Towards Systematic Analysis of Continuous User Input

Dennis Pagano
Technische Universität München
Munich, Germany
pagano@cs.tum.edu

ABSTRACT

Novel requirements elicitation approaches suggest to continuously gather and communicate user input to engineering teams. The resulting data usually consists of a large amount of unstructured information in the form of natural language and may include conflicting user needs that have to be detected and resolved to obtain consistent requirements. This position paper provides a first step towards a systematic analysis of continuous user input by identifying its main challenges and aligning helpful techniques from requirements engineering research to address the challenges in a common framework.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specifications

General Terms

Human Factors, Documentation, Measurement

Keywords

Social software, continuous user input, requirements elicitation, systematic analysis, user community

1. INTRODUCTION

The success of a software product depends to a substantial amount on its users [3]. Engineers gather input and feedback from users throughout the software lifecycle, since adapting to changing user demands is necessary to continuously meet users' expectations [10]. To elicit requirements for a system, analysts typically mediate users while they provide information defining their needs. Further, when actually using a system, users may give feedback in the form of bug reports and feature requests.

Novel approaches suggest to continuously and remotely gather input from the users of a system to be able to adjust to changing requirements and user needs [13, 17–19]. However, eliciting requirements from continuous user input poses several technical challenges, in particular because a large amount of data has to be analyzed. We argue that these challenges have to be addressed in a systematic way to facilitate the analysis of continuous user input.

The contribution of this paper is threefold. First, we identify the main technical challenges for the analysis of continuous user input. Second, we provide a summary of techniques and approaches from requirements engineering research that help to solve the occurring problems. Third, we align these techniques and propose a common framework for the systematic analysis of continuous user input.

The paper is structured as follows. We start by describing which factors complicate eliciting requirements from continuous user input (Section 2). We then summarize techniques from requirements engineering research that address the identified challenges on similar data (Section 3). Next, we illustrate how these methods can be combined in a common framework to systematically analyze continuous user input (Section 4). Finally, we describe our plans for future work (Section 5) and conclude the paper (Section 6).

2. CHALLENGES

Engineers have to interpret, analyze, model, and validate user input to elicit formal requirements of a system [14]. Traditional techniques for requirements elicitation rely on input gathered from a small number of selected users and during a pre-defined time period. Requirements elicitation becomes more complicated if users continuously, and while using the system, provide information without being guided by analysts. In the remainder of this section we describe the main technical reasons for this.

C1. Quantity of Data.

Continuously gathering user input results in large amounts of data, complicating manual analysis techniques. For instance, Maalej et al. [13] suggest to instrument software to continuously gather usage and feedback data from all users. The *quantity* of collected user input data, however, imposes limits on how this information can be processed by requirements engineers. With increasing number of users and frequent collection of their input, manual analysis techniques become infeasible.

C2. Missing Structure.

Gathering user input without formal limits results in unstructured data, complicating automated analysis techniques. Engineers have to interpret information provided by users to

SSE'11, September 5, 2011, Szeged, Hungary.

understand their needs. The *structure* of this information determines which analysis techniques can be applied and how difficult an automated interpretation is. For instance, Seyff et al. [19] introduce a framework that allows users to capture their individual needs using natural language text. However, informal data such as natural language is generally hard to analyze using automated techniques because of its high degree of freedom.

C3. Content and Quality.

Gathering user input without mediation results in unpredictable content and quality, complicating automated analysis techniques. If users provide input without professional support, the resulting information *content* and *quality* is unpredictable, what can lead to misunderstandings [13, 20]. For example, users might express system properties using their own, possibly inhomogeneous terminology. In addition, users are often not motivated to invest the necessary time to provide useful information. Although studies showed that engineers are able to *manually* transcribe remotely collected user needs into requirements [19], their content and quality pose challenges for automated approaches.

C4. Conflicting Preferences.

