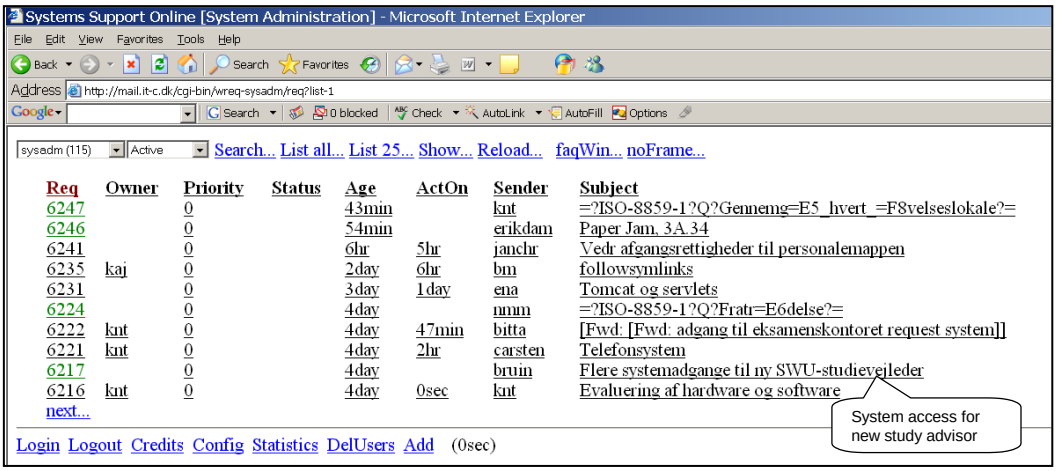

Figure 3. A list of hotline requests in the existing support system.

The fact that analysts strive for independent user stories makes it hard to understand the context in which the user stories are done. Lack of context causes developers to misunderstand user needs [35], potentially resulting in a cumbersome user interface.

6.4. Comparison with Problem-Oriented Requirements

Reply S was written by the second author for the actual hotline project—years before the case study. It uses this approach:

Problem-oriented requirements: Do not specify what the system shall do, but what it will be used for and which problems it will eliminate. It uses two-column tables with the needs/problems in column 1. Column 2 may initially be empty. Later, it shows a potential or promised solution. See the example below and Lauesen [36].

As Table 3 shows, reply S covers 29 out of the 30 stakeholder issues. It achieves this with problem-oriented requirements.

A user task is an example of problem-oriented requirements. It describes what a user can do with the system from a trigger (e.g., a supporter receives a request) until the user cannot do more with the system right now (e.g., because the supporter has handled the request and replied to the user, or passed the request on to second-line).

Table 2 shows a simple but important task in the hotline: A supporter changes his role. The top part of the table describes the user and when he carries out the task. It corresponds roughly to *As a …*. Column 1 shows what user and system try to do together—or a problem to be eliminated. On purpose, we do not specify who does what. A good system does much of it. Column 1 corresponds roughly to *In order to*. Column 2 shows solutions, but may be blank—waiting for the supplier to suggest a solution, maybe in dialog with a stakeholder. It corresponds roughly to *I want*.

Table 2. A problem-oriented task.

C13. Change role
Users: Any supporter.
Start: A supporter has to change line, leave the hotline, etc.
End: New role recorded.
Frequency: Around 6 times a day for each supporter.
All subtasks are optional and may be performed in any sequence

Subtasks and variants:	**Example solutions:**
1. Change own settings for line, absence, etc. (see data requirements in section D3).	
2. Change other supporter's settings, for instance if they are ill.	
3. When leaving for a longer period, transfer any taken requests to 1st or 2nd line.	
3p. Problem: The supporter forgets to transfer them.	The system warns and suggests changing status for all of them.
4. When leaving 1st line, check that enough 1st line supporters are left.	
4p. Often forgotten.	The system warns.

A task corresponds to several user stories: one for each action the user/system can do during the task. The actions can preferably be done in any sequence.

Supporters can carry out three different tasks: Handle a request (10 possible actions), Handle a message from an external supplier (2 actions) and Change role (Table 2, 4 actions and two problems to eliminate). End-users and managers have tasks too.

