CRAFT: A Crowd-Annotated Feedback Technique

Mahmood Hosseini[1], Eduard C. Groen[2], Alimohammad Shahri[1], Raian Ali[1]
[1]Faculty of Science and Technology, Bournemouth University, UK
[2]Fraunhofer Institute for Experimental Software Engineering, Germany
Email: [1]{mhosseini, ashahri, rali}@bournemouth.ac.uk, [2]eduard.groen@iese.fraunhofer.de

***Abstract*—The ever increasing accessibility of the web for the crowd offered by various electronic devices such as smartphones has facilitated the communication of the needs, ideas, and wishes of millions of stakeholders. To cater for the scale of this input and reduce the overhead of manual elicitation methods, data mining and text mining techniques have been utilised to automatically capture and categorise this stream of feedback, which is also used, amongst other things, by stakeholders to communicate their requirements to software developers. Such techniques, however, fall short of identifying some of the peculiarities and idiosyncrasies of the natural language that people use colloquially. This paper proposes CRAFT, a technique that utilises the power of the crowd to support richer, more powerful text mining by enabling the crowd to categorise and annotate feedback through a context menu. This, in turn, helps requirements engineers to better identify user requirements within such feedback. This paper presents the theoretical foundations as well as the initial evaluation of this crowd-based feedback annotation technique for requirements identification.**

***Keywords*-crowdsourcing; requirements elicitation; feedback categorisation; crowdsourced text mining**

I. Introduction

The advent of Web 2.0 and social media has provided millions of users and stakeholders around the world with the possibility to create their own content. These contributions also include the requirements of different stakeholders of software systems, which are normally expressed in the form of textual representations of their needs and opinions in forums, feedback platforms, etc. [1]. Typically, the users of a software system state their requirements by providing comments in a given text box on a relevant online feedback platform (in-house or public), or by using a built-in feedback mechanism within the software to voice their opinions by filling out text boxes, ticking checkboxes, etc. Text mining and data mining tools have been devised to facilitate requirements engineering processes through automated means for capturing and categorising requirements. For example, the utilisation of data mining and recommender systems has been proposed to create an open, scalable, and inclusive requirements elicitation process over a moderated requirements generation and discussion forum, which can support a large number of stakeholders [2]. Such a large group of current or potential users of a software product is commonly referred to as a "crowd" [1]. Furthermore, the use of text mining has been proposed to identify and analyse requirements on privacy protection and vulnerabilities in policy documents [3]. Moreover, the use of text mining for improving requirements specifications has been studied in [4].

Natural language, however, has numerous peculiarities that often pose a challenge for machine recognition approaches. When providing feedback, stakeholders do not typically think about how their feedback will be processed later by requirements engineers or text miners. Instead, they use the language that suits them best and the message they would like to communicate to developers and other users, and they usually do so in the writing style that is most idiosyncratic to them. As a result, apart from lack of punctuation, possible poor writing and spelling, or the use of flowery language and emoticons, user-provided feedback may be flooded with hints and cues (e.g., "this program works just like Spiderman"), rhetorical expressions (e.g., "who says this program has no flaws?"), newly-coined terms and slang or alternative spellings (e.g., "this program is supey-dupey"), and irony and sarcasm (e.g., "this program runs so smoothly and fast, my foot!") (see Fig. 1). Moreover, text mining applications become less usable when other languages are used in feedback forums (also see Fig. 1).

Fig. 1. Five exhibits showing possible deficiencies in text mining applications (courtesy of Google Play)

That the feedback language is often informal, ambiguous, or less structured, does not mean it has no value. For example, Spiderman can refer to the stealthiness and smoothness of the software; the tone and the context of "who says this program has no flaws?" suggest that the software does have bugs and flaws; "supey-dupey" can mean the software is very effective and works very well; and the sarcasm suggests the

software does not run smoothly and is very slow. But current text mining methods cannot capture these obscure intentions and hidden messages. On the contrary, these methods may even wrongly interpret users' sarcasm and irony and produce inaccurate results, and there is much work ahead before they will be able to correctly handle these language aspects.

