Why We Need a Granularity Concept for User Stories

Olga Liskin, Raphael Pham, Stephan Kiesling, and Kurt Schneider

Software Engineering Group, Leibniz Universität Hannover, Germany
{olga.liskin, raphael.pham, stephan.kiesling, kurt.schneider}
@inf.uni-hannover.de

Abstract. User stories are a widespread instrument for representing requirements. They describe small user-oriented parts of the system and guide the daily work of developers. Often however, user stories are too coarse, so that misunderstandings or dependencies remain unforeseeable. Granularity of user stories needs to be investigated more, but at the same time is a hard-to-grasp concept.
This paper investigates *Expected Implementation Duration (EID)* of a user story as a characteristic of granularity. We want to find out, whether it is suitable as a quality aspect and can help software teams improve their user stories.
We have conducted a study with software engineering practitioners. There, many user stories had a relatively high EID of four or more days. Many developers state to have experienced certain problems to occur more often with such coarse user stories. Our findings emphasize the importance to reflect on granularity when working with user stories.

Keywords: user stories, user requirements, requirements quality

1 Introduction

Communication plays a crucial role in software development [3]. Collaboration of a group of developers demands for an effective way of exchanging information and coordination. Communication is especially important with regard to software requirements. Inadequate communication between team members can lead to team members misunderstanding core requirements [6] and subsequently to the development of undesired software - thus jeopardizing the project's success.

User stories, as one form of requirements, have the potential to divide a complex system into small user oriented pieces, which can be implemented independently. At the same time, such user stories have a great influence on the daily work of all involved team members. The quality of user stories impacts communication and coordination in a project and therefore plays an important role. When trying to understand, how user stories impact the daily work of a software team, their granularity is an interesting aspect.

The granularity of a user story can heavily impact its quality: A user story with otherwise good quality features (such as a clear priority and attached acceptance tests) could still be disastrously underestimated by developers, if it is too coarsely grained.

The reason behind this is that a flawed granularity indicates a number of problems with a user story. A coarsely grained user story could be formulated unclearly and pose difficulties for estimating the associated effort. This can also point to inter-team and customer-related communication gaps.

Granularity of a user story has many facets. It can be understood in terms of:

- *Clarity/vagueness*. If a user story leaves out a lot of information, it is written vaguely.
- *Concreteness/abstraction*. A user story can describe the desired functionality as an abstract concept or already sketch a concrete manifestation of this concept.
- *Scope*. This represents the scope of the system functionality that is described *or meant* by the information given in a user story. A user story that implies a lot of system functionality (and according implementation work) would have a large scope.

While all three aspects are important for the quality of a user story, we focus on granularity in the sense of scope size. In order to change the scope of a user story, the desired functionality must be changed. For example, in order to reduce implementation work, some of the desired functionality must be removed or the story must be split into smaller ones. In contrast, the clarity or abstractness of a user story is varied by providing different information about the desired functionality that the customer has in mind, while the functionality remains the same.

The three aspects are orthogonal. For example, a user story can be clear and have a high scope size (imply a lot of implementation work), while another story can have the same scope size, but be vague. However, there is a chance that reducing the scope of a user story by splitting it into multiple smaller stories can improve its clarity and help make it more concrete. This aspect needs further research though.

In practice, some scope-related aspects like effort or complexity of user stories are estimated in order to characterize user stories [5]. For this, abstract scales, like t-shirt sizes or Fibonacci numbers are used, so that, again, individual values mean different things to different teams.

Therefore, we see the need for a more tangible and comparable concept in order to be able to investigate granularity of user stories. We define the *Expected Implementation Duration (EID)* of a user story as a viable concept for quantifying the scope of a user story. Then, we demonstrate why it is a good quality aspect for user stories and why it should be taken into account when working with user stories.

We picked user stories for our study because we think that differences in the understandings of what a user story is are smaller for user stories than for other requirements concepts. Further, user stories are known to take a relatively small amount of time to be implemented [4]. However, we think that our findings are applicable to other requirements concepts as well.

The paper is structured as follows: In the next section, we present related literature. Section 3 shows the research questions we have based our study upon. In Section 4 the study design is sketched, followed by the study results in Section 5. Section 6 discusses Threats to Validity. We conclude with a discussion and outlook.

2 Related Work

In agile software development, user stories and story cards are a widely practiced form of documenting requirements. The activity of estimating user stories has been in the focus of recent literature [5]. Miranda et al. [10] focus on improving these estimation strategies as well as improving the estimations. Haugen, Mahniç et al. and Tamrakar et al. [7][9][13] examine whether introduction of planning poker improves the team's ability to estimate user stories. Furthermore, Imaz and Benyon [8] describe a way to enhance traceability between user stories as pre-requirements and semi-formal requirements such as use cases. Cohn [5] specifies how to split up user stories. However, Patel and Ramachandran [11] describe the lack of clear guidelines or rules for aspects of a *good* user story and motivate research in this area.