The task descriptions are 4 pages in reply S. Typical user-story replies are also 4 pages. However, the entire reply S is 32 pages containing around 50 tables arranged in 12

chapters. As an example, Chapter D handles data requirements with 6 tables (one for each class in the data model). Chapter H handles security with 6 tables, Chapter L handles response time, maintenance, etc. with 5 tables.

Problem-oriented requirements have been used successfully in more than 100 real-life projects, starting in 2007 where the approach was named SL-07. Based on experience with these projects, the chapter structure has gradually been improved. It is no wonder that it covers 97% of the stakeholder issues in the hotline case, while user stories cover only 33%.

7. Threats to Validity

We identified the following threats to the validity of this study:

1. **Participation Selection:** Since it is hard to recruit qualified analysts who will commit several hours for such a study, the sample of analysts was small and may not have represented the entire population of IT practitioners. To mitigate the threat, the two authors recruited the analysts from their networks independently. Further, we only recruited analysts that had reasonable professional or academic experience with user stories. As a result, there was some diversity in the sample in terms of the analysts' education, location, and experience. It could be argued that some analysts did not have sufficient experience to participate in this study. However, according to Table 1, six analysts had used user stories professionally as part of their daily work, and only two had theoretical knowledge of user stories (analysts B and H). However, when we compared the quality of their replies with that of the analysts who had professional experience with user stories, there was no significant difference. Nonetheless, with such a small sample, the authors do not claim that the results of this study are generalizable. The purpose of the study was not to obtain generalizable, statistically significant results, but to explore new territory: the quality of user stories as requirements in practice.

Participant Incentive: Half of the analysts (A, B, D, F) were not financially compensated for their involvement in the case study. The participants simply volunteered to do it as a challenge. It could be argued that these analysts did not have the incentive to write quality user stories. Nevertheless, when we compared the quality of their replies with that of the analysts who were paid (C, E, G, H), there was no sign of the paid analysts performing better. In fact, analyst B scored the highest in terms of completeness (43%), followed by analyst D (37%). None of them were financially compensated.

Case Study Setup: It could be argued that user stories are a natural language tool used for brainstorming. As such, it is more natural to let analysts talk to the client and develop the user stories gradually as opposed to basing the user stories on a pre-made report. Such a setting would be more realistic but impossible to do with many participants. We mitigated the issue by inviting the analysts to ask the authors questions, as they would ask a customer in a realistic setting, but nobody did. We also asked them whether they were sure they covered everything, and they all said yes. Further, the analysts received our evaluation for comments. There were few comments, and they were easily reconciled.

Author Bias: One of the authors has used an alternative technique to write requirements, problem-oriented requirements [36]. This might cause him to be biased against user stories. We mitigated this threat by getting consensus on the evaluation from both authors and the analysts, as explained above.

Construct Validity: The quality of user stories is not something that can be directly observed or measured. To avoid subjective measurement of effectiveness, we relied on the IEEE 830 requirements criteria to measure quality. Further, as discussed in Section 4.1, we implemented several checks and balances to maintain objectivity. First, the authors assessed the criteria independently, reconciled the differences, and

arrived at consensus. Second, in assessing the criteria, we intentionally ensured we did not make a harsh judgment. For instance, as explained in Section 4.1, we considered a case fully covered even if it imposed an inconvenient solution. We considered a case partly covered even if it just mentioned the issue. Second, we invited the analysts to challenge our assessment and measurement. We exchanged a few e-mails in cases where there were doubts about our assessment. We easily agreed. Third, we invited experts and researchers to challenge our assessment by making our assessment publicly available here [31].

8. Conclusions

User stories are easy to write and widely used. However, the result of this study was that the eight analysts squeezed all stakeholder issues into the pattern: *As X, I want Y*, In order to Z. What did not fit the pattern was ignored. The analysts often filled in X, Y and Z erroneously, leading to incorrect or unverifiable requirements. Further, there did not seem to be a systematic way to group user stories into epics. Each analyst did it differently.

