User Story Writing in Crowd Requirements Engineering: The Case of a Web Application for Sports Tournament Planning

Abel Menkveld*†, Sjaak Brinkkemper*, and Fabiano Dalpiaz*
*Utrecht University, Utrecht, The Netherlands
{s.brinkkemper, f.dalpiaz}@uu.nl
†Tournify, Amsterdam, The Netherlands
abel@tournify.nl

***Abstract*—Although users feel more engaged when they are involved in the elicitation, negotiation and prioritization of requirements for a product or service they are using, the quality of crowdsourced requirements remains an issue. Simple notations like user stories have been highly adopted by practitioners in agile development to capture requirements for a software product, but their utilization in crowdsourced requirements engineering is still scarce. Through a case study of a web application for sports tournament planning, we investigate how a dedicated platform for user story writing in crowd requirements engineering is valued by its users and we show that the delivered requirements are not inferior to those written by professionals.**

***Index Terms*—CrowdRE, user stories, requirements engineering, case study.**

I. Introduction

The process of extracting informal stakeholders' needs and translating them into formal specifications is a key process in Requirements Engineering (RE). These requirements are used as an input for software development. More specifically, they serve as the basis for project planning, risk management, trade-off analysis, acceptance testing, and change control [1]. Clear statements of requirements are one of the project's success factors, but at the same time incomplete requirements are an important reason why projects are impaired [1].

Together with the shift from traditional (waterfall) development to agile software development, the RE processes is changing accordingly. This is necessary because traditional requirement activities – elicitation, analysis and negotiation, documentation, validation, and management – do not take the iterative processes of agile software development into account. However, agile RE does not only alleviate challenges of traditional RE, but also poses new ones. Minimal documentation, customer inability, and time estimation are reported as some of the challenges of agile RE [2]. A proper use of agile RE artifacts is necessary to overcome these problems [3].

In this study, we focus on one type of RE artifacts: user stories (USs). USs are estimated to be used by over half of the practitioners in the software industry to capture requirements [4] and there is tight coupling of USs with agile methods [5]. A US is *"a description of a feature written from the perspective of the person who needs this"* [6]. The written text is a semi-structured natural language statement. The most widespread format of a US is: *"As a <role>, I want <goal>, so that <benefit>*, as used in the following example [7]:

> As an administrator, I want to receive an email when a contact form is submitted, so that I can respond.

Next to the use of simple notations like USs to capture requirements, user involvement is vital in RE. Involving users in RE can not only improve system acceptance, diminish project failure, and deliver greater system understanding by the user; it also helps to improve customer loyalty and broaden the market [8]. Therefore, crowdsourced RE has been investigated by a series of studies [9]. For example, a research group from RWTH Aachen University developed *Requirements Bazaar*, an open source web-based platform for crowd-based RE [10]. Snijders *et al.* [11] advocate Crowd-Centric RE, by combining crowdsourcing and gamification to involve users in the elicitation, negotiation and prioritization of requirements. According to the researchers, this helps fostering user involvement, is valuable in all stages of RE and gives equal priority to both customers and end users when they are not the same.

The embodiment of the Crowd-Centric RE vision is REfine, a gamified platform for eliciting and refining requirements. Dalpiaz *et al.* [8] showed in a case study how users perceived this crowdsourcing platform as more useful and more engaging compared to previous feedback experiences. However, they were worried that the quality of requirements would not match the quality resulting from experts' methods, and the requirements may not be detailed enough for a focus group or product backlog. This is an interesting observation, since one of the main incentives to involve the crowd in software development in general [12] and RE in specific [9] is to achieve higher quality. It can be a challenge or limitation at the same time [13], [14], [15], [10]. It is argued that simple notations such as USs may improve the quality of crowdsourced requirements and can therefore mitigate this risk [8], but to the best of our knowledge no study has yet been performed on US writing by crowd workers.

Therefore, in this paper we investigate how a crowdsourced

RE platform can be employed to enable crowd workers to express requirements in the form of USs. We implemented and validated the platform in the case of a web application for sports tournament planning.