The challenges in processing natural language and user-generated content are recognised in the literature on text mining. For example, the use of Knowledge Discovery in Databases (KDD) has been proposed along with text mining at the term level, instead of the word level, to produce more reliable results in text mining applications [5]. FrameNET[1] is a lexical database that can help a text miner understand the intended meaning from the context in which a word is placed, for example, whether "tree" refers to a living organism or a part of a diagram. Also, semantic issues in text mining have been discussed along with some approaches and methodologies in the existing literature [6]. While these methods are successful in resolving some of the aforementioned problems, they cannot fully eliminate them altogether. For example, they cannot manage newly-coined terms until their databases are updated, and they cannot cater for grammatical and lexical mistakes.

As techniques for automatically detecting the finesses of language are still at an early stage, this paper proposes the use of volunteer crowdsourcing [7] as a solution for correctly and efficiently identifying users' requirements in their feedback. Using the wisdom and power of the crowd, volunteering crowd members get the opportunity to annotate already existing feedback. Although motivating the crowd is always an issue, by deeply integrating the feedback system into a product, its requirements can be elicited by crowd members on an equal basis, from which the personal benefit will be evident. Since the crowd is generally familiar with the aforementioned textual peculiarities, it is expected that only few natural language processing problems will occur. This means that requirements are directly elicited from the masses who currently use the system-as-is, so that these can be incorporated into software evolution processes for the masses who will use the system-to-be. We have already investigated crowdsourcing and its potential in the domain of requirements engineering [8], [9]. However, although some studies have been conducted on crowdsourcing user annotation [10], little work exists to date on crowd-based requirements annotation. So far, annotation on bug reports and functional requirements [11], or on non-functional requirements [12] has been performed by experts and researchers, rather than by the end-users themselves.

The remainder of this paper is structured as follows: Section II describes our proposed CRowd-Annotated Feedback Technique (CRAFT). As a proof of concept, Section III showcases the usefulness of CRAFT in a case study. Section IV describes the challenges that CRAFT needs to address in order to obtain more reliable results. Section V concludes this paper and presents future work.

[1]https://framenet.icsi.berkeley.edu/

II. What is CRAFT?

The CRowd-Annotated Feedback Technique (CRAFT) utilises crowdsourcing as a method for harnessing the wisdom of the volunteering crowd to annotate other users' feedback. CRAFT allows requirements engineers to utilise this power instead of or in addition to automated text mining solutions, as shown in Fig. 2. In this way, crowd members can annotate any piece of feedback they want at any given time in context, and a piece of feedback can be annotated several times by several crowd members. The outcome is a list of statements that may represent a requirement expressed in user feedback. Requirements engineers can then use any aggregation method, such as averaging, to collect and analyse these annotations. We also foresee that CRAFT annotations can be contrasted with text mining classifications to reveal requirements that were missed or incorrectly categorised by the automated analysis due to the language aspects discussed above.

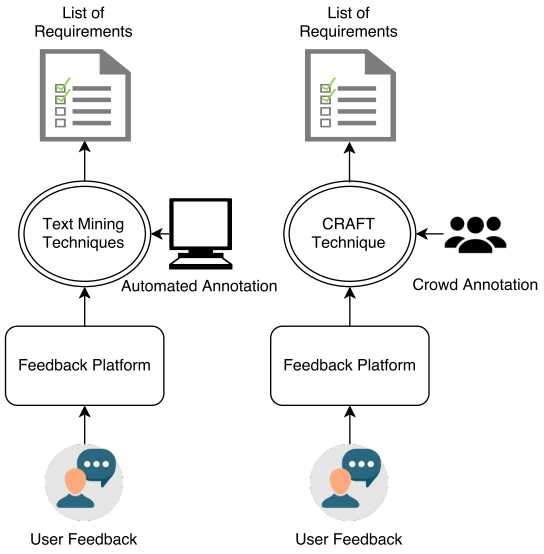

Fig. 2. Feedback annotation using text mining and using CRAFT

CRAFT allows the crowd to annotate user feedback using predefined categories, which is essential because taxonomies have already been proposed in the literature related to user feedback [13], [14]. On the other hand, CRAFT also allows crowd members to add new categories, because as time goes by, new types of requirements not yet known to requirements engineers may emerge [15].