This study has shown that user stories in a small, real-world project covered only 33% of the issues mentioned in the customer's analysis report. Further, the user stories largely missed other kinds of requirements needed in most software projects, for instance, ease of learning, response time, system maintenance, and phasing-out (e.g., transferring your data to another product). In the real project behind the case, these kinds of requirements were 27 pages in length, while the parts corresponding to user stories were 4 pages.

The user stories often prescribed wrong or restrictive solutions. Most user stories were hard to verify, especially the *In-order-To* parts. Further, many user stories were not traceable, as they lacked IDs.

If implemented independently, the user stories could cause a cumbersome user interface. Moreover, the user stories did not support acquisition projects, as clearly was the situation in this case. Analysts assumed that the user stories were for a system to be built from scratch.

Even though user stories are supposed to be conversation starters, it is dubious that conversations with stakeholders about the user stories would bring up the many missing requirements.

An important implication for practitioners is that user stories should specify the problems users want to avoid, rather than the solution the practitioner thinks the user wants. Not doing so may result in a restrictive solution, as this case study showed. Further, user stories are not sufficient to describe all types of requirements. Other methods should be used to specify other types of requirements, such as data, integration, usability, phasing-out, etc. Lastly, the "so-that" part should be an optional field. Practitioners should refrain from filling the "so-that" unjustifiably, as this may lead to an unverifiable requirement.

This case study has identified several directions for future research. More empirical investigations on a larger scale need to be conducted to assess the quality of user stories in real-world projects. Researchers may consider a different context, e.g., assessing user stories in an agile environment or comparing the quality of user stories with alternative techniques. Further, researchers may consider different quality criteria, for instance, the Quality User Framework [12], to evaluate the user story quality.

Author Contributions: M.A.K.: literature review, writing, editing, project administration, conceptualization, methodology, data curation, and validation. S.L.: writing, editing, conceptualization, methodology, and data curation. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: This research received ethical clearance from the research ethics committee at Zayed University (Application No. ZU21_065_F).

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study

Data Availability Statement: The data analyzed in this study can be found in [31].

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table 3. Stakeholder Issues (Needs and Wishes).

ID	Stakeholder Issue (Need or Wish)	Replies									Coverage
		A	B	C	D	E	F	G	H	S	A–H
A1	The Z-Department is a department of the Danish government. It has around 1000 IT users. It has its own hotline (help desk). They are unhappy with their present open-source system for hotline support and want to get a better one. They do not know whether to modify the system they have or buy a new one in a tender process.	0	0	0	0	0	0	0	0	1	0%
A2	The users encounter problems of many kinds. For instance, they may have forgotten their password, so they cannot start their work; or the printer lacks toner; or they cannot remember how to make Word write in two columns. The problem may also be to repair something, for instance a printer, or to order a program the user needs.	0	½	0	0	0	0	0	1	0	19%
A3	The easiest solution is to phone hotline or walk into their office. In many cases, this solves the problem right away. However, hotline prefers to receive the problem request by e-mail to hotline@ ... Sometimes this is impossible, for instance if the problem is that the user has forgotten his password.	½	1	0	½	1	0	0	1	1	50%
A4	If the user cannot have his problem solved right away, it is annoying not knowing when it will be solved. How often will he for instance have to go to the printer to check whether it has got toner now? In many cases the problem has been solved, but the user does not know.	0	1	1	½	½	1	½	½	1	63%
A5	The present support system allows the user to look up his problem request to see what has happened, but it is inconvenient and how often should he look?	0	1	0	0	½	0	0	½	1	25%
A6	Hotline is staffed by supporters. Some supporters are first line, others are second line. First-line supporters receive the requests by phone or e-mail, or when the user in person turns up at the hotline desk.	½	1	1	1	1	0	0	0	1	56%
A7	In busy periods, a first-line supporter may receive around 50 requests a day. Around 80% of the requests can be dealt with right away, and for these problems it is particularly hard to ensure that supporters record them for statistical use.	0	½	½	0	0	0	½	0	1	19%