According to Cohn [5], the *ideal* days measurement is as good as *story points* – as long as the organizational overhead is ignored. He proposes to estimate user stories in *ideal* days and emphasizes to not rely on *elapsed* days as a measurement. EID is related to Cohn's notion of ideal days. This study is a first step to understand how this measure is perceived and handled by practitioners.

Wake [14] suggests with INVEST[1] six quality criteria for a good user story. One of these says that a user story should be *small*. With our work, we try to further concretize this criterion. First, EID is a possible means to determine whether a user story is *small*. Further, it is not known yet, whether it is easy to measure and act on story size in practice. Second, so far publications only explain theoretically why a small story is good. We want to substantiate this with real data and experiences from practice.

Many studies have been conducted in the field of agile requirements engineering in order to understand the practitioners' perspectives. Cao and Ramesh [2] revealed agile RE practices in an empirical study. Bjarnason et al. [1] examine overscoping and therewith bring up a topic that is also related to planning with requirements. With a user study on user story implementation duration we try to complement these works to help understand how agile requirements are handled in industry.

3 Expected Implementation Duration as a Quality Aspect for User Stories

In our context, the term granularity represents the scope of the system functionality that is described or meant by the information given in a user story. Expected implementation duration is a way of quantifying the scope size of a user story.

By *Expected Implementation Duration* we mean the estimated time (in days) that a developer or pair will need to implement a user story. Implementation of a user story includes coding as well as all other tasks that the developer performs to deliver a user story, such as designing or testing. Like Cohn's concept of ideal days [5], EID ignores tasks that are not related to a user story but are done in-between its implementation.

[1] Independent, Negotiable, Valuable, Estimable, Small, Testable.

Expected implementation duration has the potential to add new value to the characterization of user stories. This can generally improve the development timetable. If the estimation of implementation time for a user story exceeds a certain threshold (for example, one week or more), chances are high that it becomes more inaccurate. Humans are better at grasping events when they are in the near future. When thinking about events that cover a long period of time, it is easy to forget an event or to misestimate one of these events. If implementation time takes a week or more, a seemingly simple user story with low complexity can still be underestimated in terms of when it will actually be finished. Likewise, a user story that is described in much detail, might still be missing information if its expected implementation duration is too long.

However, the expected implementation duration is still an estimation. And, as with other metrics, estimated values can be wrong. In this context, we are not aiming at hourly precision. Literature suggests that a user story should not take more than one or two days to be implemented [12]. The reality of implementation durations of user stories often looks different. In the study presented by this paper, 50% of the participants have stated that 30% or more of their user stories require more than four days to be implemented. We want to raise awareness to this quality aspect, so that in the future, user stories are shrunk to an implementation time of no more than one day.

In order to show that Expected Implementation Duration is a valid aspect, we propose the following **research questions**:

RQ1: Is Expected Implementation Duration easy to measure?

A quality aspect is only applicable, if the user is able to measure it and is able to obtain meaningful values. We investigate if practitioners are able to express the EID of a user story. Furthermore, we inquired how practitioners measure other aspects of a user story, such as its complexity.

RQ2: Which actual Expected Implementation Duration values do user stories in current software projects have?

In order to understand how teams handle their user stories with respect to Expected Implementation Duration, it is also meaningful to get a picture of current EID values in real projects. It is especially interesting to see, how big differences are among projects as well as among user stories within the same project.

RQ3: Is it possible to control Expected Implementation Duration for user stories by splitting them?

Obtaining the current EID characteristics for a user story is beneficial. However, users should also be able to influence it. We investigate what opinion practitioners have of splitting user stories and at what size user stories should be split. We are interested in experiences they have made when splitting user stories and which problems arise when doing so.

RQ4: Is Expected Implementation Duration a relevant factor for user stories?

We want to clarify whether the EID is worth investigating by investigating its relevance. If other quality aspects, such as a clear priority, are more important for a workable user story than its EID, developers will not need to focus on it.

We found that EID or related aspects are perceived as easy to assess. The participants used different strategies for this, but had problems with under-/overestimation (RQ1, Section 5.1). The actual EID values stated in the study vary among most projects. User stories that take four days or longer have a relatively high portion of 32% (RQ2, Section 5.2). The participants believed that it is possible to split user stories in many cases, especially when they take longer than four days. However, we found various challenges with splitting user stories (RQ3, Section 5.3). Further, we found that the scope size of a user story controls many relevant aspects such as communication, planning, and detection of dependencies (RQ4, Section 5.4).

