Democratic Mass Participation of Users in Requirements Engineering?

Timo Johann and Walid Maalej
University of Hamburg
Hamburg, Germany
{johann, maalej}@informatik.uni-hamburg.de

***Abstract*—A large part of Requirements Engineering is concerned with involving system users, capturing their needs, and getting their feedback. As users are becoming more and more demanding, markets and technologies are evolving fast, and systems are getting more and more individual, a broad and systematic user involvement in Requirements Engineering is becoming more important than ever. This paper presents the idea of pushing user involvement in Requirements Engineering to its extreme by systematically delegating the responsibility for developing the requirements and deciding about future releases to the crowd of users. We summarize the pros and cons of this vision, its main challenges, and sketch promising solution concepts, which have been proposed and used in E-Participation and E-Democracy. We discussed our vision with ten experts from the fields of Requirements Engineering, politics, psychology, and market research, who were partly supportive partly skeptical.**

I. Introduction

Conventional Requirements Engineering typically involves users in developing and prioritizing requirements through interviews, surveys, or workshops. Open source projects allow users to publicly report issues and ideas through bug trackers. More recently software vendors also started involving users and collecting their feedback through social media channels. In particular, app users can easily write reviews about new app releases, report bugs, rate the app and its features, or request new features and changes [18]. Such user feedback can be used for defining the requirements of the system, solving conflicts, and planning future releases.

Research has shown that user involvement is a major success factor for software projects. Bano and Zowghi identified about 290 publications that agree about the positive impact of user involvement and its relationship to system success [1]. Pagano and Bruegge [17] interviewed developers and found that user feedback contains important information to improve software quality and to identify missing features. Recent work also showed that users are willing to contribute to software projects and their feedback often contains valuable information for developers and analysts [7], [18]. In this paper, we discuss the vision of pushing user involvement to its extreme with the aim to maximize its impact. The goal is to systematically and continuously "delegate" the development, discussion, and decision-making about requirements to the crowd of users. The domains of politics and market research inspire our vision. User involvement is the foundation of democratic systems and with E-Democracy citizens are able to participate in the proposal, development, and creation of political and legal decisions using information and communications technology [9].

The contribution of this paper is twofold. First, it summarizes the main challenges, which need to be addressed for mass and systematic participation of users in Requirements Engineering and drafts a research agenda for the field. Second, the paper draws parallels between Requirements Engineering and E-Democracy discussing potentials and limitations of using democratic concepts such as delegated voting and structured refinement. We discussed our vision and the resulting challenges with ten experts from Requirements Engineering, politics, and market research (see Section VII). We conducted face-to-face interviews each lasting about 30 minutes. The paper also reflects the discussions along its sections.

The remainder of the paper is structured as follows. Section II describes the challenges for realizing a mass participation of users in Requirements Engineering. Section III describes how fundamental concepts from E-Democracy, in particular Liquid Democracy [19] can help realizing the vision. In Section IV we introduce a combination of Liquid Democracy and Requirements Engineering and how this can help to meet the challenges. Finally, Section V reviews the related work while Section VI concludes the paper with a research preview.

II. Challenges of a Mass Participation

1) **Scalability** - *Internet-Scale User Participation:* A systematic and continuous participation of a large user basis generates a large amount of unstructured data of different quality. Recent studies of user reviews in app stores showed that one single popular app can get several thousands of reviews per day [18]. Analyzing and managing an internet-scale user input requires a large amount of human resources and a sophisticated tool support.

One recurrent comment in our discussions with the experts was that current RE approaches such as interviews and workshops do not scale to the crowd of users. Interviewed politicians reported that a higher participation prevents a quick and flexible consideration of the inputs. A large mass typically leads to more idleness. Processes take longer and the management overhead becomes unproportional to the participation benefits, leading to less motivation of users and vendors.

2) ***Motivation*** - *Incentives for Vivid Participations:* Motivating people to qualitatively contribute to tasks for which they

978-1-4673-6905-3/15/$31.00 © 2015 IEEE

RE 2015, Ottawa, ON, Canada
RE:Next! Track

Authorized licensed use limited to: National University Fast. Downloaded on September 24,2020 at 06:44:03 UTC from IEEE Xplore. Restrictions apply.

are not accountable can be very hard [2], [6]. Typical users neither have contractual obligations nor ethical commitments to contribute. It is easier to complain and be destructive than to be constructive [18]. One expert stated: "Who wants change? Everybody! Who wants *to* change? Nobody!"