A8	The remaining 20% of the requests are passed on to second line. Based on the problem description and talks with the user, first line can often give the request a priority and maybe an estimated time for the solution. (Experience shows that users should not be allowed to define the priority themselves, because they tend to give everything a high priority.)	0	1	1	1	0	1	0	1	1	63%
A9	Half of the second-line requests are in principle easy but cannot be dealt with immediately. The supporter may have to move out of the office, for instance to change toner in the printer or help the user at his own PC. Usually this ends the request, but it may also turn into a long request because a specialist or a spare part is needed. Often the supporter visits several locations when he moves out of the office.	0	0	½	0	0	0	½	1	1	19%
A10	Around 10% of all problems are long requests because the problem has to be transferred to a hotline person with special expertise, or because spare parts and expertise have to be ordered from external sources. Transferring the problem often fails. The supporter places a yellow sticker on the expert's desk, but the stickers often disappear. Or the expert misunderstands the problem. For this reason, it is important that the expert in person or by phone can talk with the supporter who initially received the request, or with the user himself.	½	½	½	½	0	0	½	0	1	31%
A11	There are 10–15 employees who occasionally or full time serve as supporters. They know each other and know who is an expert in what. The supporters frequently change between first and second line, for instance to get variation. It happens, unfortunately, that a supporter moves to second line without realizing that nobody remains in first line.	1	0	0	0	0	0	½	½	1	25%
A12	The request is sometimes lost because a supporter has started working on it, but becomes ill or goes on vacation before it is finished.	0	½	0	1	0	0	0	0	1	19%
A13	Managers ask for statistics of frequent and time-consuming requests in order to find ways to prevent the problems. However, with the present system it is cumbersome to record the data needed for statistics. Gathering this data would also make it possible to measure how long hotline takes to handle the requests.	½	½	1	0	1	0	0	0	1	38%
A14	In busy periods, around 100 requests may be open (unresolved). Then it is hard for the individual supporter to survey the problems he is working on and see which problems are most urgent.	½	½	1	0	½	½	½	0	1	44%
A15	The present system automatically collects all e-mails sent to hotline@ ... and put them in a list of open requests. Figure 3 [was enclosed] shows an example of such a list. You can see the request number (Req), the	½	0	1	0	½	0	0	½	1	31%

	user (Sender, i.e., his e-mail address), the supporter working on it (Owner), and how long ago the request was received (Age).										
A16	You can also see when someone last looked at the request (ActOn). However, this is not really useful. It would be better to see when the request should be closed according to what the user expects. It would be nice if the system warned about requests that are not completed on time.	1	½	0	1	½	0	0	0	1	38%
A17	When a user calls by phone or in person, the supporter creates a new request. It will appear in the normal list of requests. However, when the supporter can resolve the request right away, he often does not record it because it is too cumbersome. This causes misleading statistics.	0	½	1	0	½	0	½	0	1	31%
A18	As you can tell from the figure, the system cannot handle the special Danish letters (æ, ø, å), and it is not intuitive what the various functions do. The user interface is in English, which is the policy in the IT-department. The system has a web-based user interface that can be used from Mac (several supporters use Mac) and mobile. However, it has very low usability and is rarely used.	½	0	0	1	1	0	½	0	1	38%
A19	Anyway, the basic principle is okay. A supporter keeps the list on the screen so he can follow what is going on. He can open an incoming request (much the same way as you open an e-mail), maybe take on the request (for instance by sending a reply mail), classify the case according to the cause of the problem (printer, login, etc.), give it a priority, transfer it to someone else, etc. When the request has been completed, the supporter closes it, and the request will no longer be on the usual list of open requests.	½	½	0	½	1	0	½	½	1	44%
A20	As you can see in Figure 3 [was enclosed], Status is not used at all. It is too cumbersome and the present state names are confusing.	0	0	1	½	½	½	½	½	1	44%
A21	Some of the supporters have proposed to distinguish between these request states: First-line (waiting for a first-line supporter), Second-line (waiting for a second-line supporter), Taken (the request is currently being handled by a supporter), Parked (the request awaits something such as ordered parts), Reminder (the request has not closed in due time), and Closed (handled, may be reopened).	½	½	½	1	½	0	1	½	1	56%
A22	Open requests are those that are neither parked, nor closed.	0	½	0	0	0	0	0	0	1	6%
A23	For statistical purposes and to support the resolution of the request, it would be useful to keep track of when a request has changed state.	0	0	0	½	0	0	1	0	1	19%
A24	The present system can be configured to record a problem cause, but then the system insists that a cause	0	0	0	1	½	0	½	0	1	25%