4 Survey

To gain a first understanding of how practitioners handle user requirements with regard to its "size", we conducted a two-staged study using two separate online questionnaires. Our target population were practitioners with experience in industrial software projects. The GitHub Archive[2] records any user activity (forking, pull requesting, commenting) on the social coding site GitHub[3]. We queried the GitHub Archive for users who had specified a company name (non-empty string) in their GitHub profile and had been active on GitHub at the beginning of September 2013.

In a preliminary study, we invited 400 GitHub users to participate in a questionnaire and received 68 answers (response rate 17%). We encouraged the user to share experiences from an industrial project, but let her answer for a private project if needed. This first questionnaire covered three topics: Which form did the participant's requirements take (user stories, use cases, UML models, plain text), what factors made a good requirement and what challenges did the participant have when handling these requirements. This first round of broad questions and answers enabled us to focus our efforts on more specific challenges: Challenges regarding inter-team communication seemed to be less prominent than others and were subsequently not investigated further. Furthermore, the number of different forms of requirements documentation (user stories, use cases, UML models, etc.) left us with only a vague understanding of how the practitioners handled their requirements in particular.

Hence, for the main questionnaire in mid-September 2013, we concretely focused on user stories. We invited 600 GitHub users to partake in this second questionnaire and received 72 answers (12% answer rate). This questionnaire enquired more deeply how the questionees judged the implementation time of a user story and whether this estimation would work as a well-defined concept (see Section Results).

[2] http://www.githubarchive.org/
[3] http://github.com

5 Results

Of the 72 answers from the main questionnaire, 27 (38%) participants stated to have no experience with user stories. The answers of these participants were removed from the final set of answers since the invitation directly addressed users with experience in using user stories. Of the remaining 45, 33 (73%) participants enlisted to report on experiences with user stories in industrial software projects. However, 4 of those were not able to estimate the expected implementation time of a user story. As this is an important foundation of our questions, we removed these participants' answers from the final set of answers as well. 11 participants reported on private projects. One participant did not state her project's origin – we counted her answers towards private projects. Eventually, we counted 41 participants from industrial projects and 12 participants from private projects, resulting in a total of 53 usable answers. The project sizes for this population are shown in Fig.1. We asked the participants, in how many projects they have worked with user stories. This gave us a better understanding of the participants' general experience with user stories.

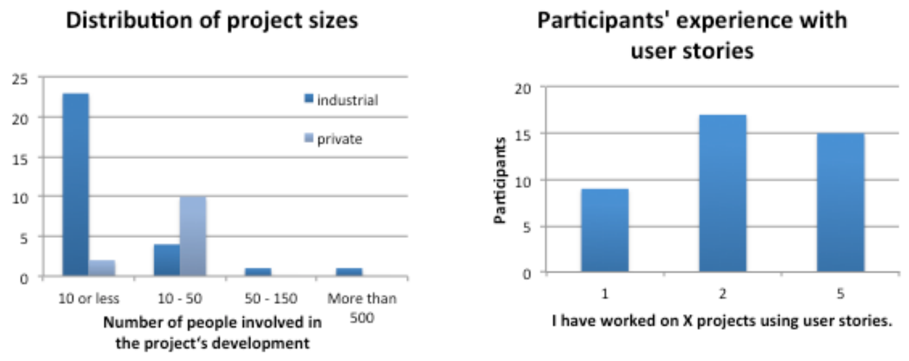

Fig. 1. Population characterization

5.1 Measurability of Expected Implementation Duration (RQ 1)

Can developers estimate the expected time for implementation?

We asked the participants, whether or not they take the estimated time to implement a user story into consideration when rating the effort of a user story. 49% (26 participants) stated that time estimation was a factor for rating a user story (see Fig. 2), while 9% (5) saw no connection. Seven participants generally did not rate their story cards, but did not comment further on the matter. We also asked the questionees whether or not they were able to estimate a story card's expected time to implement. 15% (8) of the participants reported to be able do so and that they always work with user stories that were estimated according to the expected implementation time. Additionally, 55% (29) of participants claimed to be able to do so, if necessary, but did not do this on a daily basis. 3 participants did not answer this question.