In order to prevent a proliferation of feedback types, crowd-generated categories are not global, i.e., each crowd member can only see the categories they have personally created. This ensures that the initial list of feedback types does not get duplicate entries or becomes excessively long. It also ensures the integrity of CRAFT. New entries are only added when a requirements engineer determines that such a new category is required, either based on their own experience or based on the frequent recurrence of the new category added by several users. Only then are the new categories made available to all crowd members.

To obtain crowd annotations on users' feedback about their requirements, we suggest a three-tier design, keeping in mind the three principles of annotation: simplicity, speed, and scalability. In the first tier, crowd members specify what type of feedback they want to annotate; in the second tier, they specify the type of requirement; and the third tier is used to provide ratings and optional comments.

TABLE I
USER FEEDBACK CATEGORIES [13]

#	Topic	Description
t1	Praise	Expresses appreciation
t2	Helpfulness	Scenario the app has proven helpful for
t3	Feature information	Concrete feature or user interface
t4	Shortcoming	Concrete aspect, user is not happy with
t5	Bug report	Bug report or crash report
t6	Feature request	Asks for missing feature
t7	Other app	Reference to other app, e.g. for comparison
t8	Recommendation	Suggests acquisition
t9	Noise	Meaningless information
t10	Dissuasion	Advises against purchase
t11	Content request	Asks for missing content
t12	Promise	Trades a better rating for a specific improvement
t13	Question	Asks how to use specific feature
t14	Improvement request	Requests improvement (e.g. app is slow)
t15	Dispraise	Opposite of praise
t16	Other feedback	References or answers other feedback
t17	How-to	Explains other users how to use app

A. Tier 1: High-Level Feedback Annotation

The first tier in CRAFT is based on the categories shown in Table I [13] to create new feedback-specific categories, with the possibility for the crowd to add their own local categories. Each category in tier 1 corresponds to one or several user feedback categories in Table I and has some overlaps, as shown in parentheses in each category below. Moreover, the categories are not mutually exclusive:

1) **Requesting a new functional requirement**, e.g., adding an info button to a dialogue box in the software system (t6, t8, t11)
2) **Suggesting the omission of an existing functional requirement**, e.g., removing the option for data backup on floppy disks from the software system (t8)
3) **Reporting a bug**, e.g., unexpected exit upon clicking on an option during the execution of the software system (t4, t5)
4) **Reporting hardware-related feedback**, e.g., RAM usage or CPU usage of the software system (t3, t14)
5) **Reporting users' opinions on non-functional requirements**, e.g., the accuracy or reliability of the software system (t3, t8, t14, t15)
6) **Reporting users' feelings**, e.g., how much a user likes a feature of the software system (t1, t2, t7, t10, t12)
7) **Exchanging information on software usage**, e.g., how the in-app purchase works, and reporting "unicorns", fake reviews, or incomprehensible text (t13, t17)

A mock-up containing the categories from the first tier used in CRAFT is shown in Fig. 3 (left).