	be recorded initially, although the real cause may not be known until later. In addition, somebody must set up a list of possible causes, and this is a difficult task. As a result, causes are not recorded, and statistics are poor.										
A25	While a long request is handled, it may receive additional information from the original user as well as from supporters. In the present system it is cumbersome to record this, and as a result the information may not be available for the supporter who later works on the request.	½	1	1	0	1	1	1	1	1	81%
A26	A supporter can set the system to send an ordinary e-mail to himself when he has to look at some request. This is particularly useful for second-line supporters who concentrate on other tasks until they are needed for support.	0	1	0	0	½	0	½	0	1	25%
A27	The Z-Department uses Microsoft Active Directory (AD) to keep track of employee data, e.g., the user's full name, phone number, office number, e-mail address, username, password and the user's access rights to various systems. The support system should retrieve data from AD and not maintain an employee file of its own.	1	0	0	1	0	0	1	0	1	38%
A28	When getting a new system, they imagine running the new and old system at the same time until all the old requests have been resolved. However, if the old requests can be migrated to the new system at a reasonable price, it would be convenient.	1	0	0	0	0	1	0	0	1	25%
A29	They expect to operate the system themselves and handle security as they do for many other systems.	0	0	0	0	0	0	0	0	1	0%
A30	They imagine connecting the new hotline system to systems that can generate hotline requests when something needs attention. Examples: The system that monitors the servers could generate a request when a server is down. Systems that keep track of employee's loan of various items, could generate a request when the item is not returned on time.	½	½	0	0	0	½	0	0	1	19%
	Total (rounded down)	9	13	11	11	11	5	10	8	29	Average 33%

References

1. Beck, K.; Andres, C. *Extreme Programming Explained: Embrace Change*; Addison-Wesley: Boston, MA, USA, 2004.
2. Cohn, M. *User Stories Applied: For Agile Software Development*; Addison-Wesley Professional: Boston, MA, USA, 2004.
3. Jeffries, R.; Hendrickson, M.; Anderson, A.; Hendrickson, C. *Extreme Programming Installed*; Addison-Wesley Professional: Boston, MA, USA, 2000.
4. Beck, K.; Fowle, M. *Planning Extreme Programming*; Addison-Wesley Professional: Boston, MA, USA, 2000.
5. Cohn, M. *Agile Estimating and Planning*; Pearson: Upper Saddle River, NJ, USA, 2005.
6. Wang, X.; Zhao, L.; Wang, Y.; Sun, J. The Role of Requirements Engineering Practices in Agile Development: An Empirical Study. In *Requirements Engineering*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 195–209. https://doi.org/10.1007/978-3-662-43610-3_15.
7. Dimitrijević, S.; Jovanović, J.; Devedžić, V. A comparative study of software tools for user story management. *Inf. Softw. Technol.* **2015**, *57*, 352–368. https://doi.org/10.1016/j.infsof.2014.05.012.