The rest of the paper is organized as follows. Section II describes the case study, covering both the company and its existing practices regarding RE. Section III presents the crowd RE platform and details the evaluation protocol. We report the results from our case study in Section IV and discuss them in Section V. Finally, Section VI presents a summary and future research directions.

II. Case study: Tournify

We performed a single-case study, which involved the design and validation of a crowdsourced RE platform using a Technical Action Research approach [16].

A. The Tournify company and tournament manager

Tournify is a software development company based in Amsterdam, The Netherlands. Their services, which are provided through an online application, are targeted at sports and e-sports tournament and competition organizers. The main product is the Tournify tournament manager. This web application allows tournament organizers to manage participants, create a match schedule based on a chosen tournament format, and process the results as the tournament processes. The organizer can also use the tournament manager to create a tournament website to present the event to the audience. The athletes and supporters are able to view the schedule, results and standings by visiting this tournament website or by looking at a big screen, as new information comes in real time.

The Tournify tournament manager is written in Javascript. It uses React, an open-source JavaScript library developed by Facebook, for the creation of the interactive user interfaces. For the dynamic content, Firebase is used: a mobile and web development platform maintained by Google that allows storing and syncing data across multiple clients. The total lines of code (LOC) is near 25k and around 110 components are used. They serve over 10k registered users (tournament organizers) and host over 25k created tournaments since the website[1] became publicly available in late 2017.

B. The current elicitation process

The company receives several feature requests from its customers. In five months, 44 unique customers requested 77 new features. Most of them are requested via email (57%) or via the support chat (38%). The requesters organize tournaments in sixteen different sports and two different e-sports. Tournify can be used free of charge, which 39% of the requesters did. The other 61% of the requesters upgraded at least one tournament to one of the paid packages, a requisite to host tournaments with more than eight participating teams.

Further analysis of the requests and conversion of the texts to USs caused no difficulties in 71% of the cases: the role was clear, a goal was expressed, and the potential benefit was highlighted. In the other cases, the benefit was not explicitly mentioned. Although this benefit is optional in a US, it may provide valuable information, especially when a request comes in without any context via email or chat. Consider the following request that came in:

> Why is the number of teams I can add to a group limited? I want to place 46 teams into one group.

A corresponding US would be:

> As an organizer, I want to place 46 teams into one group.

This is a valid US but raises questions since it is unknown why one wants to place this many teams into one group: even the biggest leagues in the world have place for a maximum of 20 teams. Only after further communication between the product manager and requester, it becomes clear the user does not want to place 46 teams into one group, he wants to host a tournament with different games (currently Tournify is built to host tournaments for a single sport). Rather than focusing on making the workaround possible, this user (and most likely, many others) will receive a higher benefit by the development of a dedicated feature for hosting multi-sports tournaments.

This lack of context information is one of the reasons that makes the current workflow time-consuming for the product owner. Responding to the feature requests, even if they are clearly stated, also takes time. And as the business grows, the number of requests will increase. Another downside of the current workflow is that it does not allow for proper requirements prioritization and does only involve a small subset of the users.

III. Crowdsourced RE platform for US authoring

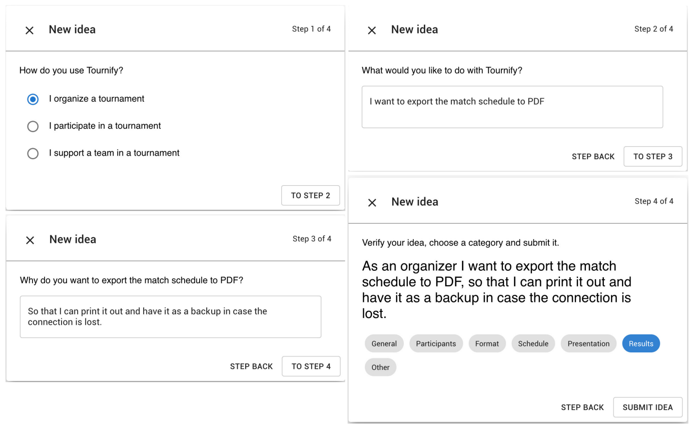
We designed a crowdsourced RE platform, integrated into Tournify, which enables users of the software application to submit feature requests in the strict format of USs: role, goal, and benefit. In order to help users formulate these stories, even if they have never seen or heard of a US before, we use a form with four simple self-explanatory and small steps (Figure 1). We deliberately decide to stop at the level of user stories, without refining them through acceptance criteria [17], to keep the crowd task simple.

