V:issue:lizer
Exploring Requirements Clarification in Online Communication over Time

Eric Knauss, Daniela Damian
SEGAL, University of Victoria, Victoria B.C., Canada
knauss@computer.org, danielad@cs.uvic.ca

***Abstract*—This demo introduces V:ISSUE:LIZER, a tool for exploring online communication and analysing clarification of requirements over time. V:ISSUE:LIZER enables managers to identify hotspots in current development activities, to analyze communication problems, and to identify developers that are knowledgeable about domain or project related issues through visualizations. Our preliminary evaluation shows that V:ISSUE:LIZER offers managers valuable information for their decision making.**

***Keywords*-requirements clarification patterns; distributed requirements engineering; communication of requirements**

TODO LIST

Remove all todos

I. INTRODUCTION

Large software projects often need to collaborate across geographically distributed sites and depend on online communication to perform requirements related activities. More and more teams employ agile approaches that aim at discovering requirements iteratively and rely on frequent communication instead of requirements documentation. In such approaches, requirements are defined in the form of user stories, and ongoing discussions around these user stories serve as the main mechanism to clarify the meaning of requirements and to coordinate their implementation [1]. Recording such discussions and decisions in online project repositories is an emerging best practice, not only in large and distributed projects [2]. IBM®'s Rational Team Concert® project, with a large distributed team, is an example in which management mandates the recording of all critical communication in the project repository for future use [3]. Consequently, online project repositories contain a valuable amount of requirements-related communication.

don't understand: other projects use wi or tickets or tasks - Jira

Communication to *clarify* requirements is an important aspect of software projects, but a predominance of clarification communication late during the implementation of a requirement can indicate that the team does not have a sufficient understanding of the underlying requirement. When stakeholders continue to clarify the requirement because it is ambiguous, incomplete, or has frequent changes, the expected evolution of a requirement from an initial idea, through clarification, to design and full implementation, often stagnates. As a result, its implementation can be delayed, never completed or sometimes never get started. Current requirements management tools offer little support for identifying requirements with progression problems, thereby lowering the project manager's ability to intervene in a timely manner.

In this demo we present a novel tool for analyzing online communication that helps managers to visualize the progression of clarification communication relative to other communication related to a particular requirement, thus providing support to differentiating between healthy and problematic requirements in the project. V:ISSUE:LIZER also visualizes stakeholder social networks related to a particular requirement to support the identification of communication breakdowns and experts for given topics.

We conducted a preliminary evaluation in two ways. Firstly, we evaluated V:ISSUE:LIZER's ability to correctly identify communication instances concerned with clarification and it's ability to derive meaningful visualizations of the clarification trajectory [4]. Secondly, we showed V:ISSUE:LIZER's visualizations to software managers and asked them, whether the visualization was useful, if it did offer information they would have missed otherwise, and what actions they would perform based on the feedback, if any. This feedback lead to the development of additional features, e.g. social network analysis.

II. BACKGROUND AND RELATED WORK

In this paper, we use the term *requirements discussion* to refer to a thread of online communication that is related to a given requirement. A requirements discussion consists of *discussion events*, i.e. contributions to the discussion. We are particularly interested in *clarification events*, which is a discussion event in which the discussant seeks to improve the understanding of the requirement by either asking for clarification or by offering additional information that make the requirement clearer. Many software projects use online