B. Tier 2: Low-Level Feedback Annotation

The second tier in the design of the CRAFT technique deals with the subcategories of every given category. These subcategories are predefined (and by no means inclusive), and like tier 1 provide crowd members with the possibility to add their own local subcategories. The subcategories as defined in the following are indicative only:

1) **Requesting a new functional requirement** subcategories can include requesting a new program functionality or requesting a new UI functionality
2) **Suggesting the omission of an existing functional requirement** subcategories can include omitting an existing program functionality or omitting an existing UI functionality
3) **Reporting a bug** subcategories can include UI bugs, error handling bugs, boundary-related bugs, calculation bugs, control flow bugs, data interpretation bugs, or hardware bugs (as shown demonstratively in Fig. 3)
4) **Reporting hardware-related feedback** subcategories can include reports related to the CPU, RAM, storage, monitor, input devices, peripherals, external ports, or incompatibilities
5) **Reporting users' opinions on non-functional requirements** subcategories can include speed, accuracy, reliability, accessibility, portability, usability, privacy, security, stability, safety, interoperability, and transparency
6) **Reporting users' feelings** subcategories can include sadness, happiness, anger, or frustration
7) **Exchanging information on software usage** subcategories can include requesting information from or providing information to users or developers

A prototypical example of the second tier for the category "Reporting a bug" as used in CRAFT is shown in Fig. 3 (middle).

C. Tier 3: Feedback and Confidence Rating

The third tier in the design of the CRAFT technique deals with three final inputs from the crowd. For every feedback that the crowd members are annotating, they will have the opportunity to 1) express the level of importance, intensity, priority, or magnitude of the user feedback, 2) express the level of confidence the crowd member has in the accuracy of their annotation, and 3) add comments to their annotation. These options will help requirements engineers make their decisions regarding software evolution based on the user feedback provided in the feedback platform in combination with the crowd annotation. The three options a crowd member is provided with in tier 3 are:

- A five-star rating scale for every annotation asks the annotator to **rate the user feedback** in terms of how they perceive the user's level of strength, priority, importance, or intensity regarding the categorised item. For example, if the user feedback is about a new requirement, it

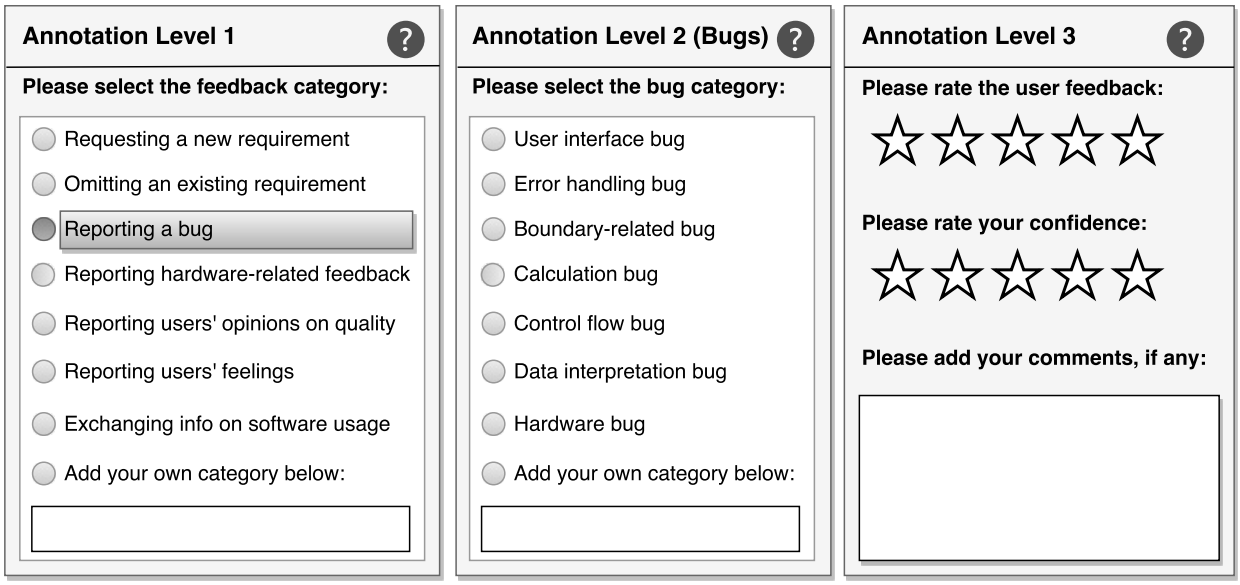

Fig. 3. Interfaces for each of the three tiers in CRAFT: (left) Tier 1: feedback category selection, (middle) Tier 2: subcategories for bug reporting, and (right) Tier 3: providing ratings and optional qualitative feedback