Incentives for motivating users to contribute to Requirements Engineering can be extrinsic or intrinsic [21]. Extrinsic motivation includes rewarding users with cash, discounts (e.g. in the software to which they contribute), or exclusive access (e.g. to advanced features). One expert stated: "In crowdsourcing they pay money to people. In this case money also matters". One problem with rewards is that it might corrupt users and bias their input.

In contrast, with intrinsic incentives users contribute for their own sake. Research has shown that one of the most important motivational factors for users to contribute to social platforms is to gain public attention [14]. In politics, transparency is a major factor for trust and participation. Thus, recognizing and publicly honoring contributions as well as a continuous bidirectional feedback with users are important to promote intrinsic motivation. One expert stated: "Users don't give feedback because they don't know what happens to it".

Several experts mentioned gamification as a potential approach to motivate for continuous and high quality contributions. User involvements would then be designed and integrated into the requirements process as a game, where users earn "credits" the more and the better they get involved. Recent research found, however, that the success of gamification is rather limited since game concepts wear out and must be continuously renewed. A gamification approach usually works only for a specific target group [10].

3) ***Conflicts*** *- Resolving Unavoidable Conflicts:* Negotiating requirements and tradeoffs is difficult even with a small group of stakeholders [4]. Typically in conventional – both open source and commercial – software project settings, a few people will take the final decisions based on a hierarchy that is recognized by stakeholders. In companies, when negotiation techniques fail there are authorities with decisive power to resolve conflicts. In open source, decisions – e.g. about the scope and priorities – are typically made by a small number of core developers [24].

With a large heterogeneous mass of users, conflicts become the standard rather than the exception. While some users want, e.g., a red color for the interface, others might prefer blue. If users feel their input is ignored they might end up frustrated and stop participating. For example, Wikipedia users often criticize that debates are won by stamina [16]. Editors who care more and argue longer and "louder" tend to get their way. Lanier describes this phenomenon as Digital Maoism. That is, collective authorship tends to produce mainstream beliefs with a false sense of authority behind information [13].

4) ***Representativeness*** *- "Equally" Involving User Groups:* Users are different. They work differently, in different contexts, and have different needs. If a particular group of users is over-represented (e.g., having a specific platform or with specific skills), the participation results might get skewed. In politics, the analogous problem is a low turnout for votes. If a whole group does not participate in a vote, the distribution of the results will not match the actual population distribution. This phenomenon is called a *dictatorship of the activists*. Similarly, in open source communities a *benevolent dictator for life*[1] refers to a person or a group with the final say in a discussion (usually the community founder). This often leads to community fragmentation, e.g. through project forks[2].

For avoiding dictatorships of activists we need to motivate all groups to participate and ensure a representativeness of the participating sample to the whole user population. Moreover, as Requirements Engineering involves other stakeholders beside users (e.g. clients and developers), it seems natural that some stakeholders or stakeholder groups are more important than others. This is to consider when balancing the representativeness.

In our interviews Requirements Engineering experts were concerned about achieving high participation rates for all users, while the interviewed politicians considered the *representativeness* of participating users and their biasless more crucial. One expert stated: "The task of politics is to provide resources and instruments for participation. But we cannot enforce the use of these options." Another expert said: "It is difficult to ensure that the small group of people who actively write reviews is representative of the user community at large".

5) **Subjectiveness** *- Correcting Self Assessment Bias:* User contributions are not necessarily based on rationale and logical reasoning, in particular since users are not trained like managers, analysts, and developers. Users might resist to changes of software even if the advantages are obvious. In human computer interaction this phenomenon is called Baby Duck Syndrome [22]. Baby ducks imprint on the first entity they are exposed and treat it as their mother. Users might behave in a similar way, judging ideas and innovations by comparing them to what they know.

Moreover, it is often difficult for users to evaluate themselves, something known as cognitive bias in psychology. This may lead to perceptual distortion, inaccurate judgment, or illogical interpretation [12]. Users might overrate their own abilities. This pattern was observed in diverse studies while reading comprehension, operating a motor vehicle, and playing chess or tennis. The higher the ignorance, the higher is the self-confidence. This concern has been confirmed by most interviewed experts, who argued that the mainstream of users "cannot tell what they want" and that "we have to educate them first [...] and tell them what a requirement actually means".