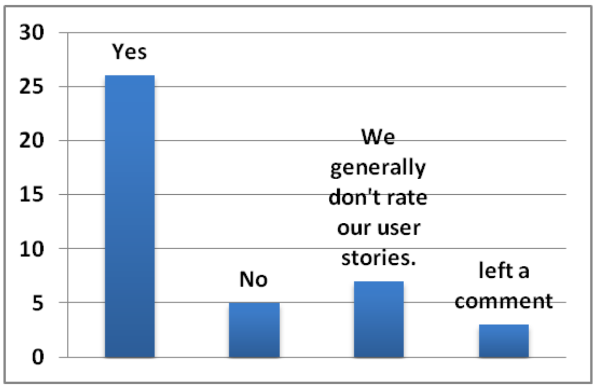

Fig. 2. Consideration of time to implement when rating a user story

Methods for estimating the expected implementation time of a user story

We inquired further and asked the developers to specify their strategy for estimating the expected implementation time of a user story. 6 participants (3 from industry and 3 from private projects) reported to use only a vague review or team discussion or just *guessing* freely and not employing any clear methodology. Interestingly, one of them admitted to often *underestimate* the implementation time ("We basically just guess. There isn't any science behind the amount of time expected to complete a story, and we don't even track the amount of hours we actually work on it so that we can gauge whether or not we were accurate. I believe we drastically underestimate the amount of time it takes to complete a story, though."). On the other hand, two industry participants used *elaborate strategies* such as the Planning Poker game and mapping abstract story points to time values. However, this seemed to work best with experienced developers ("Based on our understanding of the value of a story point we can then directly map this to a time estimate. This *only* works with a seasoned team."). Participants reported more generally on estimation strategies for user stories, which involved time estimation. These started by a *clarification phase* in which the developers tried to gain a thorough understanding of the given user story. When estimating the implementation time, developers often relied on their *general experience* and based their estimates on *similar, existing systems*. Often, user stories would be *compared to another*, using the simplest user story as a *well-known baseline*. Participants tried to *eliminate unknown technologies* and unknown risks and would *refuse* user stories that did not comply their requirements ("… I refuse to estimate implementation time if they involve technologies that are foreign to me. Foreign technologies have no place in implementation user stories as they introduce numerous unknowns. Such issues have to be figured out before the implementation phase."). User story estimation sometimes involved *skill assessment* of the current team. Developers would also try to *analyze* the user story's *impact* on existing components of the *system, its test suite* or *legacy systems* ("Breakdown of tasks, evaluation of the code quality of the legacy (any existing code is legacy in this sense) code that the new functionality needs to integrate with"). Eventually, most developers tried to *split user stories up*. Some participants used abstract *measures* (“as-simple-as-possible”), others had found hard time values to work best (“16 hours”, “around a day”, “Make sure the story is a good size (a week of work max IMO)”).

Problems when estimating the expected implementation time

We enquired what kind of problems were experienced, when estimating the time of implementation for a user story. Firstly, we wanted to know, whether over- or underestimation was a problem and proposed each an over- and underestimation option. Secondly, we intended to gain further insights into other problems and added a third, user-editable option. All three choices could apply in any combination [checkboxes]. Of the final data set of 53 usable participants, none had left the answer to this question blank. Seven participants added their own answer, which we will explain further below, and we counted these seven answers against over- or underestimation. Table 1 shows the distribution of answers in detail. 12 participants found over- and underestimation to be problematic and we counted these answers both towards over- and underestimation. In Table 1, we see that underestimation of implementation time seems to be more problematic for developers than constant overestimation. One could argue that overestimation does not seem as problematic to a developer – she has finished her work before deadline after all – and, thus, overestimation is underrepresented here. However, no such comment or indication was given by our questionees.

Two participants deemed the expected implementation time to be unreliable, especially when handling unknown technologies (“When utilizing unfamiliar technologies I've noticed that story estimates are mostly inaccurate.”). Two other participants found user stories to inaccurately state user intentions and thus being hard to estimate time wise (“The user stories in general rarely reflect what the actual user will use the software for causing unforeseen changes during the scrumlike iterative feedback process”). Another two participants seemed to have no problems with a little over- or underestimation and also linked this to the complexity or novelty of the user story (“Sometimes over, sometimes under, but for simplest or repetitive stories the estimates are quite precise.”).

Table 1. Number of participants who have experienced over- and underestimation. 'Com.' stands for 'Comments'

		underestimation		
		yes	no	
over-estimation	yes	12	5	**17**
	no	17	Com.: 7	
		29		

5.2 Characterization of Expected Implementation Duration of User Stories (RQ2)

We asked the participants which expected implementation durations user stories they work with typically have. We presented 5 options – a few hours, one day, 1-3 days, 4-5 days, and more than one week - and asked the participants to distribute 100 percent points to reflect the distribution of the different durations for their user stories.

34 participants have answered this question. In total, they have distributed 3400 points. Figure 3 depicts how the 3400 points were distributed to the different EID options in total. 43% of the total quantity of points was assigned to user stories that last only one day or less (answers for option "one day" and "a few hours" aggregated). Thus, 57% of all rated user stories in this study take longer than one day. More specifically, 13% even take longer than one week.