allows the crowd member to indicate how important they perceive that new requirement to be. In this case, one star means the user perceives the new requirement as having very low importance, and five stars means the user perceives the new requirement as having very high importance. To provide another example, if the user feedback is about a quality characteristic such as speed, this rating can mean how fast the software system is perceived by the user. In this case, one star means the user thinks the software system is very slow, and five stars means the user thinks the software system is very fast. Based on this rating scale, a requirements engineer will be able to quantify the degree of user feedback importance or intensity in any annotated feedback.

- Another five-star rating scale for every annotation asks the annotator how **confident** they are about their own annotation, ranging from one star (very unsure) to five stars (very confident). Based on this rating scale, a requirements engineer will be able to determine the strength of each annotation. This can be considered as a quality measure for every annotation.
- A free text box allows the annotator to give their final **comments** on the user-provided feedback. While the first two parts are quantitative and can be used in statistical analyses if and when needed, this part is qualitative and is mainly a way of communication between annotators and requirements engineers. This part can also be seen as a major difference between automated text mining tools, which only provide machine-generated information, and CRAFT, which produces crowd-generated comments.

Our prototypical implementation of the third tier of CRAFT is shown in Fig. 3 (right). Similar to other techniques, crowd members will need some basic training before they can efficiently use CRAFT to annotate user feedback. Using CRAFT, a crowd member can then complete the annotation with only four clicks after highlighting a particular part of a user review, provided the crowd member decides not to leave a comment.

III. Evaluation of CRAFT

To observe how crowd annotation using the CRAFT technique can actually benefit requirements engineers, we conducted an initial case study as a proof of concept. It was performed with 12 randomly selected postgraduate computer science students who responded to an open call. The participants were asked to use the CRAFT technique to annotate a collection of eight feedback statements on a mobile application on Google Play.

The feedback page provided to the participants, along with their annotations, is shown in Fig. 4. The participants were asked to identify requirements-related statements in user feedback and report on them. Because of page limitations, we only report on the mode of the participants' annotations in all three design tiers. The results are reported close to each annotated box in Fig. 4 and tabulated in Table II. The two numbers in each report represent the tier 3 ratings of the participants' perception of user feedback intensity or importance and their confidence in their annotation, respectively. In a real-life context, the overlap between crowd members' annotations will allow for statistical analyses to weigh the annotations.

As Table II illustrates, the participants were able to identify several requirements in user feedback statements. Although a few participants made mistakes in their categorisation, the results of the feedback categorisation were satisfactory overall. The participants mainly rated the intensity (or importance) of the user feedback as medium to high (3, 4, or 5 stars), and had high confidence in annotating the user feedback statements (4 or 5 stars).

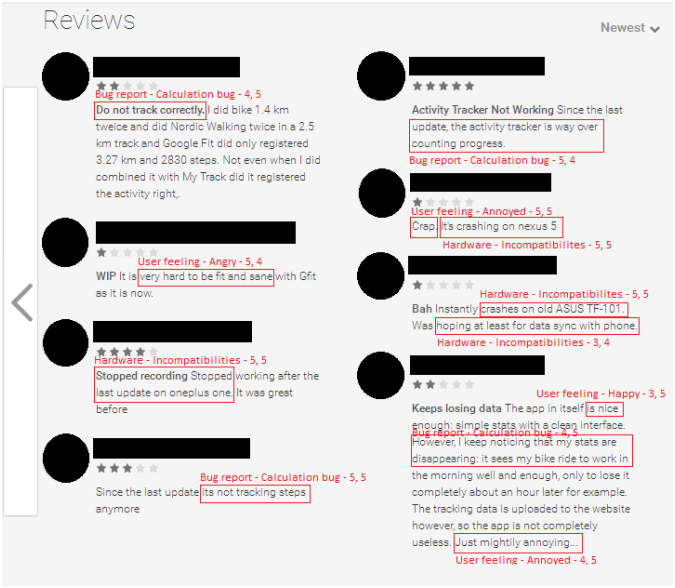

Fig. 4. The crowd annotation results

It was easy for the participants to pinpoint sarcasm, such as "it is very hard to be fit and sane with Gfit as it is now". User feelings were also easily spotted by the participants in everyday expressions such as "crap". The participants could also identify technical feedback statements, such as hardware incompatibilities, relatively well.