8. Lucassen, G.; Dalpiaz, F.; Werf, J.M.; Brinkkemper, S. The Use and Effectiveness of User Stories in Practice. In *Requirements Engineering: Foundation for Software Quality: Proceedings of the 22nd International Working Conference, REFSQ 2016, Gothenburg, Sweden, 14–17 March 2016*; Springer: Cham, Switzerland, 2016; pp. 205–222. https://doi.org/10.1007/978-3-319-30282-9_14.
9. Cao, L.; Ramesh, B. Agile Requirements Engineering Practices: An Empirical Study. *IEEE Softw.* **25**, *2008*, 60–67. https://doi.org/10.1109/MS.2008.1.
10. Ramesh, B.; Cao, L.; Baskerville, R. Agile requirements engineering practices and challenges: An empirical study. *Inf. Syst. J.* **2010**, *20*, 449–480. https://doi.org/10.1111/j.1365-2575.2007.00259.x.
11. Society, I.C. 830-1998-IEEE Recommended Practice for Software Requirements Specifications. *IEEE Std* **2009**.
12. Lucassen, G.; Dalpiaz, F.; Werf, J.M.; Brinkkemper, S. Forging High-Quality User Stories: Towards a Discipline for Agile Requirements. In Proceedings of the 23rd IEEE International Conference on Requirements Engineering, Ottawa, ON, Canada, 24–28 August 2015. https://doi.org/10.1109/RE.2015.7320415.
13. Savolainen, J.; Kuusela, J.; Vilavaara, A. Transition to Agile Development-Rediscovery of Important Requirements Engineering Practices. In Proceedings of the 18th IEEE International Requirements Engineering Conference, Sydney, NSW, Australia, 27 September–1 October 2010. https://doi.org/10.1109/RE.2010.41.
14. Danevaa, M.; Veena, E.V.; Amrita, C.; Ghaisasb, S.; Sikkela, K.; Kumarb, R.; Wieringaa, R. Agile requirements prioritization in large-scale outsourced system projects: An empirical study. *J. Syst. Softw.* **2013**, *86*, 1333–1353.
15. Ernst, N.A.; Murphy, G.C. Case studies in just-in-time requirements analysis. In Proceedings of the Second IEEE International Workshop on Empirical Requirements Engineering (EmpiRE). Chicago, IL, USA, 25 September 2012. https://doi.org/10.1109/EmpiRE.2012.6347678.
16. Paetsch, F.; Eberlein, A.; Maurer, F. Requirements engineering and agile software development. In Proceedings of the Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Linz, Austria, 11 June 2013; pp. 308–313. https://doi.org/10.1109/ENABL.2003.1231428.
17. Behutiye, W.; Karhapää, P.; Costal, D.; Oivo, M.; Franch, X. Non-functional Requirements Documentation in Agile Software Development: Challenges and Solution Proposal. In *Product-Focused Software Process Improvement: Proceedings of the 18th International Conference, PROFES 2017, Innsbruck, Austria, 29 November–1 December 2017*; Felderer, M., Méndez Fernández, D., Turhan, B., Kalinowski, M., Sarro, F., Winkler, D., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2017; Volume 10611. https://doi.org/10.1007/978-3-319-69926-4_41.
18. Farid, W.M.; Mitropoulos, F.J. NORMATIC: A visual tool for modeling Non-Functional Requirements in agile processes. In Proceedings of the IEEE Southeastcon, Orlando, FL, USA, 15–18 March 2012; pp. 1–8. https://doi.org/10.1109/SECon.2012.6196989.
19. Wake, B. INVEST in Good Stories, and SMART Tasks. 2003. Available online: http://xp123.com/articles/invest-in-good-stories-and-smart-tasks (accessed on 16 October 2020).
20. Heck, P.; Zaidman, A. A Quality Framework for Agile Requirements: A Practitioner's Perspective. *arXiv* **2014**. https://doi.org/10.48550/arXiv.1406.4692.
21. Wautelet, Y.; Heng, S.; Kolp, M.; Mirbel, I. Unifying and Extending User Story Models. In *Advanced Information Systems Engineering: Proceedings of the 26th International Conference, CAiSE 2014, Thessaloniki, Greece, 16–20 June 2014*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2014; Volume 8484. https://doi.org/10.1007/978-3-319-07881-6_15.
22. Wautelet, Y.; Heng, S.; Kolp, M.; Mirbel, I.; Poelmans, S. Building a rationale diagram for evaluating user story sets. In Proceedings of the 2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS), Grenoble, France, 1–3 June 2016; pp. 1–12. https://doi.org/10.1109/RCIS.2016.7549299.
23. Wautelet, Y.; Velghe, M.; Heng, S.; Poelmans, S.; Kolp, M. On Modelers Ability to Build a Visual Diagram from a User Story Set: A Goal-Oriented Approach. In *Requirements Engineering: Foundation for Software Quality: Proceedings of the 24th International Working Conference, REFSQ 2018, Utrecht, The Netherlands, 19–22 March 2018*; Lecture Notes in Computer Science; Kamsties, E., Horkoff, J., Dalpiaz, F., Eds.; Springer: Cham, Switzerland, 2018; Volume 10753. https://doi.org/10.1007/978-3-319-77243-1_13.
24. Tsilionis, K.; Maene, J.; Heng, S.; Wautelet, Y.; Poelmans, S. Conceptual Modeling Versus User Story Mapping: Which is the Best Approach to Agile Requirements Engineering? In *Research Challenges in Information Science: Proceedings of the 15th International Conference, RCIS 2021, Limassol, Cyprus, 11–14 May 2021*; Lecture Notes in Business Information Processing; Cherfi, S., Perini, A., Nurcan, S., Eds.; Springer: Cham, Switzerland, 2021; Volume 415. https://doi.org/10.1007/978-3-030-75018-3_24.
25. Condori-Fernandez, N.; Daneva, M.; Sikkel, K.; Wieringa, R. A Systematic Mapping Study on Empirical Evaluation of Software. In Proceedings of the Third International Symposium on Empirical Software Engineering and Measurement, Lake Buena Vista, FL, USA, 15–16 October 2009; pp. 502–505.
26. Dalpiaz, F.; Sturm, A. *Conceptualizing Requirements Using User Stories and Use Cases: A Controlled Experiment. REFSQ 2020: Requirements Engineering: Foundation for Software Quality*; Springer: Cham, Switzerland, 2020. https://doi.org/10.1007/978-3-030-44429-7_16.
27. Dalpiaz, F.; Gieske, P.; Sturm, A. On deriving conceptual models from user requirements: An empirical study. *Inf. Softw. Technol.* **2021**, *131*, 106484. ISSN 0950-5849. https://doi.org/10.1016/j.infsof.2020.106484.
28. Inayat, I.; Salim, S.S.; Marczak, S.; Daneva, M.; Shamshirband, S. A systematic literature review on agile requirements engineering practices and challenges. *Comput. Hum. Behav.* **2015**, *51*, 915–929.