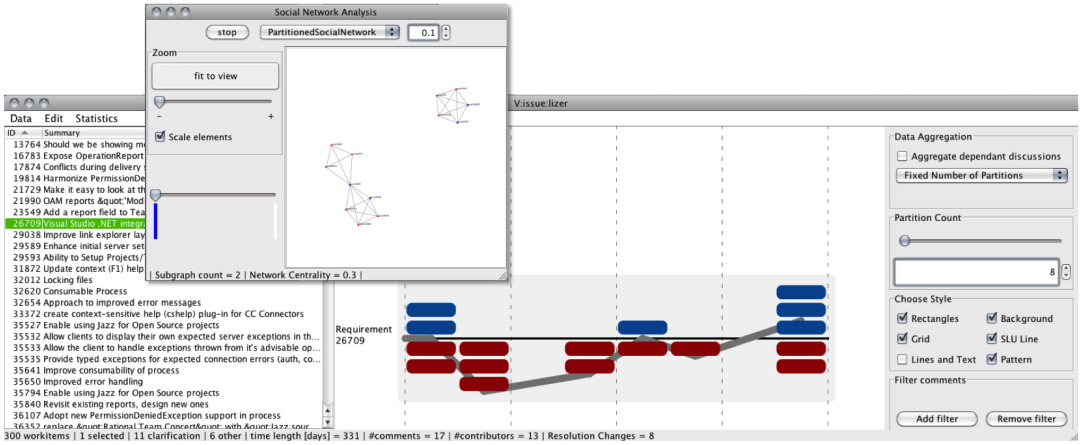

Figure 1: The main window of the V:ISSUE:LIZER shows a list of requirements (e.g. user stories in an issue tracker) and their clarification trajectories.

tools to store requirements discussions, e.g. issue trackers or task management systems like bugzilla, or jira. In such projects, requirements are distinguished by a certain type of issue (e.g. *user story, enhancement*) and the requirements discussion is stored as a series of comments to this issue.

Issues in such systems crosscut both the technical aspects of a software project and social aspects of collaboration and communication [5]. Thus, giving managers the ability of finding the right tasks at the right time can be crucial to the success of a project. Treude and Storey identified the lack of visualizations as one of the most important short comings of today's task management systems [6]. They also found that dashboards that report on the state of a task management system can become pivotal to task prioritization in critical project phases.

Systems like Bugzilla can play a key role in managing software projects, as Ellis et al. [7] report based on results from interviews of how developers use Bugzilla. The motivation for their study was the design of a visualization tool for tasks. Their visualization reveals social and historical patterns in tasks, but it does not focus on exploration clarification activities.

Many related studies focus on mining and analyzing quantitative data to reveal information about the evolution of the system or to predict future behaviours but only few works are concerned with visualizing and exploring this information space. Treude et al. [8] present the workitemexplorer, a related tool that allows the exploration of information stored in a task management system. Compared to this work, V:ISSUE:LIZER allows the analysis of a specialized aspect in this information space, i.e. the analysis of online communication related to clarification of requirements.

III. V:ISSUE:LIZER

V:ISSUE:LIZER is an interactive tool that allows software developers and managers to dynamically explore the discussion of requirements in online repositories with a focus on highlighting the difference between clarification and implementation related communication.

The main window shows a list of requirements on the left (e.g. workitems in jazz, items in jira, issues in other systems) (see Figure 1). V:ISSUE:LIZER adds visualizations to the selected discussions in the centre or in an extra window. These visualizations help to assess the communication through discussion events (e.g. comments) related to these requirements. The panel on the right allows users to adjust parameters of the visualization. Most importantly, the resolution of time intervals for the visualization can be adjusted, either as a fixed number of intervals (here: three and eight intervals), or as a fixed time interval (e.g. days, weeks, month). V:ISSUE:LIZER currently supports two different visualizations:

1) Clarification Trajectories: This visualization shows how the percentage of clarification events to other discussion events related to a requirement changes over its lifetime. As the visualization of the clarification trajectory (see Figure 2) is a new concept, it needs some explanation. The black line represents the lifetime of the requirement discussion from the creation of the requirement in the system to the last recorded discussion event. Dashed lines divide the lifeline into quarters and help to see in which part of the lifetime discussion occurs. Discussion events are depicted by rectangles. They are shown below the lifeline, if they are clarification events and above the lifeline if not. A grey line shows the sum of clarification. In a classic trajectory with clarification