Continuously gathered user input can lead to frequent conflicts, complicating manual identification and resolution techniques. A major aim of continuous feedback approaches is to collect user input individually [17], whereas traditional requirements elicitation approaches typically generalize the input of single users into categories to satisfy the majority of all users. However, different users will have different, possibly *conflicting* preferences, needs, and expectations regarding the system. To construct requirements and features from this data, conflicts have to be identified and resolved. Manual techniques such as requirements negotiation attempt to resolve such conflicts between stakeholders [14]. However, research has shown that users' preferences regarding system features do not remain stable [9].

3. ENABLING TECHNIQUES

In this section we discuss techniques from requirements engineering research that address the challenges we identified above. We focus on automated or semi-automated methods, due to the large amount of data generated when continuously gathering user input data. Most of the techniques we found originate from research for large-scale and distributed projects.

We first study techniques that can help requirements engineers to deal with a large amount of user input. Second, we investigate methods to analyze requirements specifications written in natural language. Third, we discuss how user input can be enriched to gain more expressive information. Last, we survey techniques that facilitate the identification and resolution of preference conflicts regarding requirements.

3.1 Addressing Quantity of Data (C1)

Requirements engineering research for large-scale projects suggests *filtering and prioritizing of information* to deal with the large number of emerging requirements. Recent publications investigate the exploitation of social network analysis techniques to select relevant stakeholders for specific requirements engineering tasks [9, 11]. The results show that social network measures provide means to estimate the relevance of specific requirements to individual stakeholders and enable the prioritization of requirements based on stakeholders' ratings. Consequently, requirements can be filtered, prioritized and proposed to stakeholders according to their preferences and competencies. Moreover, several authors [6, 9, 11] suggest recommendation techniques such as collaborative filtering to proactively recommend additional relevant requirements to stakeholders.

3.2 Addressing Missing Structure (C2)

A growing body of research investigates how data mining and information retrieval methods can support the construction and refinement of requirements and feature models from natural language specifications. Their goal is to support requirements engineers in large-scale projects by automatically *consolidating user input*. Castro-Herrera et al. [6] analyze tf-idf[1] values to identify key topics in a large number of feature requests written in natural language. Using unsupervised clustering methods, they further group the feature requests according to these topics. Similarly, Alves et al. [1] derive similarities between requirements from different requirements specifications using the Latent Semantic Analysis (LSA) and Vector Space Model (VSM) techniques, and identify requirements clusters based on these results. Dumitru et al. [7] utilize text mining techniques and propose incremental diffusive clustering (IDC) to discover domain-specific features from product descriptions.

Although they are set up in different contexts, all approaches consolidate a large number of natural language requirements documents in two steps. First, they apply text mining methods to identify important concepts across the documents. Second, they use clustering techniques to identify a structure according to these concepts.

3.3 Addressing Content and Quality (C3)

Information content and quality of user input directly impact how precisely requirements engineers can understand users' needs. User input with low quality and in particular inhomogeneous language is hard to analyze. Several researchers suggest to use semantically rich context information to *augment user input* and thus reduce communication gaps. Schneider et al. [18] let users themselves annotate their feedback with a selection regarding the type of feedback (e.g. complaint) and the context where the feedback applies (e.g. subsystem). Though this additional input facilitates automatic data pre-processing, not all users might be able to provide such information. Maalej et al. [13] propose a framework to automatically collect relevant context information by instrumenting the software and the users' working environment.

3.4 Addressing Conflicting Preferences (C4)

Research has shown that there are ways to *identify and resolve inconsistencies* in stakeholder preferences regarding requirements. Lim et al. [11] present a tool that identifies stakeholders with conflicting preferences for requirements based on a social network of these stakeholders. They further suggest that social network measures can be utilized to guide the conflict resolution process. Felfernig et al. [8] introduce a method to automatically detect minimal sets of