Step 1: Role. The user who is requesting a feature is asked which role they play regarding the way s/he uses the application. The user can select one of the roles from the predefined options using radio buttons. In the case of Tournify, three roles are defined: organizer, participant, and supporter.

Step 2: Goal. The user is asked what s/he wants to do with Tournify: a feature that is missing. The textbox contains static text before the user input, i.e., the phrase ‘I want to’.

Step 3: Benefit. A textbox is employed here too, through which the user is asked why s/he wants to have the requested feature, to know what the user sees as the potential benefit

[1] www.tournifyapp.com


New idea — Step 1 of 4

How do you use Tournify?

- [✓ I organize a tournament]
- I participate in a tournament
- I support a team in a tournament

New idea — Step 2 of 4

What would you like to do with Tournify?

I want to export the match schedule to PDF

New idea — Step 3 of 4

Why do you want to export the match schedule to PDF?

So that I can print it out and have it as a backup in case the connection is lost.

New idea — Step 4 of 4

Verify your idea, choose a category and submit it.

As an organizer I want to export the match schedule to PDF, so that I can print it out and have it as a backup in case the connection is lost.

General · Participants · Format · Schedule · Presentation · [✓ Results] · Other

Fig. 1. The four steps to author a US regarding Tournify

when the feature would be implemented in the software application. The answer starts with 'so that'.

Step 4: Verification and category selection. Before submitting the idea, the user is able to verify the US that has been formulated based on the answers he or she provided in the first three steps. The user also has to select one of the predefined categories. These categories are part of the main menu of the application, so the users are already familiar with the terms. Labeling the requests with the corresponding category allows for easy categorization later on.

All requests are published on a feature request overview page in the Tournify web app, which can be accessed via the support menu. Besides showing the outputs of elicitation, the platform also supports a second goal: to negotiate and prioritize requirements utilizing the crowd. This is done by two simple means: voting and commenting. For the requirements prioritization, we follow the general advice of Maiden & Ncube [18], also advocated by Berander & Andrews [19], to use the simplest appropriate prioritization technique. This is especially true in crowd-centric requirement engineering [8], since end-user crowd workers are likely to be less experienced with requirement prioritization than product managers.

The prioritization technique should also allow for easy reprioritization, as requirements will be added, changed or deleted continuously. We used the *confirmation or negation* feedback type, in which users agree or disagree on problems or opinions of other users [20]. This feedback type is also used in the Requirements Bazaar [10] and REfine [14] platforms. Lastly, a commenting section enables users and product managers to respond or add suggestions to the requests.

The crowdsourcing platform was deployed and announced on February 25th, 2019. The announcement was sent via an email to a selected group of 337 users (63% of which opened the link). These users had either requested a feature in the past, subscribed to the newsletter, or made a purchase recently. A reminder was sent one month later (55% of which opened the link). The total data collection period was five weeks, so all requests submitted after March 31st, 2019 are not included in this research. Among all requesters, voters and commenters, one free tournament upgrade has been raffled. Users were also informed of the feature request platform via a snack bar message which was shown when opening the Tournify tournament planner. The researchers initiated the first request and commented on some of the requests during the study. They were also able to label features as *in development* or *done*. This first request by the researchers will be included in the report on the number of requests, because users were able to comment and vote on this request. However, it is not further evaluated regarding the quality and complexity criteria we discuss below. Comments from the researchers are excluded from the results.