6) ***Misuse*** *- False Users, Data Security, and Privacy:* An important challenge for a systematic user involvement is to ensure the protection of vendors' information and the privacy of the user's data. Transparency in the contribution means that the access to project data is public at least to contributing users. Unlike in politics, software organizations are competing with

[1] http://www.artima.com/weblogs/viewpost.jsp?thread=235725

[2] http://catb.org/~esr/writings/homesteading/homesteading/ar01s16.html

Authorized licensed use limited to: National University Fast. Downloaded on September 24,2020 at 06:44:03 UTC from IEEE Xplore. Restrictions apply.

others. Opening the requirements processes with requirements knowledge can be conflicting with business goals.

Similarly, contributing users might share their data, preferences, or contexts to argue for and explain their opinions and requests. In particular if user data is representing a minority, their data might become sensitive and easily misused.

Finally, it is important to ensure the authenticity of the participation. In social media, e.g., votes can simply be bought or fake. This phenomenon can become risky since requirements decisions are crucial to software's success.

III. The E-Democracy Approach

Democracy is one of the oldest participatory concepts. E-Democracy uses Internet and information technology to promote democracy. We think that concepts from E-Democracy, when combined with social media, are highly promising to partly address the challenges of a massive and continuous user participation in Requirements Engineering.

Liquid Democracy is a participative democracy approach, being a hybrid form of direct democracy and representative democracy. Both democracy forms empower people. The difference is that in direct democracy people directly decide about all policy initiatives, while in representative democracy, people elect representatives for a given period of time and empower them to decide about policy initiatives. Most modern democracies have a representative form but allow, in some cases, actions that provide a limited form of direct democracy. In Liquid Democracy, the term liquid means that the boundaries between direct and indirect democracy become blurred. In the literature, it is hard to identify a single original reference to Liquid Democracy as discussion can be traced back hundreds of years. However, Paulin [19] and Jabbusch [11] (in German) gave a good summary of the history of Liquid Democracy and its application in online systems. We introduce the key concepts of Liquid Democracy and discuss how the Software and Requirements Engineering community can adapt them. Many political parties such as the German Pirate Party use these concepts and tools that implement them.

A. Structured Collaborative Decision

Similar to issue trackers, user feedback systems, and review systems, users are able to actively submit proposals, issues, or ideas at any time – no matter whether these are concerned with software features, technical issues, tasks, priorities, or methodologies. Proposals can be made by everyone and do not have to be fully developed ideas. After submission, they undergo a discussion and refinement process, to which everyone is eligible to contribute. Such collaborative processes are also common in social media, e.g. in Wikis or in Stack Exchange. At the end of the discussion and refinement, users can vote for or against the proposal, no matter whether they have contributed to it or not.

Liquid Democracy often employs collaborative and structured decision-making processes to allow a collective drafting while avoiding unsound, dubious or low quality input. However, no standard process for the discussion and refinement of proposals and ideas exists. Some tools that implement Liquid Democracy allow for unlimited discussions without imposing an endpoint, e.g. *Votorola*. Others have fixed dates and ballots and a proposal ends with a final decision. Some approaches such as LiquidFeedback [3] have a freeze phase between the discussion and the hold of the ballot. This might be interesting for Requirements Engineering as some users might not follow the discussion and might be only interested in the results.

An interesting democratic concept is the use of *quorums*. A quorum is the minimum number of "members" that is necessary for a proposal or an idea to be discussed and qualify for a decision or vote. In politics quorums are used to ensure that a ballot with low participation will not result in unrepresentative majorities. In a software project a quorum can help to reduce low quality or irrelevant contributions.

With such a process, a user of a restaurant finder app might, e.g., propose to add a feature for searching restaurants with accessibility for wheelchairs. The user has to wait for others to support this proposal, which will probably be feasible as the idea is sound. After reaching a certain quorum, the discussion of the details can start and interested users will collaboratively develop the feature and its requirements. When the discussion ends, users can vote for or against it regardless whether they actively participated in the discussion or not.

B. Delegated Voting

Delegated voting extends conventional voting. Every user is able to vote for or against every proposal. Votes count equally, but users can accumulate votes by deciding *for* others who have empowered them. The delegation of a vote is not bound by a specific period of time and can be withdrawn at any time. Proposals and ideas can be categorized by topics, types of requirements, or system components. Users can decide for which proposal or topic they delegate their votes and to whom.