[1] term frequency, inverse document frequency

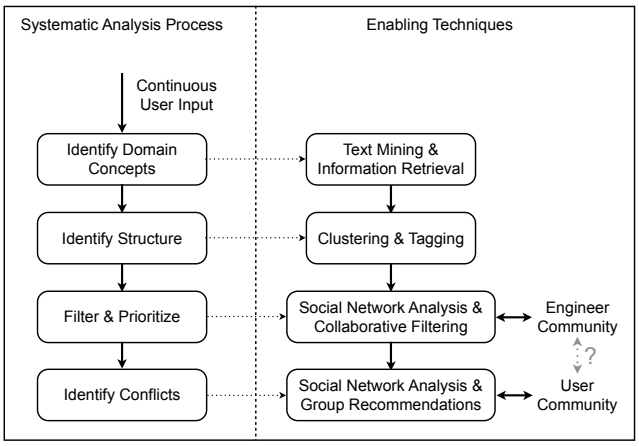

Figure 1: Proposed Framework for Systematic Analysis of Continuous User Input.

inconsistent stakeholder preferences. Moreover, they propose to exploit group recommendations to proactively support stakeholders in their decision making [9].

Though we found techniques that address conflicting preferences, they only work with clearly specified preferences over defined requirements. Consequently, user input first needs to be transformed into more formal requirements and preferences, before these methods can be applied. We could not find studies showing if and how it is possible to transform user specified information into preferences regarding requirements. However, recent research from social network analysis and data mining investigates the identification of themes and sentiments in social media [16]. Though the results show that it is possible to extract users' opinions regarding specific products or features, more research has to be done to evaluate the applicability of such approaches in the context of continuous user input.

4. PROPOSED SOLUTION

Researchers spend considerable effort to improve requirements elicitation by providing tool support for complex and time consuming tasks. Consequently we could find existing work that addresses one or even more of the identified challenges that continuous user input presents. However, there is little work on holistic approaches or frameworks that describe how to tailor requirements elicitation to continuous user input. Castro-Herrera et al. [6] describe a framework that identifies and groups similar feature requests and proactively recommends relevant stakeholders to collaboratively transcribe these requests into more formal requirements. Although this approach is capable of supporting the analysis of user input in projects with many stakeholders, it does not include activities to identify conflicting requirements preferences.

We claim that approaches to continuously gather user input call for a common framework to *systematically analyze* the collected information during requirements elicitation. Figure 1 shows our proposal towards such a framework. Our approach is based on the suggestion of several researchers [13, 17–19] to continuously and remotely gather user input in the form of user feedback (e.g. modification, enhancement, or feature requests), specified in natural language artifacts. The goal of the proposed framework is to *support* requirements engineers during the elicitation of formal requirements from continuous user input by *reducing* the amount of information that engineers have to analyze manually. We identified four main steps that constitute a systematic analysis process. Moreover, we claim that these steps can be realized by a hybrid approach that uses techniques we surveyed in Section 3.

In the first step, gathered user input has to be analyzed to *identify included domain concepts* that represent common semantic entities. To this end, a document corpus is generated from user input data, using a word stemmer and removing stop words. Text mining and information retrieval techniques such as LSA can then identify important terms in the raw data and measure their overall influence (e.g. using tf-idf). As a result, each user input artifact is finally annotated with a weighted vector of terms representing the contained domain concepts.

In the second step, these domain concepts are used to *identify underlying structure* in the user input data. To this end, unsupervised clustering techniques such as bisect k-means clustering are applied on the user input data. The optimal number of clusters k can be determined using heuristics based on the identified domain concepts [5]. User input artifacts are then grouped into the emerging clusters, leading to a structure based on the identified domain concepts. The identified structure can finally be represented using tags. Novel unsupervised clustering methods like IDC [7] already include a pre-processing routine to identify semantic entities, and thus combine the first two steps we propose.

The third step is in charge of *filtering and prioritizing* data to reduce the amount of information for engineers. In our approach, user input can be prioritized according to its relevance for both users and engineers. First, the relevance of a certain user input to the user community can be assessed by measuring the amount of similar information gathered from other users (expressed for instance by the cluster size). On the other hand, we propose to predict the relevance of user input clusters to specific engineers using social network analysis methods. This information can aid to determine which engineer should be responsible for a given set of user input data [9]. As a prerequisite for this step, there has to be a social network of engineers that can be analyzed. Research has shown that such networks can be constructed [11]. Additionally, and more proactively, information can be suggested to engineers using techniques like collaborative filtering.