We acknowledge that this is work in progress and the concept needs fine-tuning and improvement. But the overall results suggest that when designed carefully, CRAFT can be an efficient technique for managing the peculiarities of natural language and deriving requirements from user feedback.

IV. Challenges

The CRAFT technique helps requirements engineers to obtain more reliable and accurate results when analysing user feedback, and can lead to cost savings for businesses conducting such analyses. However, we have identified four challenges that must be addressed before CRAFT can be implemented efficiently .

TABLE II
Crowd annotations on user feedback

Feedback Category	Feedback Subcategory	Feedback Intensity Mode	Annotation Certainty Mode
Bug report	Calculation bug	4	5
User feeling	Angry	5	4
Hardware	Incompatibilities	5	5
Bug report	Calculation bug	5	5
Bug report	Calculation bug	5	4
User feeling	Annoyed	5	5
Hardware	Incompatibilities	5	5
Hardware	Incompatibilities	5	5
Hardware	Incompatibilities	5	5
Hardware	Incompatibilities	3	4
User feeling	Happy	3	5
Bug report	Calculation bug	4	5
User feeling	Annoyed	4	5

The quality of the obtained annotations. In our case study, all participants were genuinely interested in contributing to our study. But in the real world, this is not always the case. In fact, one of the most cited issues with crowdsourcing in the literature is the issue of the quality of the obtained results [16], [17]. Several studies have been conducted to assess and increase the quality of results obtained from the crowd [18], [19]. Apart from such assessments, another main quality control routine suggested in the literature is to attract the right crowd with the right incentives [20], [21]. In CRAFT, it is easier to spot and remove low-quality results because a piece of user feedback can be annotated by several crowd members, which can help identify the correct category by popular vote.

Retention of crowd annotators. Studies suggest that maintaining the crowd's involvement in crowdsourcing platforms can be a big challenge, and that even money cannot guarantee long-term engagement with such platforms [22]. To mitigate this challenge, the CRAFT technique proposes an easy-to-use, simple, and quick annotation process by breaking the annotation task into several click-through micro-tasks to keep the crowd interested and motivated. Another proposed solution is to use digital motivation techniques such as gamification, which is defined as the use of game elements in non-game contexts [23]. Several studies have shown that gamification, if properly applied, has the potential to improve user engagement and retention in general environments such as the cloud [24], as well as in crowdsourcing environments [25].

Reliable requirements identification. As potentially anyone could be a crowd member, their level of familiarity with requirements should be considered. This CRAFT-specific challenge can also negatively affect the quality of the annotated feedback. There are several ways to mitigate this challenge. The use of a crowd-friendly language, i.e., a language that crowd members can easily relate to, is essential to minimise the training of the crowd. In CRAFT, the repetitive nature of the requirements annotation task can positively affect the learning curve [26], leading to more familiarity and less cognitive load on crowd members over time. Tooltips can also be used to explain each category in order to facilitate the crowd members' understanding. The reliability of the classification should furthermore be established by comparing the performance of the crowd against that of automatic classifiers.

Proliferation of the lists of feedback categories and subcategories. Crowd members can add their own categories to the list, which potentially can cause the list to become quite long and deflect them from the category taxonomy provided in CRAFT. Therefore, a cautious approach should be taken to managing the list of crowd-generated feedback categories. On the other hand, this challenge can turn into an opportunity with the emergence of new, unprecedented requirements, which crowd members can bring out by devising new categories. At the same time, the categories may also differ in different platforms over time depending on the software application's domain and the purpose of the annotation.