Perceived Usefulness. After the data collection process, the users who submitted an idea received an email with a link to a short questionnaire. This questionnaire measures the perceived usefulness of the platform from an end user perspective through four questions that use a five-point Likert scale. Each question concerns the perceived usefulness of a functionality of the platform: requesting, viewing, voting, and commenting. One closed question is included to verify if the requester had experience with formulating USs before, and one open text field can be used to comment on the experience with the platform.

Quality. The Quality US framework [7] was used to assess the USs individually based on their syntactic, semantic and pragmatic quality. While QUS was originally proposed to assess how well the product team members would formulate user stories, we use it here to test whether external stakeholders are able to author high-quality user stories. The eight criteria and their descriptions are shown in Table I.

TABLE I
THE EIGHT CRITERIA TO ASSESS USS INDIVIDUALLY FROM THE QUALITY US FRAMEWORK [7]

Criteria	Description
	Syntactic
Well-formed	A US includes at least a role and a means
Atomic	A US expresses a requirement for exactly one feature
Minimal	A US contains nothing more than role, means, and ends
	Semantic
Conceptually sound	The means expresses a feature and the ends expresses a rationale
Problem-oriented	A US only specifies the problem, not the solution to it
Unambiguous	A US avoids terms or abstractions that lead to multiple interpretations
	Pragmatic
Full sentence	A US is a well-formed full sentence
Estimatable	A story does not denote a coarse-grained requirement that is difficult to plan and prioritize

Each US was evaluated on its quality manually by three experts individually. The experts used the description of the criteria from Table I, as well as the additional information from the accompanying article, to analyze the USs. The USs were distributed among six members of the Requirements Engineering Lab at Utrecht University. They analyzed one third of the USs each and the main author of this paper analyzed all USs. If there was no consensus in the judgement of the experts, majority voting was employed to decide.

Complexity. We made an estimation of the amount of work it would take to implement each US individually, based on the assessment of the lead developer of Tournify. For the scaling, the Fibonacci sequence (1, 2, 3, 5, 8, 13, 21) was used. We assigned a value of '0' when it concerned a feature that has already been implemented but overlooked by the requester. The other numbers represent development hours. Since it is difficult to estimate large work items with a high degree of confidence, the upper limit for our estimation was 21 hours. In practice, USs who take more than 21 hours to implement have to be broken down into smaller items.

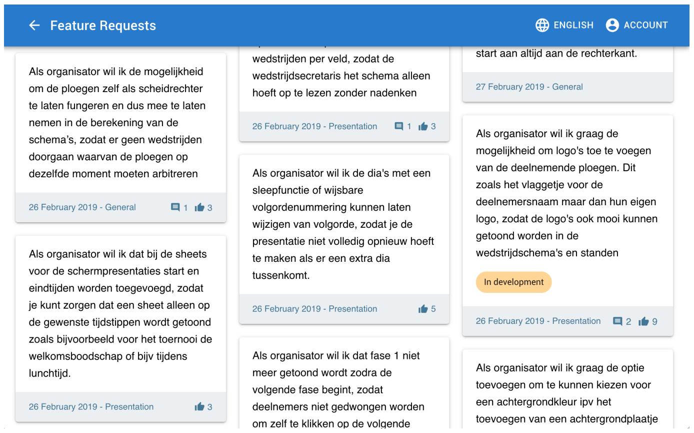
IV. Results

In the five-weeks period, 157 unique visitors accessed the feature request platform. From those visitors, 39 users interacted with the platform by submitting an idea (23), voting on an idea (28), and/or commenting on an idea (9). Together, they submitted 57 ideas, voted 89 times and commented 14 times (Table II). The functionality to downvote an idea ('I don't need this') was not used and in five times a requester voted on its own idea, which was not prevented by the platform.

TABLE II
Use of the crowdsourced RE platform

Value	Total	Unique users
Page views	247	157
Interactions	160	39
Requests	57	23
Votes	89	28
Comments	14	9

More than half of the requesters (15, 65%) submitted only one idea, two users submitted respectively two and three ideas and four users submitted five or more ideas (respectively 5, 6, 7, and 14 ideas). All ideas are written in Dutch and constructed based on the template of a US. A screenshot of part of the Feature Requests overview page is shown in Figure 2.