In a fourth step, *conflicting user preferences* have to be *identified*. We claim that social network analysis techniques should be utilized to identify possible users whose preferences are conflicting. However, as a prerequisite for conflict identification gathered user input data first has to be interpreted to determine user preferences. We are not aware of an automated way of performing such an interpretation in the current state of research. However, recently described techniques to automatically identify sentiments from user generated content [16] could help to derive user preferences from user input data by selecting according candidates that have to be reviewed by engineers. Moreover, group recommendation techniques can support users proactively while creating their preferences to reduce or even avoid conflicts [9].

The degree of automation that can be achieved in the described steps is limited. Consequently, the proposed frame-

work represents a set of tools that help to analyze large amounts of user input data in a systematic way rather than an autonomous solution. In particular conflict identification techniques can only generate candidates for conflicting user preferences which have to be reviewed, discussed, and acknowledged by engineers.

The goals of both last steps can be reached performing social network analyses on engineer and user communities. However, social networks of engineers and users allow for a bidirectional communication. We currently see two benefits, which have to be elaborated in future work. First, analysis results can be integrated in social networks of engineers. Second, social networks of users can be utilized to make clarification requests and to report about current progress regarding a specific issue.

While not explicitly mentioned in the proposed framework, we expect additional context information to facilitate the analysis process. The complementary nature of this information could in particular support the identification of domain concepts and conflicting preferences.

5. FUTURE WORK

This position paper represents a starting point. Further effort needs to be put into the development and evaluation of the proposed framework, as well as into the investigation of the degree of automation that can be reached. Moreover, systematic analysis of continuous user input is no end in itself. Further research has to investigate how the proposed process fits into existing requirements elicitation approaches.

Likewise, further research is necessary to investigate how the different techniques we proposed in our approach can be combined best, and what are necessary preconditions. Moreover, existing work does not analyze data that was gathered directly from users, but rather requirements specifications or other data generated by general project stakeholders. Consequently, future studies should evaluate how these techniques perform on real user input data.

Our proposed framework suggests to utilize both engineer and user communities in a systematic analysis process. We think that communication channels between engineers and users are essential, for instance to request clarifications when the gathered information is not clear enough. Begel et al. [4] claim that social media supports two-way communication between users and companies in a scalable way. Moreover, Pagano and Maalej [15] found that engineers in open source communities already use social media to discuss requirements. We plan to further study communication and influences between both communities, what is highlighted by the question mark in Figure 1.

Social media enabled platforms like UserVoice[2] and GetSatisfaction[3] allow users to collaboratively share new ideas and vote on existing suggestions for new features. Bajic and Lyons [2] found that this collaboration focuses users' efforts, which leads to more homogeneous feature requests. We plan to investigate if and how such platforms can be beneficial for gathering and analyzing continuous user input.

During our research, we found relations to requirements engineering research for large-scale and distributed projects. Both have to deal with similar challenges, such as information overload, inadequate stakeholder input, and the need for requirements prioritization [12]. We plan to further investigate possible alignments of both research lines.

[2]http://uservoice.com

[3]http://getsatisfaction.com

6. CONCLUSION

Novel requirements elicitation approaches propose to continuously collect and communicate user input to engineering teams. We identified several challenges for requirements elicitation activities that arise from continuous user input. Resulting data usually consists of a large number of natural language information with possibly inhomogeneous terminology. It may further include conflicting user needs that have to be detected and resolved to obtain consistent requirements. We discussed several techniques from requirements engineering research that address the identified challenges on similar data. Moreover, we aligned these techniques in a common framework to systematically analyze continuous user input during requirements elicitation. Our results call for further research that aims at developing and evaluating the proposed framework in a real-world setting.

7. ACKNOWLEDGEMENT

This work has been supported by the FastFix project, which is funded by the 7th Framework Programme of the European Commission, grant agreement no. FP7-258109.