V. Conclusion and Future Work

This paper proposes a new technique for feedback annotation, called CRAFT, empowered by harnessing the power and wisdom of the crowd. CRAFT addresses some of the challenges in existing text mining applications in dealing with the peculiarities of natural language processing. The initial results of a case study suggest that CRAFT has the potential to be an effective means for feedback annotation in place of or in addition to text mining applications to reduce monetary and time-related costs of user feedback analysis.

Our future work will include creating a plugin or an application for the CRAFT technique following the gamification design principles. This plugin will then be used in real-world feedback forums to obtain crowd feedback annotations, based on which a systematic comparison to automated classification using text mining techniques will also become possible. This will allow for a thorough evaluation of CRAFT's abilities and limitations. Another part of future work will include enriching the feedback categories by assembling user-generated categories and shaping them into well-defined categories of feedback. This will give requirements engineers a powerful taxonomy of user-perceived feedback categories.

Furthermore, the identification of a crowd-friendly language to be used in CRAFT can be studied so that crowd members will have no difficulty in understanding each item listed in the feedback categories. This might include several representations (e.g., synonyms, translations, media formats) of the same feedback category item so that it suits the cognitive abilities of crowd members. Finally, triangulation between the results obtained from text mining methods and our proposed technique can be an aspect of future work. Such triangulation can help requirements engineers detect potential limitations in automated interpretations of text mining methods, while also enabling the detection of anomalies in user input and potential user clustering or ignored minorities. Moreover, to keep CRAFT scalable, it is also possible to present only those portions of the text that have not been classified with a high degree of confidence to crowd members in order to optimally benefit from the crowd's annotation efforts.

References

[1] E. C. Groen, N. Seyff, R. Ali, F. Dalpiaz, J. Doerr, E. Guzman, M. Hosseini, J. Marco, M. Oriol, A. Perini *et al.*, "The crowd in requirements engineering: The landscape and challenges," *IEEE software*, vol. 34, no. 2, pp. 44–52, 2017.

[2] C. Castro-Herrera, C. Duan, J. Cleland-Huang, and B. Mobasher, "Using data mining and recommender systems to facilitate large-scale, open, and inclusive requirements elicitation processes," in *Proceedings of the 16th RE Conference*, 2008, pp. 165–168.

[3] A. Massey, J. Eisenstein, A. Anton, and P. Swire, "Automated text mining for requirements analysis of policy documents," in *Proceedings of the 21st RE Conference*, 2013, pp. 4–13.

[4] B. Sateli, E. Angius, S. S. Rajivelu, and R. Witte, "Can text mining assistants help to improve requirements specifications," in *Proceedings of the Mining Unstructured Data (MUD)*, 2012.

[5] R. Feldman, M. Fresko, Y. Kinar, Y. Lindell, O. Liphstat, M. Rajman, Y. Schler, and O. Zamir, "Text mining at the term level," in *Principles of Data Mining and Knowledge Discovery*. Springer Berlin Heidelberg, 1998, pp. 65–73.

[6] A. Stavrianou, P. Andritsos, and N. Nicoloyannis, "Overview and semantic issues of text mining," *SIGMOD Record*, vol. 36, no. 3, pp. 23–34, 2007.

[7] M. Hosseini, K. Phalp, J. Taylor, and R. Ali, "The four pillars of crowdsourcing: A reference model," in *Proceedings of IEEE 8th RCIS Conference*, 2014, pp. 1–12.

[8] ——, "Towards crowdsourcing for requirements engineering," in *Proceedings of the 20th REFSQ Conference - Empirical Track*, 2014.

[9] M. Hosseini, A. Shahri, K. Phalp, J. Taylor, R. Ali, and F. Dalpiaz, "Configuring crowdsourcing for requirements elicitation," in *Proceedings of IEEE 9th RCIS Conference*. IEEE, 2015, pp. 133–138.

