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 analyzing clarification of requirements over time. V:ISSUE:LIZER supports managers and developers to identify requirements with insufficient shared understanding, 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. (Demo video: http://youtu.be/Oy3xvzjy3BQ).**

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

I. INTRODUCTION

Large software projects often require collaboration across geographically distributed sites and depend on online communication to perform activities related to clarifying requirements. More frequently, teams are employing agile approaches that aim to discover 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 its 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.

Communicating 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 is lacking a sufficient understanding of the underlying requirement. Often, stakeholders continue to clarify the requirement because it is ambiguous, incomplete, or has frequent changes. As a result, the expected evolution of a requirement from an initial idea, through clarification, to design and full implementation, can stagnate and its implementation can be delayed, never completed or in some cases never begin. Current requirements management tools offer little support for identifying requirements that cause 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 by visualizing the progression of clarification communication relative to other communication related to a particular requirement, thus providing support in 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 developed the tool iteratively and through a number of evaluation sessions with software practitioners. After a first case study to check the correctness of our communication classifier with the IBM Rational Team Concert data (see [4]), we performed two rounds of tool evaluation with software managers to understand how useful the tool is in their work. The visualization of stakeholder social networks for each requirement discussion was added after such evaluation.

II. BACKGROUND AND RELATED WORK

In this paper, we use the term *requirement discussion* to refer to a thread of online communication that is related to a given requirement. A requirement discussion consists of *discussion events*, i.e. contributions to the discussion. We are particularly interested in *clarification events*, i.e. discussion events in which the discussant seeks to improve the understanding of the requirement by either asking for clarification or by offering additional information that clarifies the requirement. Many software projects use online tools to store requirement discussions, e.g. issue trackers or task management systems like bugzilla, or jira [5]. In such projects, requirements are distinguished by a certain type of issue (e.g. *user story, enhancement*), and the requirement 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 [6]. Giving managers the ability to find important tasks at the right time can be crucial to project success. Treude and Storey identified the lack of visualizations as one of the most important short comings of today's task management systems [7]. Accordingly, dashboards that

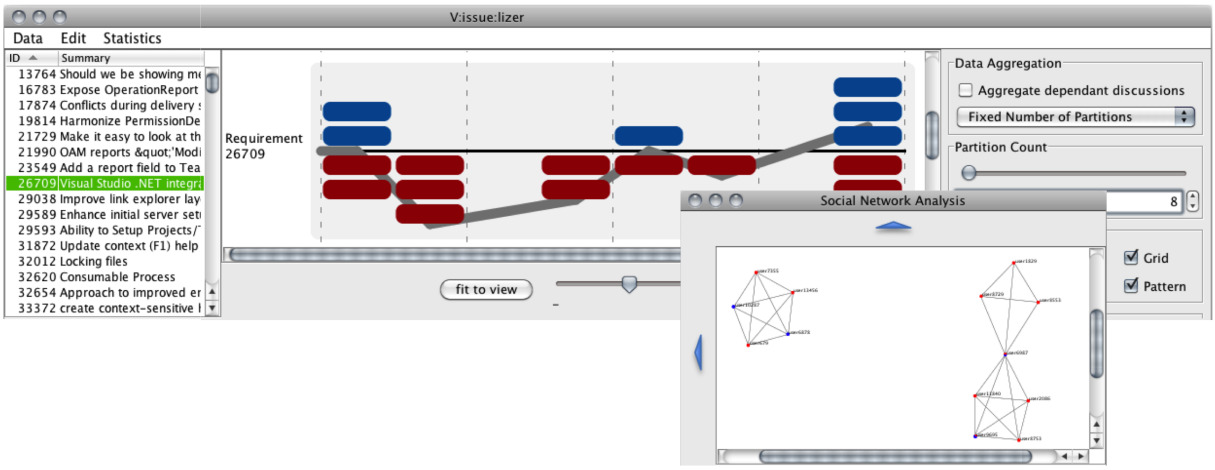

Figure 1: The main window of the V:ISSUE:LIZER shows a list of requirements (e.g. user stories or enhancements in an online repository such as jazz.net, jira, or Bugzilla) and their clarification trajectories. It is complemented by a social network analysis view.

report on the state of a task management system can be pivotal to task prioritization in critical project phases.