Fig. 3 also distinguishes the distribution of points for industrial (2500 points assigned) and private projects (900 points assigned). Here, some peculiarities can be seen. In both groups, short under-one-day user stories make up for about half of the user stories. However, in private projects, a greater part (30%) of that amount falls into the "a few hours"-category. In industry, these hours-long user stories only add up to 16%. On the other hand, one-day user stories are more present in industrial projects (26%) than in private projects (18%). Then again, user stories that take more than one week are more prominent in private projects (21% vs. 11%).

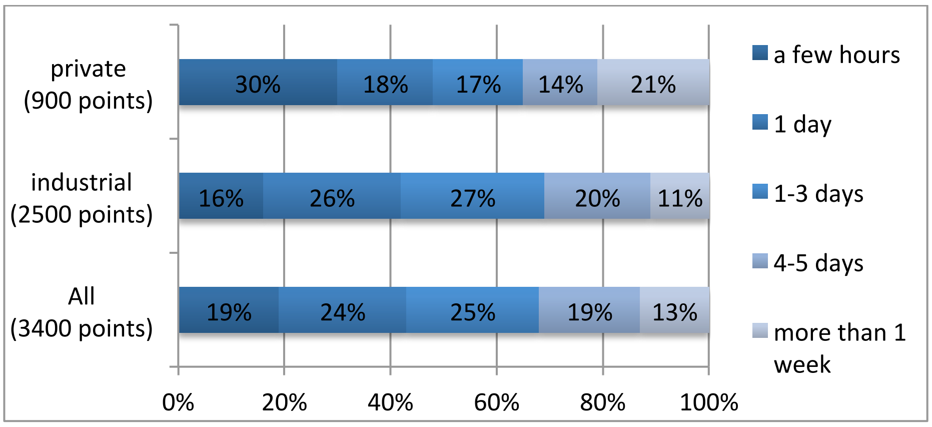

Fig. 3. Distribution of different user story durations

In Table 2, we aggregated the distributed points. First, we aggregated the different possible durations. For example, the column "more than 4 days" contains information about user stories with an EID of 4-5 days and user stories with an EID of more than 1 week. Further, we differentiated, how many users have allocated a number of percent points that is higher than a certain threshold (left column) to each of the aggregated groups. From this table, we can see that more than half of the respondents (53%) have assigned 30 or more of 100 points to user stories that take more than 4 days to complete. This means, for more than half of the respondents about one third of their user stories take more than 4 days. Furthermore, 58% of the respondents have stated that about two thirds (60%) of their user stories last more than one day.

Table 2. Percent of the questionees who have allocated x or more points to user stories of the according sizes

Allocated Points	One day or more	more than 1 day	more than 4 days	more than 1 week
>=0	100	100	100	100
>=10	97,06	97,06	88,24	58,82
>=20	97,06	94,12	70,59	29,41
>=30	97,06	85,29	52,94	11,76
>=40	97,06	73,53	38,24	11,76
>=50	97,06	67,65	29,41	5,88
>=60	88,24	58,82	17,65	2,94
>=70	85,29	35,29	8,82	2,94
>=80	67,65	26,47	2,94	2,94
>=90	50	11,76	0	0

5.3 Controlling Expected Implementation Duration (RQ3)

Experience in splitting user stories

Of the 39 people who answered this part of the questionnaire, 90% stated to have split up a user story before while 10% had no experience with splitting a user story.

Possibility to split user stories

We proposed different durations for expected implementation time (1 day, 1 to 3 days, 4 to 5 days) for one user story and asked the participants whether they thought that they could split such a user story. The results are depicted in Fig. 4. For 1-day-long user stories most participants stated that such a story can be split sometimes. Four participants even said that such a user story always should be split into smaller parts. For a user story which is expected to take 4-5 days, most participants thought that it should always be split into smaller user stories. Only three participants stated that a user story of 4-5 days could only sometimes be split into smaller pieces.