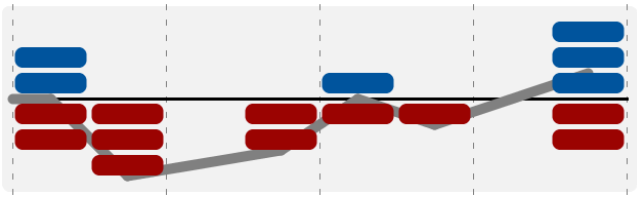

Figure 2: Example of a requirements discussion's clarification trajectory

up-front and only implementation related communication in the end, this grey line will start in the bottom left and raise to the upper right corner. V:ISSUE:LIZER generates these trajectory based on automatic classification of discussion events and identifies reoccurring patterns (*textbook-example, indifferent, discordant, procrastination, back-to-draft, and happy-ending*) [4]. Typically, there is a number of requirements without pathological findings, e.g. user stories with some clarification in the beginning and other communication events later on that show progress. More suspicious trajectories show large amounts of clarification late in the iteration, perhaps even after the issue seemed to be solved, or no clarification at all, even though the requirement seems to be complex.

remove last two sentences?

2) Social Networks: This visualization shows the communication pattern of those participating in a requirement's discussion. The developers are presented as nodes; connections between nodes are weighted by the amount of communication both developers share about the requirement in a specific time interval (as defined above) in a requirement's lifetime. Thus, subgraphs of the contributors to the discussion in each time interval are created. Two subgraphs are connected, if stakeholders appear in several time intervals. V:ISSUE:LIZER also displays the developer node as a pie chart to show the percentage of clarification vs. implementation-related communication in which the developer is involved (as highlighted in Figure 4).

IV. EXAMPLE SCENARIOS

In this section we describe examples that highlight the functionality that V:ISSUE:LIZER tool offers to developers and managers (referred to as users henceforth).

A. Are there problematic requirements?

To identify requirements for which the clarification trajectory indicates potential problems in their development, the user performs the following steps in V:ISSUE:LIZER :

1) The user loads a set of requirements (shown as the requirements list on the left panel in Figure 1).
2) V:ISSUE:LIZER automatically analyzes the online communication repository and the discussion related to each of these requirements.
3) When a requirement is selected from the requirements list its *clarification trajectory* is displayed in the middle panel (see Figure 2)
4) V:ISSUE:LIZER displays suggestive pattern names for the trajectory, e.g. *happy-ending* in Figure 2, because the trajectory shows a dominance of clarification even in the second half of the timeline and the grey line only raises above the lifeline in the very end.
5) By scrolling through the list of requirements and associated trajectories the user can decide based on this rich information where to invest more resources to tackle requirements that appear to have problematic development.

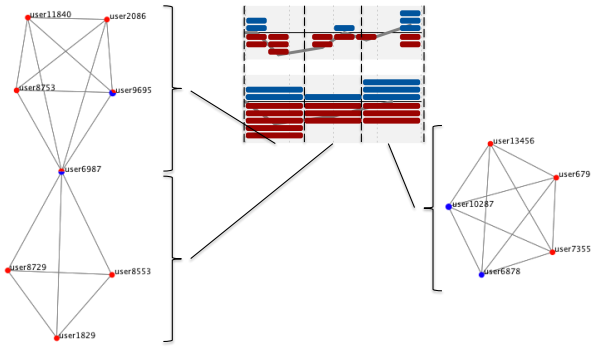

Figure 3: Example of a requirements discussion's social network

B. Are there communication breakdowns?

Having identified potentially problematic requirements, V:ISSUE:LIZER allows the user to more closely investigate who participates in the communication around a requirement (Figure 3 shows the social network for the requirement presented in Figure 2) as follows:

1) The user opens the social network analysis view.
2) V:ISSUE:LIZER displays the social network for the selected requirement.
3) The user analyzes the network and investigates if structural communication problems exist.