Als organisator wil ik de mogelijkheid om de ploegen zelf als scheidrechter te laten fungeren en dus mee te laten nemen in de berekening van de schema's, zodat er geen wedstrijden doorgaan waarvan de ploegen op dezelfde moment moeten arbitreren

26 February 2019 - General 1 3

Als organisator wil ik dat bij de sheets voor de schermpresentaties start en eindtijden worden toegevoegd, zodat je kunt zorgen dat een sheet alleen op de gewenste tijdstippen wordt getoond zoals bijvoorbeeld voor het toernooi de welkomsboodschap of bijv tijdens lunchtijd.

26 February 2019 - Presentation 3

wedstrijden per veld, zodat de wedstrijdsecretaris het schema alleen hoeft op te lezen zonder nadenken

26 February 2019 - Presentation 1 3

Als organisator wil ik de dia's met een sleepfunctie of wijsbare volgordenummering kunnen laten wijzigen van volgorde, zodat je de presentatie niet volledig opnieuw hoeft te maken als er een extra dia tussenkomt.

26 February 2019 - Presentation 5

Als organisator wil ik dat fase 1 niet meer getoond wordt zodra de volgende fase begint, zodat deelnemers niet gedwongen worden om zelf te klikken op de volgende

start aan altijd aan de rechterkant.

27 February 2019 - General

Als organisator wil ik graag de mogelijkheid om logo's toe te voegen van de deelnemende ploegen. Dit zoals het vlaggetje voor de deelnemersnaam maar dan hun eigen logo, zodat de logo's ook mooi kunnen getoond worden in de wedstrijdschema's en standen

In development

26 February 2019 - Presentation 2 9

Als organisator wil ik graag de optie toevoegen om te kunnen kiezen voor een achtergrondkleur ipv het toevoegen van een achtergrondplaatje

Fig. 2. Screenshot of the feature request overview page

Next to the feature description, each element also contains the submission date and selected category. If applicable, the element also contains the number of votes, number of comments, and development status. Two feature requests got nine upvotes, which is the most times a feature has been upvoted. The categorization of USs turned out to be a difficult task for the crowd workers, judging by the numbers. In more than half of the cases (52%), the category selection of the requester does not match the category assignment done by the first author.

After the study period, thirteen users who interacted with the crowdsourcing platform responded to the questionnaire that was sent to them via email. Most of them (10) requested a feature themselves, while the other three respondents only voted for a feature. They perceived the platform as very useful, regarding all four possible interactions when rated on a five-point Likert scale: requesting (M = 4.9; SD = 0.28), viewing (M = 4.8; SD = 0.38), voting (M = 4.5; SD = 0.88), and commenting (M = 4.5; SD = 0.66). One user who requested a feature, voted for and commented on an idea and had previous experience in writing USs commented:

> *"You implemented the agile methodology in a very fun way. In such a manner the users get involved better and at least have the feeling their opinion matters"*

Others found it *"a fantastic way to improve the application"*, *"very useful to allow users to submit ideas"* and see it as a way to *"improve the software for your own tournament"*. Another user noted how *"every user gets new ideas while using Tournify on their tournament"* and how this is *"the best feedback to improve the application"*.

Out of the people who requested a feature, 70% had never written a US before. When asked if they find it helpful to formulate the ideas as USs, compared to free text, the average score was 3.5 (SD = 0.85). There is hardly any preference to write the feature requests in free text (M = 3.2; SD = 1.14).

The results of the quality analysis are shown in Table III. In total, 52% of the USs met all requirements, meaning that 48% of the USs contained one or more easily preventable error(s). A Pearson Chi-Square test showed that there is a strong association between the minimal and full sentence criteria, which is statistically significant ($\chi^2 = 31.6$, $p < .001$). This might provide an explanation for the higher number of USs with two defects (11%), compared to those with only one defect (7%).