Systems like Bugzilla can play a key role in managing software projects, as Ellis et al. [8] report, based on results from interviews of how developers use Bugzilla. The motivation for their study was the design of a visualization tool that reveals social and historical patterns in tasks. However, their tool does not support exploring 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. [9] present the workitemexplorer, a related tool that allows the exploration of information stored in a task management system. Compared to this tool, 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 (Sources: https://github.com/oerich/ReqtDisc). V:ISSUE:LIZER can read data from the IBM Jazz webservice and the RSS feed of Atlassian Jira. Support for other issue trackers can be easily added.

The main window shows a list of requirements on the left (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) that are 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 (e.g. eight intervals in Figure 2), or as a fixed time interval (e.g. days, weeks, month). This allows to adjust the visualization to the particularities of a given process.

V:ISSUE:LIZER offers two 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. Since 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 locate in which part of the lifetime discussion occurs. Discussion events are depicted by rectangles that are shown below the lifeline if they are clarification events, and above the lifeline if they are not. A grey line shows the sum of clarification. In a classic trajectory with clarification up-front and only implementation related communication in the end, this grey line will start in the bottom left corner and raise to the upper right corner. V:ISSUE:LIZER currently uses a Bayesian classifier to identify clarification events, i.e. a supervised machine learning algorithm. Based on this classification, V:ISSUE:LIZER generates trajectories and identifies reoccurring patterns (*textbook-example, indifferent, discordant, procrastination, back-to-draft, and happy-ending*) [4]. Typically, there are many requirements without pathological findings, e.g. user stories with some clarification in the be-

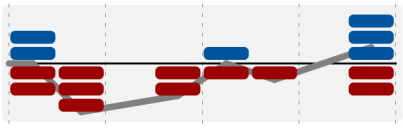

Figure 2: Example of a requirement discussion's clarification trajectory

ginning 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. Others show no clarification at all, even though the requirement seems to be complex.

2) Social Networks: This visualization shows the communication pattern of those participating in a requirement's discussion. Developers are presented as nodes and 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. Subgraphs are connected if stakeholders appear in several time intervals. V:ISSUE:LIZER displays developer nodes as pie charts to show the percentage of clarification vs. implementation-related communication in which a 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 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. This is 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 make an informed decision, based on this rich information, where to invest more resources to tackle requirements that appear to have problematic development.

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 of 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.

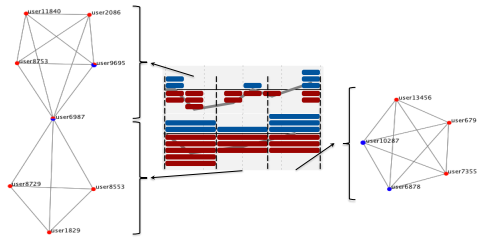

Figure 3: Example of a requirement discussion's social network

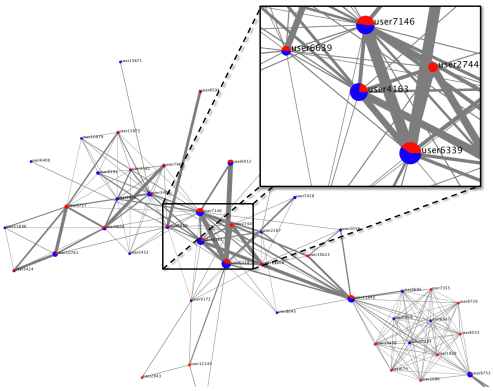

Figure 4: Example of a social network for a set of requirement discussions.