Let's assume one topic is the user interface of a system. Within this topic users submitted different ideas and comments. A specific user, say Alice, might be more interested or more knowledgeable in other topics such as privacy and data protection. Alice can delegate her votes for the subject user interface to Bob since she trusts him based on previous interactions with him. Bob's votes now have a higher weight within this subject. Alice can still take back her delegated vote for any specific proposal within the subject and vote herself, e.g. if she disagrees on a specific proposal. With the possibility to give and get back votes, the decision process becomes "liquid". Users can delegate or change their delegated votes at any time and at the levels of topic or proposal.

IV. Discussion

We argue how Liquid Democracy could help to realize mass user participation, discuss the applicability of such a paradigm in RE, and sketch main realization directions.

A. Liquid RE for Addressing Mass Participation Challenges

Liquid Democracy is a promising approach that has been developed and successfully applied for collective decision-making within political parties and in companies [11], [19].

Authorized licensed use limited to: National University Fast. Downloaded on September 24,2020 at 06:44:03 UTC from IEEE Xplore. Restrictions apply.

Tools that implement Liquid Democracy are Adhocracy (liqd.net), Liquidizer (liquidizer.github.io), LiquidFeedback (liquidfeedback.org), and Votorola (zelea.com/project/votorola/home.html). Analogously, Liquid Democracy concepts can be used to partly solve the challenges of implementing a mass participation in RE (described in Section II) – towards a *Liquid Requirements Engineering*. Table I summarizes the following discussion.

TABLE I
LIQUID RE FOR ADDRESSING MASS PARTICIPATION CHALLENGES

Challenge	Collaborative Decision	Delegated Voting
Scalability	★★☆	★☆☆
Motivation	★★☆	★☆☆
Conflicts	★★★	★★☆
Representativeness	★☆☆	★★☆
Subjectiveness	★★☆	★★★
Misuse	★★☆	☆☆☆

1) **Scalability:** Liquid Democracy allows for a mass participation of users while outbalancing the scalability of the process. The community serves as a valuable resource to manage the participation process. Users manage themselves through votes and coordinated discussions. Similarly, the Apple App Store and the popular Q&A website Stack Overflow enable their users to rate the helpfulness of others' posts. This ranks the posts and identifies the most relevant ones.

A structured, collaborative decision process supports the monitoring, filtering, and refining of user input. Users perform the filtering and the refinement through votes and discussions. E-Democracy tools enable to monitor the activities and track changes. Delegated voting minimizes the users' workload since they only participate to the extent they want to.

Nevertheless duplicates, or irrelevant contributions and ideas might overcharge the community. Additional automated approaches for automated assessment and preprocessing of contributions would help users as well as analysts, developers, and managers to overview the community and contributions. Such automated tools might help comparing user input, assessing its quality, and assigning it to a specific group of requirements, topic, or a system component, e.g. by using machine learning, data mining, and natural language processing techniques [8].

2) **Motivation:** Liquid RE will, as in politics, not solve the problem of unmotivated users. However, it can establish an inclusive, transparent, fair, and easy-to-use environment that motivates committed users to contribute. It might converge to a platform for gathering high quality input by active users.

A true involvement of users means that internal project processes need to become transparent. A Liquid RE platform can equally be used by all stakeholders. Within such a platform, discussions – at least those that are related to requirements and the users – will be conducted publicly. Users can get feedback and follow what happens to their input at any time. Users become a real part of the project and the "community". The opinions of users are recognized and discussed by users, developers, and other stakeholders. All these represent intrinsic incentives for motivating constructive user contributions. Finally, users who are uninterested in a certain user proposal or a topic can delegate their voices to other concerned users of their choice – and possibly enjoy the influence. Being aware of the power of their voices and that even a small effort can make a change is an important participation incentive. Nevertheless, additional incentives such as monetary rewards, game badges, or special recognition in the software itself might lead to a higher motivation of (other) users and reach a broad and representative participation.

3) **Conflicts:** A major benefit of Liquid Democracy is the possibility to resolve conflicts in a collaborative and fair way. Delegated voting allows users with limited time or interest to delegate their votes. This ensures that not only the voices of a few core participants who "scream loudest" are recognized. The recursive delegation also creates an implicit hierarchy that typically has the final say in classical conflict situations.