TABLE III
Quality of the crowdsourced USs

criterion	# USs with defect	% USs with defect
Well-formed	3	5.4
Atomic	5	8.9
Minimal	24	42.9
Conceptual	5	8.9
Problem-oriented	8	14.3
Unambiguous	9	16.0
Full sentence	19	33.9
Estimatable	3	5.4

The crowdsourced USs are evaluated based on their complexity by the developer of Tournify. Nine out of ten crowdsourced USs can be developed within one workday, according to his estimation. One US could not be estimated, because it was formulated too vaguely. Seven USs were already implemented but overlooked by the user. They are not included in the estimation shown in Figure 3, which includes 48 USs.

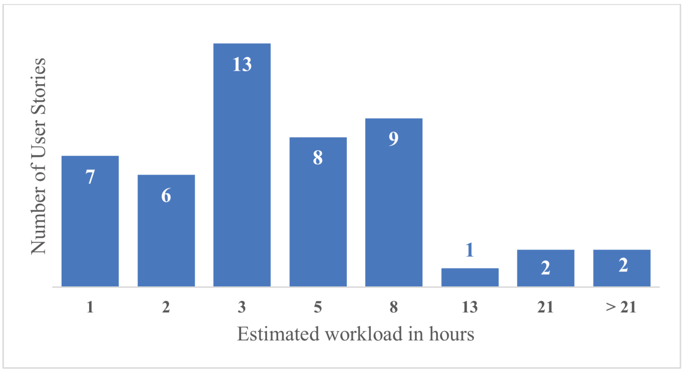


Fig. 3. Complexity of the identified USs, according to their estimated effort

V. Analysis & Discussion

Our results show that the use of crowdsourcing in RE is perceived as very useful by the end-users of a software product, while at the same time empowering the product owner with a better overview of feature requests. Commenting and voting on ideas is not only valued by the crowd workers, but may also help in the prioritization and negotiation process. Interestingly, our four-step US formulation wizard was not perceived as an extra difficulty by the users while expressing feature requests: there is hardly any preference to submit ideas in free text instead. This is promising, as research has shown how *"stakeholders enjoy working with USs, using a common template benefits RE and the simple structure of USs enables developing the right software"* [5]. Furthermore, we have found that 95% of the crowdsourced USs are both easy to estimate and easily implementable based on our quality analysis and hour estimation as done by the main developer. Almost 90% of the feature requests can even be implemented within one workday.

In terms of US quality, the most frequent occurring defects are USs violating the minimal criterion (42.9%) or not being written as one full sentence (33.9%). Lucassen *et al.* [7] tested 1,000+ USs written by professionals from different companies and found that the minimal criterion is violated in 13.3% of the cases. In our case, this defect occurs three times more often: compared to user stories written by professionals, those written by crowd members include more comments, descriptions of the expected behavior, or testing hints. In our aim, this additional information should be left to the 'comment' section of the platform. The violation of the minimal criterion is also reflected by the length of the crowdsourced USs. The *goal* is expressed in 108 characters on average and crowd workers needed 97 characters on average to formulate the *potential benefit*. When compared to 551 real-world English USs from eight different projects, retrieved from a publicly available data set [21], we found the *means* plus *end* of the Dutch crowdsourced USs (204 characters) to be over two times longer than the USs written by professionals (97 characters), which had an average *goal* description of 51 characters and *benefit* expression in 55 characters, if present. The length of the crowdsourced User Stories is similar to the length of the feature requests that were sent in by email or the support chat (192 characters) prior to the deployment of the platform. Note that we did not count the terms from the US format (*I want to* and *so that*) and did not control for the information density of the different languages the USs are written in. Moreover, 17% of the real-world USs lack a description of its *benefit*. There is also a major difference in the use of *roles*. Three roles are defined for the Tournify application (organizer, participant, supporter). All requesters indicated they are organizers, whereas the examined data set from professionals shows, on average, 12 roles.