29. Lauesen, S.; Kuhail, M.A. Use Cases versus Task Descriptions. In *Requirements Engineering: Foundation for Software Quality*; Lecture Notes in Computer Science; Berry, D., Franch, X., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6606. https://doi.org/10.1007/978-3-642-19858-8_13.
30. De Lucia, A.; Qusef, A. Requirements Engineering in Agile Software Development. *J. Emerg. Technol. Web Intell.* **2003**, *2*, 212–220. https://doi.org/10.4304/jetwi.2.3.212-220.
31. User Story Experiment Assignment and Replies. Available online: http://www.itu.dk/~slauesen/UserStories/ accessed on 25 March 2021).
32. Atlassian. Jira Software Tool. Available online: https://www.atlassian.com/software/jira (accessed on 19 March 2021).
33. Gotel, O.C.; Finkelstein, A.C. An Analysis of the Requirements Traceability Problem. In Proceedings of the IEEE International Conference on Requirements Engineering, Colorado Springs, CO, USA, 18–22 April 1994; pp. 94–101.
34. Consortium, A.B. *The DSDM Agile Project Framework Handbook*; Buckland Media Group: Dover, UK, 2014.
35. Kulak, D.; Guiney, E. Use Cases: Requirements in context. *ACM SIGSOFT Softw. Eng. Notes* **2001**, *26*, 101. https://doi.org/10.1145/505894.505926.
36. Lauesen, S. Problem-Oriented Requirements in Practice–A Case Study. In *Requirements Engineering: Foundation for Software Quality*; Springer: Cham, Switzerland, 2018. https://doi.org/10.1007/978-3-319-77243-1_1.