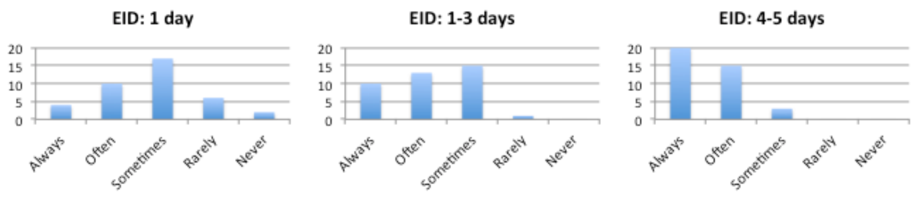

Fig. 4. Could a user story with the given expected implementation duration (EID) be split into smaller stories? (n= 39)

Challenges with splitting user stories

We asked the participants to describe problems they have encountered, when trying to split a user story. In total, 14 participants have mentioned problems. The most prominent problem was regarding *dependencies*. Six of the participants mentioned dependencies in their replies. Four respondents noted that user stories were sometimes *lacking in clarity* and therefore *additional communication with the customer* was required for splitting these user stories. This is an interesting aspect: The process of splitting a user story might reveal a user story that needs clarification. Further, one participant mentioned that a user story which gets too small, is *not very valuable* without other user stories.

5.4 Relevance of Expected Implementation Duration (RQ4)

Problems occurring when granularity is wrong

We wanted to know whether a long or a short EID caused specific problems. In total, 24 users commented to having experienced problems with stories that had a long EID, while seven users declined. Four participants found *hidden additional effort needs* problematic: This was the case, when requirements were unclear and additional effort was needed for clarification. Also, longer user stories were associated with a higher probability of encountering problems. Four participants mentioned *problems brought by change*, both for customer-initiated change or inter-team driven change.

Three users suggested different strategies to avoid these problems a priori. For example, the user story with the longest EID would be given the lowest priority by the customer or removed altogether.

When asked whether they have encountered problems with user stories with a short EID, 13 participants affirmed while 20 participants stated to not have had any problems. The 13 comments about problems with short user stories all came from industrial projects. Again, one person stated *hidden additional effort* as a problem, referring to a case when important functionality had simply been forgotten to specify. The most dominant problem, brought up by five participants, was the *difficulty to stick to the estimation*. In their experience, small user stories had been under- as well as overestimated. This was caused, for example, by wrong expectations of third party libraries.

While changes were stated as a problem with long user stories, nobody had brought up this problem for short user stories.

Relevance of EID for Communication

We believe that an increase in EID can lead to a less sketched-out user story and that this increase has effects on the communication between customer and developer. We tried to find out if participants have experienced effects of insufficient communication when handling a user story with a high EID. We presented a selection of communication problems to the participants and inquired whether they have experienced any of these effects or what they thought about them. Fig. 5 shows the suggested problems and the respective distributions among the participants.

Noticeably, most participants have experienced clarification problems when handling big user stories: either the number of post-development changes increased ("one has to change more details afterwards") or details were not clearly sketched out, causing unnecessary development ("more probable that one develops more than what was expected"). Also, bigger user stories influence the customer-developer feedback cycle. A significant number of users reported to have to wait longer for gathering feedback for a user story ("it takes longer until one can gather feedback about this story").
This finding is very important and underlines the important role of a valid concept for EID when handling user stories: short and effective feedback cycles with the customer are the basis of any serious agile software development. Without a useful guideline for estimating the implementation duration correctly, the developer is at risk of misjudging the EID considerably. In this situation, the developer will ultimately have to deal with user stories that take longer than expected – and the aforementioned effects follow: the crucial customer feedback cycle is disturbed and the number of change requests rises.

Additionally, some practitioners have experienced that it is harder to deliver exactly what the customer wanted when a user story is large ("it is harder to meet the customer's requirements"). This statement, however, had the least affirmation. Many participants also disagreed or had experienced the contrary.

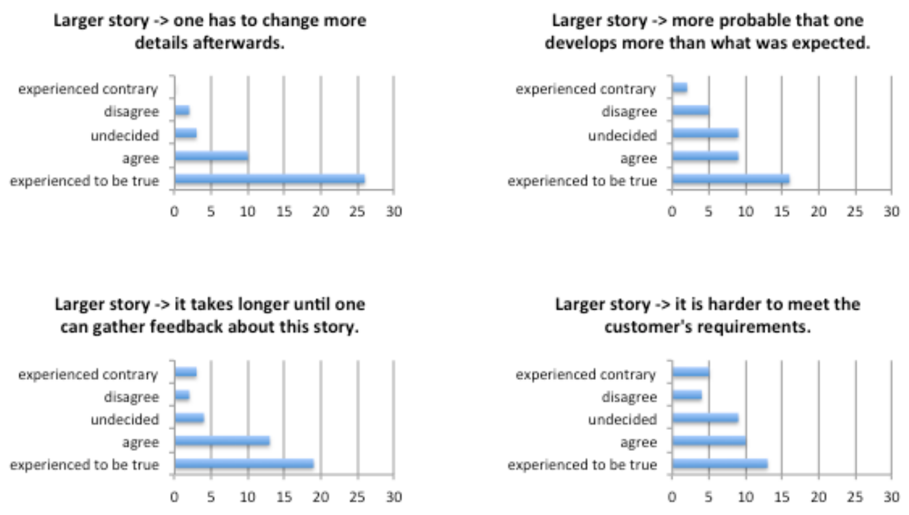

Fig. 5. Degrees of agreement on suggested communication problems (n=41)