The crowdsourced USs and the USs written by professionals show a similar number of defects regarding the well-formed criterion (5.4% crowd, 4.5% professionals) and atomic criterion (8.9% crowd, 10.3% professionals). In total, 52% of the crowdsourced USs meet all requirements, meaning that 48% of the USs contains one or more easily preventable error(s). Lucassen *et al.* [7] conclude that 56% of USs written by professionals have at least one defect as detected by their automatic testing tool. However, these results are difficult to compare as Lucassen *et al.* [7] tested against less, but different, criteria from the framework than we did.

Based on our results, we see opportunities for improving the crowdsourced RE platform to enhance the quality of the USs. Defects on the minimal and full sentence criteria can be prevented with simple means like a spelling checker and warnings when there is additional text after a dot, hyphen, semicolon, or other separating punctuation marks. Text between brackets should also trigger a warning message on the screen.

During the five-weeks testing period, 17 features were requested via email or through the support chat, bypassing the feature request platform. In most cases, those users were unaware the platform existed. When compared to the engagement prior to the deployment of the platform, while correcting for the duration of the measurement, the number of ideas that were sent per day remained unchanged. However, since the number of organizers using the service exhibited a 175% increase, we estimate the platform could save the requirements engineer circa 2 hours of work per month. This estimation is based on the 10 minutes it takes the requirements engineer to process each request, and the decrease in the number of requests via email or chat per 1,000 unique page views from 13.6 to 7.2. This little time saving may have a higher impact if the business grows in the number of customers. Nevertheless, we believe that the main incentive to employ a crowdsourced requirements platform should be to engage users and gather, prioritize and negotiate high-quality requirements.

A. Validity threats

The main threat affecting the generalizability of this study is the focus on a single case. However, generalizability was not our prime concern: this study is mainly exploratory and it constitutes one of the first attempts to let a crowd write USs for a software product they use.

The personal involvement of the first author (who is one of the co-founders of Tournify) allows for full access to the development artifacts and stakeholders, and a comprehensive knowledge of the organization and business processes. At the same time, it raises relevant questions about possible bias and prejudice. Action Research in general has been criticized for its *"lack of methodological rigor, its lack of distinction from consulting and its tendency to produce either research with little action or action with little research"* [22]. To ensure the rigor and relevance of this study, we made sure the five principles of Canonical Action Research (CAR) were taken into account at all stages of the research. This set of principles and associated criteria are developed by Davison, Martinsons & Kock in 2004, to allow for a study in which organization problems are addressed while at the same time contributing to scholarly knowledge [22]. However, it is still possible that the results are influenced by the respondents' knowledge of their participation in a study (the Hawthorne effect).

The biggest limitation is the small sample size. Over half of the crowdsourced USs is written by four users. This means that their expertise highly influenced the overall results regarding the quality evaluation, even though it is likely that also in other software products there will be a group of highly engaged users with presumably more technical expertise.

VI. Summary and Future research

This paper presented one of the first attempts (for another recent effort, see Kolpondinos and Glinz [23]) to let a crowd of external stakeholders express their requirements via the user story format. The results are positive: the participants appreciated the platform and our results did not provide reasons why crowd workers would be unable to author USs.

The work is a first step to combine user stories and crowdsourcing and paves the way for future directions. Feedback techniques can be implemented to assist the stakeholders during the authoring of USs, aiming to improve the syntactic quality. Further research should also investigate the relevance and usefulness of having a *role* in USs specified by the crowd members, and study how the identified user stories are refined and implemented. Finally, the expertise of crowd workers, in relation to their involvement with the software product, is also worth investigating.

Acknowledgment

The authors would like to thank the members of the Grimm research group for the valuable discussions about this work, Jesse Bouma for implementing the platform, and all the Tournify users who provided their feedback.

References

[1] J. Dick, E. Hull, and K. Jackson, *Requirements engineering*. Springer, 2017.

[2] I. Inayat, S. S. Salim, S. Marczak, M. Daneva, and S. Shamshirband, "A systematic literature review on agile requirements engineering practices and challenges," *Computers in Human Behavior*, vol. 51, pp. 915–929, 2015.

[3] O. Liskin, "How artifacts support and impede requirements communication," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 2015, pp. 132–147.