The structured decision-making avoids exhausting circular discussions. Setting a time frame to take a decision is important for Requirements Engineering, as time is critical. The freeze phases allow rethinking contributions aside from emotional discussions. Users that do not agree with a proposal are not forced to discuss it and can propose a different one instead. This can mediate a debate and lead to different proposals. Depending on the implementation of delegated voting, it is also possible to vote for favorites and alternatives so that different proposals do not get lost.

However, other than in politics, it will be hard to decide how to delegate votes to unknown users. Users – and other stakeholders as well – must build up a reputation and trust first. In politics this happens outside of E-Democracy platforms.

4) **Representativeness:** Mass participation brings an inherent risk to exclude minorities. As for every democratic participatory system, the challenge is to ensure the inclusion and protection of minority groups – keeping a balance between the "mainstream requirements" and those of the outliers.

At the same time, Liquid Democracy can give mavericks a platform to promote new and innovative ideas. It enables underrepresented groups or "outlier" users to have a say. Users get a power to influence decisions about the software even with low participation effort (i.e. one click for a vote).

Further, minorities become stronger through delegation. In politics, the delegated representatives can pick up minorities' wishes that otherwise wouldn't gain importance.

5) **Subjectiveness:** Conservative users can hinder innovation, as some experts highlighted. Experienced users, e.g., originally complained about the change in the user interface of Microsoft Office 2007 (use of ribbons)[3]. Today ribbons are widely recognized and used in other tools as well.

Liquid Democracy can help addressing the problem of user subjectiveness and of communicating changes to users. Open

[3]redmondmag.com/articles/2007/10/01/word-2007-not-exactly-a-musthave

Authorized licensed use limited to: National University Fast. Downloaded on September 24,2020 at 06:44:03 UTC from IEEE Xplore. Restrictions apply.

discussions of innovations and ideas can lead to more objective views. Users themselves will convince others, who can base their decisions on the input of experts or of fellow users.

Politicians reported that early information and involvement is a key factor to promote fundamental change. Liquid Democracy allows communicating and promoting such changes at an early stage and reducing users' concerns. To be applied in RE, Liquid Democracy should be extended to support experts and innovators. It might be necessary to bypass democratic concepts and to give specific users and stakeholders higher weighting without disregarding or discriminating others. The rules should be transparent. Experts and innovators can discuss ideas with users and explain decisions. Users do not have to accept innovations when they are released with the software.

6) **Misuse:** The collaborative refinement and decision-making can help to avoid misuse. An open discussion helps to ensure the participation authenticity, as it is much harder to manipulate a discussion between humans. However the voting mechanisms (quorum, delegated voting) are points of attack. Two types of manipulation are possible: i) a fake majority can *suppress* a requirement wanted by the real majority or ii) a fake majority can *push* a requirement that is unwanted by a real majority. The latter will be less problematic, since a user proposal has to undergo a whole refinement and decision process, but might become crucial if the participation is biased.

Opening internal project data to the public is crucial and risky as well. Expensive innovations from the company are less suitable to publish in an early stage. The management must outbalance to what extent the data can be disclosed. Anonymization can reduce threats to users' privacy. In politics, this is not always possible or wanted. In Requirements Engineering anonymization is more important as a user group should be targeted rather than individuals.

B. Applicability of Liquid Requirements Engineering

Software products and their development projects and processes are different and include various complex facets. One important question is whether and which participation concepts are applicable for which products and in which settings.

When discussing with the experts whether democratic concepts can foster massive user participation, the answers varied. Those who did not know about the capabilities of E-Democracy platforms were more skeptical. Those who were aware of such platforms were more confident, but also limited the possible usage to particular tasks.

All experts were convinced about the potential of the approach but stated that a meaningful use depends on the type of project or the product. They suggested that this approach would be fitting for open source projects, apps, and websites. One went further stating: "It would be suitable for the product development, but not for customized software". We think that while these settings represent "a good place to start" market trends might force other rather conservative organizations to rethink their approaches long term and include "more user involvement" as this is expected in the age of social media, open source, and high competition. Certainly, it is an important task for the RE community to discuss and evaluate mass participation approaches on different types of projects and products. One expert said: "Far more stakeholders can be involved in the RE process while the amount of overhead stays about the same, though the activities and techniques used will be different". One claimed: "Instead of transferring traditional RE activities we should develop new ideas".