8. REFERENCES

[1] V. Alves, C. Schwanninger, L. Barbosa, A. Rashid, P. Sawyer, P. Rayson, C. Pohl, and A. Rummler. An Exploratory Study of Information Retrieval Techniques in Domain Analysis. *2008 12th International Software Product Line Conference*, pages 67–76, Sept. 2008.

[2] D. Bajic and K. Lyons. Leveraging social media to gather user feedback for software development. In *Proceeding of the 2nd international workshop on Web 2.0 for software engineering*, pages 1–6. ACM, 2011.

[3] H. Barki and J. Hartwick. User participation and user involvement in information system development. In *System Sciences, 1991. Proceedings of the Twenty-Fourth Annual Hawaii International Conference on*, pages 487–492, Hawaii, USA, 1991.

[4] A. Begel, R. DeLine, and T. Zimmermann. Social media for software engineering. In *Proceedings of the FSE/SDP workshop on Future of software engineering research*, pages 33–38. ACM, 2010.

[5] F. Can and E. A. Ozkarahan. Concepts and Effectiveness of the Clustering Methodology for Text Databases. *ACM Transactions on Database Systems*, 15(4):483–517, 1990.

[6] 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. *Proceedings of the 16th IEEE International Requirements Engineering Conference*, pages 165–168, Sept. 2008.

[7] H. Dumitru, M. Gibiec, N. Hariri, J. Cleland-Huang, B. Mobasher, C. Castro-Herrera, and M. Mirakhorli. On-demand feature recommendations derived from mining public product descriptions. In *Proceedings of the 33rd International Conference on Software Engineering*, pages 181–190. ACM, 2011.

[8] A. Felfernig, M. Schubert, M. Mandl, and P. Ghirardini. Diagnosing Inconsistent Requirements Preferences in Distributed Software Projects. In *Proceedings of the 3rd International Workshop on Social Software Engineering*, pages 1–8, Paderborn, Germany, 2010.

[9] A. Felfernig, M. Schubert, M. Mandl, F. Ricci, and W. Maalej. Recommendation and decision technologies for

requirements engineering. *Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering - RSSE '10*, pages 11–15, 2010.

[10] M. Lehman. Programs, life cycles, and laws of software evolution. *Proceedings of the IEEE*, 68(9):1060–1076, 1980.

[11] S. L. Lim, D. Damian, and A. Finkelstein. StakeSource2. 0: Using Social Networks of Stakeholders to Identify and Prioritise Requirements. In *Proceedings of the 33rd ACM/IEEE International Conference on Software Engineering, in press*, 2011.

[12] S. L. Lim and A. Finkelstein. StakeRare: Using Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation. *IEEE Transactions on Software Engineering*, pages 1–32, 2011.

[13] W. Maalej, H. Happel, and A. Rashid. When users become collaborators: towards continuous and context-aware user input. *Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*, pages 981–990, 2009.

[14] B. Nuseibeh and S. Easterbrook. Requirements engineering: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 35–46. ACM, 2000.

[15] D. Pagano and W. Maalej. How Do Developers Blog? An Exploratory Study. In *Proceedings of the 8th Conference on Mining Software Repositories*. ACM, 2011.

[16] J. Pal and A. Saha. Identifying Themes in Social Media and Detecting Sentiments. In *2010 International Conference on Advances in Social Networks Analysis and Mining*, pages 452–457. IEEE, Aug. 2010.

[17] N. Qureshi, N. Seyff, and A. Perini. Satisfying User Needs at the Right Time and in the Right Place: A Research Preview. *Proceedings of the 17th International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 94–99, 2011.

[18] K. Schneider, S. Meyer, M. Peters, F. Schliephacke, J. Mörschbach, and L. Aguirre. *Feedback in Context : Supporting the Evolution of IT-Ecosystems*. Springer Berlin / Heidelberg, 2010.

[19] N. Seyff, F. Graf, and N. Maiden. Using Mobile RE Tools to Give End-Users Their Own Voice. In *Requirements Engineering, IEEE International Conference on*, pages 37–46, 2010.

[20] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schroter, and C. Weiss. What Makes a Good Bug Report? *IEEE Transactions on Software Engineering*, 36(5):618–643, Sept. 2010.