In this example, the user might conclude that there is no single person who is coordinating the analysis and work related to this requirement, because there is no actor who participates in all relevant time intervals. Thus, the user decides to assign this responsibility to a more experienced developer.

C. Who is knowledgeable about a given topic?

Integrating the right persons in the loop for an important feature is a crucial ability for managers. To support managers in this task, V:ISSUE:LIZER distinguishes between two types of knowledge: domain knowledge that shows in discussion events related to clarification and implementation specific knowledge that shows in other discussion events.

1) The user selects a number of requirements that relate to a feature or higher-level topic.
2) V:ISSUE:LIZER creates the social networks generated from requirements' discussions and displays them in a single large network (see Figure 4).
3) Based on the pie charts in the social network, the manager identifies candidates with a balanced percentage of clarification and implementation communication.
4) The manager looks for central developers.

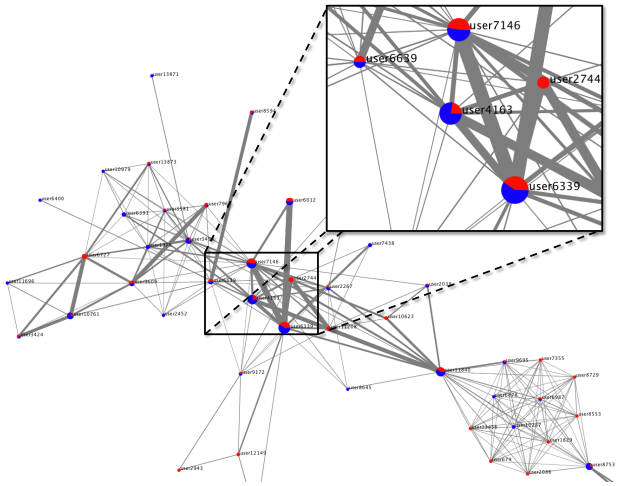

Figure 4: Example of a social network for a set of requirements discussions

Central actors with many connections might already have a very high workload, but there exist good candidates that are less central and connect two subnets.

V. Preliminary Evaluation

Dana, feel free to improve conclusion

In order to evaluate the V:ISSUE:LIZER and its underlying concepts, we need to investigate several things. First we need to show that V:ISSUE:LIZER is able to distinguish between communication events that deal with clarification and other events. Secondly, we need to determine if and when the feedback from our V:ISSUE:LIZER tool is beneficial for practitioners.

A. Ability to identify clarification events

V:ISSUE:LIZER currently uses a Bayesian classifier to identify clarification events, i.e. a supervised machine learning algorithm. In order to evaluate the ability to identify clarification events, we need to show(i) that the classifier reaches an acceptable performance with realistic amount of training and (ii) that this performance is sufficient to generate meaningful trajectories on the fly. We investigated both aspects based on a case study in the IBM Rational Team Concert (RTC) project [4]. RTC is a globally distributed software project and as such employs online communication to an extend that guarantees a sufficient amount of the overall communication to be available.

In order to provide training data, two raters manually classified ca. 1200 communication events with an acceptable inter-rater agreement. Based on this training data, we applied 10-fold cross evaluation and measured a recall of 0.943 and a precision of 0.678, resulting in an acceptable f-measure of 0.789. Especially the high recall leads to acceptable trajectories that are comparable with those constructed based on manual classification (c.f. discussion in [4]).

B. Ability to support decisions of managers

We started to evaluate the ability to support software managers in decision making, using the feedback for continues improvement. For the first round of evaluation, we presented V:ISSUE:LIZER to four software managers, followed by a semi-structured interview. We were able to interview participants of the VIATEC Software Management Round Table, a local group of software managers that regularly meet in Victoria to exchange experiences. Our interviewees agreed that the clarification trajectories provide significant information to support decisions on resource allocation and risk management. In addition, they suggested that the V:ISSUE:LIZER could also support project retrospectives and process improvement efforts. The main issue raised by Victoria's software managers was the need of seeing who was participating in a given requirements related discussion. Accordingly, a trajectory without clarification would be suspicious if no experienced developer participated in its underlying discussion.