We think that the complexity of Liquid Democracy platforms must be reduced or completely hidden from users to lower entry barriers. The participation should be possible in-situ [23] within the work environment of the users, e.g. in social media platforms, in the operating system, or in the app itself. Finally, we think the challenge of representativeness can be addressed by extending Liquid Requirements Engineering with *proactive participation* requests and *stratified sampling* – two concepts that are successfully used in the field of market research. This requires knowing the whole user population and establishing communication channels with the user – at least implicitly, e.g., via distribution platforms such as app stores. Finally, in Requirements Engineering it is, other than in democracy, meaningful to give different weighting of the participants' voices, according to the user group or expertise.

C. Research Directions

Liquid Democracy can not surely solve the challenges of a new kind of Requirements Engineering. But it shows that the RE community should think out of the box to accommodate the complexity and the scale of the crowd and ensure that we get their requirements and voices efficiently and precisely. We proposed considering Liquid Democracy as starting point to think about multidisciplinary concepts for realizing the vision of "requirements from the masses and requirements for the masses". To deal with the challenges of this vision we propose to investigate the following research directions.

Diversity over Majority. Majority decisions may exclude minorities. Masses might hinder innovation but amongst the masses one can find mavericks. Thus, it is important to include outliers. A highly diversified crowd shall be attempted instead of a broad mainstream. Approaches such as proactive participation and appropriate sampling should be further investigated.

Transparency over Feedback. Users are different. While some favor to be asked for their opinion others get quickly fed up. Every user should decide how far she wants to contribute. The processes and decisions must be transparent. Researchers should examine how to open up internal processes without threatening the business goals of software projects.

Appreciation over Reward. Rewards, e.g. in form of monetary or non-monetary benefits can be responsible for corruption and bias users and their input. A direct recognition of what users have achieved will influence them less. The active involvement of users in decisions together with developers and other stakeholders is motivating. Qualitative participation must be publicly honored. The only motivation within social platforms for users is to gain public attention. Testing and evaluating different incentives will be a crucial task.

Authorized licensed use limited to: National University Fast. Downloaded on September 24,2020 at 06:44:03 UTC from IEEE Xplore. Restrictions apply.

Dynamic over solid hierarchies. Crowdsourcing helps to keep knowledge about the users up to date. It is not only about what users want, but also how the requirements and preferences evolve. Competitive technology appears, trends prevail, users themselves and their perception change after a while. Thus, solid structures must be able to change dynamically over time. Future research must work out *liquid* concepts and processes that support these changes.

V. Related Work

The topic is present for over four decades and widely known realization approaches have been developed and are reflected in software development methodologies [1], [8], [23].

Concerning massive and systematic user involvement in RE there are two recent trends: direct (or targeted) and indirect (or implicit) involvement. Requirements Bazaar covers the direct involvement and is perhaps the closest to our work. In this approach users can register to issue trackers and formulate wishes, which can then be adopted and implemented by software developers [20]. The work focuses on open source projects while we discuss the potentials and risks of mass, systematic, and democratic involvement for RE in general. While extending issue trackers with social media features is promising to realize collaborative and structured refinements of user proposals, it does not address all challenges of massive user participation [5], as discussed in Section II. For an indirect, implicit user involvement researchers suggested to use machine learning, data mining, and natural language processing techniques to analyze user feedback (e.g. in app stores) [8], [18], as well as usage and context data at run time to better understand users and their needs [15]. We think that these approaches are complementary to our proposed vision and should be combined in the future.

VI. Conclusion

We discussed the vision of "requirements for the masses and requirements from the masses" with experts from RE, politics, and market research. Experts who already have experience with massive user involvement think that traditional RE approaches and tools will not work for systematic, crowd-based requirements. We revealed critical voices saying that the RE community must revisit traditional RE approaches to turn this vision into reality. Main challenges that occur when users massively participate in RE include dealing with scalability, motivation, conflicts, representativeness, subjectiveness, and misuse. We proposed using E-Democracy and Liquid Democracy concepts such as delegated voting, quorums, and a community-based management of users' contributions for dealing with these challenges. Future research can investigate for which projects a massive, democratic user participation is feasible and to which extent. Our research directions will hopefully help adjusting RE to the age of critical demanding users who expect their voices to be taken seriously.

VII. Acknowledgement

We thank the RE15 reviewers and the experts for their feedback: S. Adam & J Doerr (Fraunhofer), R. Ali (Bournemouth U.), C. Brosda (City of Hamburg), A. Hoffmann (Siemens), S. Körner (Pirate Party), K. Schneider (Leibniz U.), M. Nayebi (U. of Calgary), F. Dalpiaz (Utrecht U.), A. Neus (GfK).