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 one actor who participates in all relevant time intervals. The user decides to assign this responsibility to an experienced developer.

C. Who is knowledgeable about a given topic?

Integrating the right persons into 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: (i) domain knowledge that shows in discussion events related to clarification and (ii) 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 requirement 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 user looks for central developers.

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

In our evaluation we sought confirmation that V:ISSUE:LIZER (1) correctly identifies clarification of requirements in online discussions and (2) provides feedback that is useful to software practitioners.

A. Ability to correctly classify clarification communication

To check the correctness of our communication classifier we performed a case study of communication in the IBM Rational Team Concert (RTC) project [4]. RTC is a globally distributed software project and as such it employs online communication to an extent which guarantees that a sufficient amount of the overall communication is available online. We analyzed 157 requirements and their online discussions (1200 communication events). To create training data for our machine learning algorithm, two raters manually classified 1200 communication events with a high inter-rater agreement ($\kappa \approx 0.7$). We then performed a 10-fold cross validation and measured a recall of 0.94 and a precision of 0.67 (f-measure of 0.789). Consequently, the automatically classified discussion events and the resulting trajectories are comparable with those constructed based on manual classification (see [4] for detailed discussion).

B. Ability to support decisions of managers

Two rounds of evaluations with software practitioners informed the iterative development of our tool. The first version of V:ISSUE:LIZER that solely visualized the clarification trajectory was presented to four software managers at the VIATEC Software Management Round Table meeting in Victoria, BC, Canada. We learned that (1) the clarification trajectories provided useful information to support resource allocation and risk management, as well as project retrospectives and process improvement. More importantly, the managers (2) suggested to add visualizations of communication patterns for each requirement, since a trajectory without much clarification would be suspicious if no experienced developer participated in the discussion. As a result, the social network visualization was added to V:ISSUE:LIZER.

After that, five managers from IBM RTC project were interviewed. The managers found the visualization of clarification trajectories and the associated social networks most useful in the project's *end game* which is the time close to the release where primarily testing and polishing are taking place. At this point, persistence of clarification communication is unusual and indicative of high risk requirements that need management attention.

VI. Conclusion and Future Work

V:ISSUE:LIZER is a visualization tool that allows analyzing requirements clarification in online communication during a project. Specifically, 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 while distributed projects often depend on online communication and challenge their project managers' ability to assess the shared understanding in the team. Without the ability to measure how this clarification diminishes over time, especially before feature delivery, managers find it hard to manage project risks during iterations. Our preliminary evaluation showed that our visualizations allow managers to identify risky requirements where the team has insufficient understanding of requirements, and to further investigate the communication patterns around a particular requirement.

In future work, we plan on deploying our tool in a distributed team and assessing its usefulness over the entire project lifetime. 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 [10] or symmetry of ignorance [11].

References

[1] L. Cao and B. Ramesh, "Agile requirements engineering practices: An empirical study," *Software, IEEE*, vol. 25(1), pp. 60–67, 2008.

[2] J. Aranda, R. Khuwaja, and S. M. Easterbrook, "Discovering the shared understanding dynamics of large software teams," in *Proc. of the 2007 Conf. of the Center for Advanced Studies on Collaborative Research.* ACM, 2007, pp. 1–4.

[3] R. Frost, "Jazz and the eclipse way of collaboration," *IEEE Software*, vol. 24(6), 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] N. Ernst and G. Murphy, "Case studies in just-in-time requirements analysis," in *Workshop on Empirical Requirements Engineering (EmpiRE '12)*, Chicago, USA, 2012.

[6] R. E. Kraut and L. A. Streeter, "Coordination in software development," *Commun. ACM*, vol. 38(3), pp. 69–81, 1995.

[7] 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.

[8] 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.

[9] 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.

[10] C. Jones, "Strategies for managing requirements creep," *IEEE Computer*, vol. 29(6), pp. 92–94, 1996.

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