[10] P.-Y. Hsueh, P. Melville, and V. Sindhwani, "Data quality from crowdsourcing: a study of annotation selection criteria," in *Proceedings of the NAACL HLT 2009 workshop on active learning for natural language processing (ALNLP)*, 2009, pp. 27–35.

[11] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? on automatically classifying app reviews," in *Proceedings of the 23rd RE Conference*, 2015, pp. 116–125.

[12] E. C. Groen, S. Kopczynska, M. P. Hauer, T. D. Krafft, and J. Doerr, "Users - the hidden software product quality experts? a study on how app users report quality aspects in online reviews," in *Proceedings of the 25th RE Conference*, 2017 (in press).

[13] D. Pagano and W. Maalej, "User feedback in the appstore: An empirical study," in *Proceedings of the 21st RE Conference*, 2013, pp. 125–134.

[14] N. Sherief, W. Abdelmoez, K. Phalp, and R. Ali, "Modelling users feedback in crowd-based requirements engineering: An empirical study," in *8th IFIP WG 8.1 working conference on the Practice of Enterprise Modelling (PoEM)*. Springer, 2015, pp. 174–190.

[15] S. D. Harker, K. D. Eason, and J. E. Dobson, "The change and evolution of requirements as a challenge to the practice of software engineering," in *Proceedings of IEEE International Symposium on RE*, 1993, pp. 266–272.

[16] G. Kazai, "In search of quality in crowdsourcing for search engine evaluation," in *Advances in information retrieval*. Springer, 2011, pp. 165–176.

[17] A. J. Mashhadi and L. Capra, "Quality control for real-time ubiquitous crowdsourcing," in *Proceedings of the 2nd international workshop on Ubiquitous crowdsouring*, 2011, pp. 5–8.

[18] A. Aker, M. El-Haj, M.-D. Albakour, U. Kruschwitz *et al.*, "Assessing crowdsourcing quality through objective tasks." in *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC)*, 2012, pp. 1456–1461.

[19] M. Allahbakhsh, B. Benatallah, A. Ignjatovic, H. R. Motahari-Nezhad, E. Bertino, and S. Dustdar, "Quality control in crowdsourcing systems: Issues and directions," *IEEE Internet Computing*, vol. 17, no. 2, pp. 76–81, 2013.

[20] A. Bozzon, M. Brambilla, S. Ceri, M. Silvestri, and G. Vesci, "Choosing the right crowd: expert finding in social networks," in *Proceedings of the 16th International Conference on Extending Database Technology (EDBT)*, 2013, pp. 637–648.

[21] L. Erickson, I. Petrick, and E. Trauth, "Hanging with the right crowd: Matching crowdsourcing need to crowd characteristics." in *Proceedings of the 18th Americas Conference on Information Systems (AMCIS)*, 2012.

[22] C. Puah, A. Z. A. Bakar, and C. W. Ching, "Strategies for community based crowdsourcing," in *Proceedings of the International Conference on Research and Innovation in Information Systems (ICRIIS)*, 2011, pp. 1–4.

[23] S. Deterding, R. Khaled, L. E. Nacke, and D. Dixon, "Gamification: Toward a definition," in *Proceedings of CHI 2011 Gamification Workshop*, 2011, pp. 12–15.

[24] A. Shahri, M. Hosseini, R. Ali, and F. Dalpiaz, "Gamification for volunteer cloud computing," in *Proceedings of the 2nd International CGCloud Workshop, Co-located with UCC 2014*, 2014.

[25] C. Eickhoff, C. G. Harris, A. P. de Vries, and P. Srinivasan, "Quality through flow and immersion: gamifying crowdsourced relevance assessments," in *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, 2012, pp. 871–880.

[26] M. J. Anzanello and F. S. Fogliatto, "Learning curve models and applications: Literature review and research directions," *International Journal of Industrial Ergonomics*, vol. 41, no. 5, pp. 573–583, 2011.