Relevance of EID for Tangibility for Implementation

The other aspect we wanted to investigate with respect to EID relevance is the tangibility of a user story. The better a developer can picture all necessary actions for implementing a user story, the more likely it is that she will notice dependencies and risks. User stories that are too coarse can carry the risk to grow unexpectedly, for example due to underestimated dependencies. In this part, we describe whether our participants have experienced a better tangibility with smaller user stories. Again, we presented a selection of ideas - benefits that smaller user stories can have - and inquired about the participants' experiences. Fig. 6 shows the suggested benefits and the respective experiences of the participants.

The two most prominent statements are that smaller user stories allow more precise duration estimations and estimations about which actions within the code will be necessary ("our estimation of how long a user story will take is more precise i.e. it is true more often" and "I can estimate better how many code parts I will have to touch in order to implement it"). Here, more than 65% of the participants were able to confirm the statements from their own experience. This indicates that smaller user stories are indeed perceived to be more tangible. Further, the participants confirmed that they could better estimate which dependencies a user story has when it is smaller ("I can estimate better with how many other

Again, these findings underline the importance of a useful concept for EID: When practitioners assessed an EID for a user story to be small, this estimation turned out to be precise more often and thus enabling the developer to correctly judge this user story's implementation duration. This way, the developer can try to avoid user stories that will take longer and create risks of communication (see previous paragraph).

The claim that a smaller user story will less often grow ("it will less often grow unexpectedly in time/complexity") was objected the most. 30% of the participants either disagreed with this statement or even have experienced the contrary. In the first topic of this section, some participants had stated problems of this kind. They reported that sometimes a user story was just small because the developers had underestimated technical details or third party libraries.

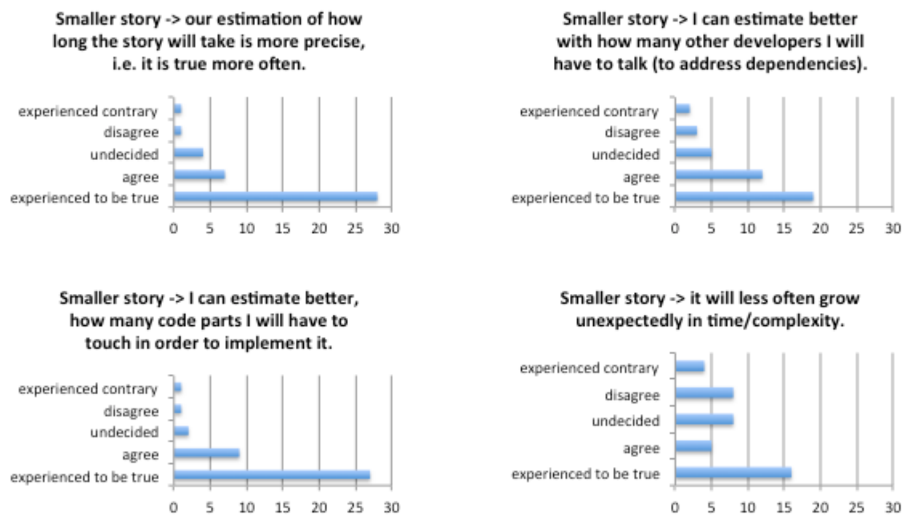

Fig. 6. Degrees of agreement on suggested benefits (n=41)

6 Threats to Validity

This study is a first step into recognizing the granularity of user stories. Our study resulted in 41 usable practitioner answers and 12 answers from private developers (results of the second, more focused questionnaire). Although we presented a fair amount of data points, we refrained from a more sophisticated statistical analysis. However, our study focuses on practitioner's opinions and real experiences and mostly deals with qualitative data. In this sense our study presents a trend analysis.

We are aware that the voluntary nature of our study is a threat to *internal validity*. Our population sample is self-chosen: we did not offer any kind of compensation and practitioners and private users participated voluntarily. This leaves the questionees' motivation to participate undefined.

This study introduces a new aspect of requirements quality, the expected implementation duration. Questioning the practitioner online and not in a personal interview could potentially lead to a different understanding of concepts like user story and user requirements. We tried to mitigate this threat to *construct validity* by explaining our understanding of such concepts in the questionnaire where necessary. Furthermore, our dataset and questionees' comments do not indicate an underlying misunderstanding.