[4] M. Kassab, "The changing landscape of requirements engineering practices over the past decade," in *International Workshop on Empirical Requirements Engineering*. IEEE, 2015, pp. 1–8.

[5] G. Lucassen, F. Dalpiaz, J. M. E. van der Werf, and S. Brinkkemper, "The use and effectiveness of user stories in practice," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 2016, pp. 205–222.

[6] E.-M. Schön, J. Thomaschewski, and M. J. Escalona, "Agile requirements engineering: A systematic literature review," *Computer Standards & Interfaces*, vol. 49, pp. 79–91, 2017.

[7] G. Lucassen, F. Dalpiaz, J. M. E. van der Werf, and S. Brinkkemper, "Improving agile requirements: the quality user story framework and tool," *Requirements Engineering*, vol. 21, no. 3, pp. 383–403, 2016.

[8] F. Dalpiaz, R. Snijders, S. Brinkkemper, M. Hosseini, A. Shahri, and R. Ali, "Engaging the crowd of stakeholders in requirements engineering via gamification," in *Gamification*. Springer, 2017, pp. 123–135.

[9] K. Mao, L. Capra, M. Harman, and Y. Jia, "A survey of the use of crowdsourcing in software engineering," *Journal of Systems and Software*, vol. 126, pp. 57–84, 2017.

[10] D. Renzel, M. Behrendt, R. Klamma, and M. Jarke, "Requirements bazaar: Social requirements engineering for community-driven innovation," in *IEEE International Requirements Engineering Conference*, 2013, pp. 326–327.

[11] R. Snijders, F. Dalpiaz, M. Hosseini, A. Shahri, and R. Ali, "Crowd-centric requirements engineering," in *IEEE/ACM International Conference on Utility and Cloud Computing*, 2014, pp. 614–615.

[12] K.-J. Stol and B. Fitzgerald, "Two's company, three's a crowd: a case study of crowdsourcing software development," in *International Conference on Software Engineering*. ACM, 2014, pp. 187–198.

[13] M. Cohn, *User stories applied: For agile software development*. Addison-Wesley Professional, 2004.

[14] R. Snijders, F. Dalpiaz, S. Brinkkemper, M. Hosseini, R. Ali, and A. Ozum, "Refine: A gamified platform for participatory requirements engineering," in *International Workshop on Crowd-Based Requirements Engineering*. IEEE, 2015, pp. 1–6.

[15] S. L. Lim, D. Damian, and A. Finkelstein, "Stakesource 2.0: using social networks of stakeholders to identify and prioritise requirements," in *International Conference on Software Engineering*. IEEE, 2011, pp. 1022–1024.

[16] R. Wieringa and A. Moralı, "Technical action research as a validation method in information systems design science," in *Design Science Research in Information Systems. Advances in Theory and Practice*, K. Peffers, M. Rothenberger, and B. Kuechler, Eds., 2012, pp. 220–238.

[17] D. North, "Introducing BDD," *Better Software*, vol. 12, 2006.

[18] N. A. Maiden and C. Ncube, "Acquiring cots software selection requirements," *IEEE software*, vol. 15, no. 2, pp. 46–56, 1998.

[19] P. Berander and A. Andrews, "Requirements prioritization," in *Engineering and managing software requirements*. Springer, 2005, pp. 69–94.

[20] N. Sherief, W. Abdelmoez, K. Phalp, and R. Ali, "Modelling users feedback in crowd-based requirements engineering: An empirical study," in *IFIP Working Conference on The Practice of Enterprise Modeling*. Springer, 2015, pp. 174–190.

[21] F. Dalpiaz, "Requirements data sets (user stories)," Mendeley Data, v1, 2018, http://dx.doi.org/10.17632/7zbk8zsd8y.1.

[22] R. Davison, M. G. Martinsons, and N. Kock, "Principles of canonical action research," *Information Systems Journal*, vol. 14, no. 1, pp. 65–86, 2004.

[23] M. Z. Kolpondinos and M. Glinz, "Garuso: a gamification approach for involving stakeholders outside organizational reach in requirements engineering," *Requirements Engineering*, 2019.