As an reaction to this feedback, we integrated the ability to generate social networks into V:ISSUE:LIZER and then talked to five managers of the IBM RTC project and related projects for a second round of evaluation. At IBM, our interviewees agreed that the clarification trajectories together with the associated social network graphs are helpful. Especially, when the software development reaches the *end game*, i.e. a time close to the release where mostly testing and polishing takes place, clarification events would be very suspicious and would indicate a high risk that needs management attention.

VI. Conclusion and Future Work

Dana, please review conclusion

With V:ISSUE:LIZER we introduced a visualization tool to analyze requirements clarification in online communication over time. Especially agile and distributed projects demand such analysis: agile projects often only sketch requirements in sufficient detail to plan the next iteration and leave the details to be clarified during the development; distributed projects often depend on online communication and challenge their project managers' ability to assess the shared understanding in the team. Our preliminary evaluation has shown that our visualizations allow managers to identify hotspots, e.g. user stories that are not clear to the team. Furthermore, V:ISSUE:LIZER supports managers in investigating the cause of those hotspots and in identifying suitable actions to disarm problematic or risky situations where the team has insufficient understanding of requirements.

In future work, we will evaluate how practitioners use our tool in their daily work. This will help us to gain further insight on how managers can use information about requirements clarification over time and to quantify the benefits that tools like the V:ISSUE:LIZER can offer.

Such further evaluation should also relate features of online communication (i.e. network centrality, late clarification, no clarification) with typical problems of requirements related discussions, such as feature creep [9] or symmetry of ignorance [10].

ACKNOWLEDGEMENTS

There is a number of people we should acknowledge.

REFERENCES

[1] L. Cao and B. Ramesh, "Agile requirements engineering practices: An empirical study," *Software, IEEE*, vol. 25, no. 1, pp. 60 –67, jan.-feb. 2008.

[2] J. Aranda, R. Khuwaja, and S. M. Easterbrook, *Discovering the shared understanding dynamics of large software teams*. ACM, 2007, pp. 1–4.

[3] R. Frost, "Jazz and the eclipse way of collaboration," *IEEE Software*, vol. 24, no. 06, pp. 114–117, 2007.

[4] E. Knauss, D. Damian, G. Poo-Caamaño, and J. Cleland-Huang, "Detecting and classifying patterns of requirements clarifications," in *Proc. of the 20th Intl. Requirements Engineering Conf. (RE '12)*, Chicago, USA, 2012, pp. 251–260.

[5] R. E. Kraut and L. A. Streeter, "Coordination in software development," *Commun. ACM*, vol. 38, no. 3, pp. 69–81, 1995.

[6] C. Treude and M.-A. Storey, "Awareness 2.0: Staying aware of projects, developers and tasks using dashboards and feeds," in *Proc. of the 32th Intl. Conf. on Software Engineering (ICSE'10)*, vol. 1, Zurich, Switzerland, 2010, pp. 365–374.

[7] J. B. Ellis, S. Wahid, and W. A. Kellogg, "Task and social visualization in software development: Evaluation of a prototype," in *Proc. of the Conf. on Human Factors in Computing Systems (CHI'07)*, San Jose, USA, 2007, pp. 577–586.

[8] C. Treude, P. Gorman, L. Grammel, and M.-A. Storey, "Workitemexplorer: Visualizing software development tasks using an interactive exploration environment," in *Proc. of the 34th Intl. Conf. on Software Engineering (ICSE'12)*, Zurich, Switzerland, 2012, pp. 1399–1402, formal research dem.

[9] C. Jones, "Strategies for managing requirements creep," *IEEE Computer*, vol. 29, no. 6, pp. 92–94, 1996.

[10] G. Fischer, "Social Creativity, Symmetry of Ignorance and Meta-Design," *Knowledge-Based Systems Journal*, 2000.