Our study design only distinguishes between private and industrial users of user stories, regardless of the actual implementation of such user stories practiced or the requirements process. This is a threat to both *construct validity* and *external validity*. However, this study should serve as a first understanding of the EID-related issues of user stories and requirements. We believe that the presented concept of EID can be applied to most of the currently practiced variants of user stories. A follow-up study could analyze this more thoroughly.

We used a non-empty string in the company-tag of GitHub profiles to find practitioners in the sense of developers that develop software for a living. However, a non-empty company name string does not automatically ensure that this person currently works in a software company. We mitigated this *threat to construct validity* by specifically asking the participant to state whether she will report experiences about an industrial or private project.

7 Discussion and Outlook

We have presented Expected Implementation Duration as a concept to grasp the granularity of user stories. In a study with 72 participants we have seen that EID is a measurable and influenceable concept. A peek into real projects reveals that user story sizes (in the sense of EID) vary widely between different projects but also among single projects. More than half of the participants have stated to work in projects where more than 30% of user stories take four days or more. At the same time the practitioners state certain problems they have with large as well as small user stories.

We have seen that working with small user stories cannot prevent a project from unexpected surprises. Small user stories still can grow unexpectedly in time or complexity. Nevertheless, keeping an eye on EID can help developers get more manageable requirements. Concretely, smaller user stories are perceived as more tangible and predictable, they help keeping feedback-cycles short, and they entail less post-development changes.

We do not present a perfect duration for user stories. The perfect duration probably does not exist, since different story granularities are well-suited for different tasks. However, this paper emphasizes the importance of EID as an aspect of user stories.

Using EID as a metric could influence the estimations. Developers could make smaller estimations because they want to look better and have smaller stories. We do not want to encourage developers to kid themselves. We also do not see the necessity for this as long as there are alternative ways to influence EID, such as splitting a user story. However, there are risks and it would be interesting to see the effects of introducing EID as a quality metric for user stories in a real project.

After revealing requirements granularity as an important factor in requirements engineering, we see many interesting directions for more research. The developers' views on user story granularity and its effects should be complemented by the *customers' or product owners' perspectives*. It is especially likely that having very small user stories, and therefore many of them, can be a burden for product owners. In this case, having different views with different granularities on the same set of user requirements can be an interesting solution.

Further, we believe that the concept of EID can also be transferred to *other types of requirements*. Here, as we have seen, measuring the EID of a user requirement introduces interesting challenges, because there are so many different understandings of what a user requirement is.

Many participants of our first questionnaire have stated to split user requirements into more technical work packages and then actually work on these. The *relation between a user requirement and a work package* would also be an interesting aspect arising from our study.

References

1. Bjarnason, E., Wnuk, K., Regnell, B.: Are you biting off more than you can chew? A case study on causes and effects of overscoping in large-scale software engineering. Information & Software Technology, 54(10):1107–1124 (2012)
2. Cao, L., Ramesh, B.: Agile requirements engineering practices: An empirical study. IEEE Software, 25:60–67 (2008)
3. Cockburn, A., Highsmith, J.: Agile software development, the people factor. IEEE Computer, 34(11):131–133 (2001)
4. Cohn, M.: User Stories Applied: For Agile Software Development. Prentice Hall (2004)
5. Cohn, M.: Agile Estimating and Planning. Pearson Education (2005)
6. Coughlan, J., Macredie, R.D.: Effective communication in requirements elicitation: A comparison of methodologies. Requirements Engineering, 7(2):47–60 (2002)
7. Haugen, N.C.: An empirical study of using planning poker for user story estimation. In: Agile Conference, 2006, pp. 9 –34 (2006)
8. Imaz, M., Benyon, D.: How stories capture interaction. In: INTERACT'99, pp. 321–328. IOS Press, IFIP TC.13 (1999)

9. Mahniç, V., Hovelja, T.: On using planning poker for estimating user stories. Journal of Systems and Software, 85(9):2086 – 2095 (2012)
10. Miranda, E., Bourque, P., Abran, A.: Sizing user stories using paired comparisons. Information and Software Technology, 51(9):1327 – 1337 (2009)
11. Patel, C., Ramachandran, M.: Story card based agile software development. International Journal of Hybrid Information Technology, 2(2):125–140 (2009)
12. Schwaber, K., Beedle, M.: Agile Software Development with Scrum. Prentice Hall (2002)
13. Tamrakar, R., Jørgensen, M.: Does the use of Fibonacci numbers in Planning Poker affect effort estimates? In: Proceedings of the 16th International Conference on Evaluation & Assessment in Software Engineering (EASE 2012), pp. 228 – 232 (2012)
14. Wake, W.C.: INVEST in Good Stories, and SMART Tasks. XP123, http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/ (2003)