References

[1] M. Bano and D. Zowghi. A systematic review on the relationship between user involvement and system success. *Information and Software Technology*, pages 148–169, 2015.

[2] G. Beenen, K. Ling, X. Wang, K. Chang, D. Frankowski, P. Resnick, and R. E. Kraut. Using social psychology to motivate contributions to online communities. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pages 212–221, 2004.

[3] J. Behrens, A. Kistner, A. Nitsche, and B. Swierczek. *The Principles of LiquidFeedback*. Interaktive Demokratie e.V., 2014.

[4] B. Boehm and A. Egyed. Software requirements negotiation: some lessons learned. In *Proceedings of the 20th International Conference on Software Engineering*, pages 503–506, 1998.

[5] A. Chadwick. Web 2.0: New challenges for the study of e-democracy in an era of informational exuberance. *I/S: A Journal of Law and Policy for the Information Society*, 2008.

[6] E. G. Clary and M. Snyder. The motivations to volunteer theoretical and practical considerations. *Current directions in psychological science*, pages 156–159, 1999.

[7] L. Galvis-Carreño and K. Winbladh. Analysis of user comments: an approach for software requirements evolution. In *Proceedings of the 35th International Conference on Software Engineering*, 2013.

[8] E. Guzman and W. Maalej. How do users like this feature? a fine grained sentiment analysis of app reviews. In *Proceedings of the 22nd International Requirements Engineering Conference*, 2014.

[9] B. N. Hague and B. Loader. *Digital Democracy: Discourse and Decision Making in the Information Age*. Routledge, 1999.

[10] J. Hamari, J. Koivisto, and H. Sarsa. Does gamification work? – a literature review of empirical studies on gamification. In *Proceedings of the 47th Hawaii International Conference on System Sciences*, pages 3025–3034, 2014.

[11] S. Jabbusch. Liquid democracy in der piratenpartei. Master thesis, University of Greifswald, 2011.

[12] J. Kruger and D. Dunning. Unskilled and unaware of it: how difficulties in recognizing one's own incompetence lead to inflated self-assessments. *Journal of personality and social psychology*, page 1121, 1999.

[13] J. Lanier. Digital maoism: The hazards of the new online collectivism. *Edge: The Third Culture*, 2006.

[14] K.-Y. Lin and H.-P. Lu. Why people use social networking sites: An empirical study integrating network externalities and motivation theory. *Computers in Human Behavior*, pages 1152 – 1161, 2011.

[15] W. Maalej and D. Pagano. On the socialness of software. In *9th International Conference on Dependable, Autonomic and Secure Computing*, pages 864–871, 2011.

[16] B. Mako Hill. The institute for cultural diplomacy and wikipedia. 2013.

[17] D. Pagano and B. Brügge. User involvement in software evolution practice: A case study. In *Proceedings of the 35th International Conference on Software Engineering*, pages 953–962, 2013.

[18] D. Pagano and W. Maalej. User feedback in the appstore : an empirical study. In *Proceedings of the International 21st Conference on Requirements Engineering*, pages 125–134, 2013.

[19] A. Paulin. Through liquid democracy to sustainable non-bureaucratic government. In *Proceedings of the Conference for e-Democracy and Open Government*, pages 205–217, 2014.

[20] D. Renzel, M. Behrendt, R. Klamma, and M. Jarke. Requirements bazaar: Social requirements engineering for community-driven innovation. In *21st International Requirements Engineering Conference*, 2013.

[21] R. Ryan and E. Deci. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary educational psychology*, pages 54–67, 2000.

[22] P. Seebach. Baby duck syndrome: Imprinting on your first system makes change a very hard thing. *IBM DeveloperWorks*, 2005.

[23] N. Seyff, G. Ollmann, and M. Bortenschlager. irequire: Gathering end-user requirements for new apps. In *22nd International Requirements Engineering Conference*, pages 347–348, 2011.

[24] A. Terceiro, L. Rios, and C. Chavez. An empirical study on the structural complexity introduced by core and peripheral developers in free software projects. In *24th Brazilian Symposium on Software Engineering*, pages 21–29, 2010.

Authorized licensed use limited to: National University Fast. Downloaded on September 24,2020 at 06:44:03 UTC from IEEE Xplore. Restrictions apply.