V:issue:lizer
Explore Online Communication and Requirements Clarification 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 as 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 by offering powerful 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**

I. Introduction

Large software projects are often affected by the need to collaborate across geographically distributed sites and to depend upon 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. Requirements are defined in the form of user stories, and ongoing discussions around 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 in large and distributed projects. IBM®'s Rational Team Concert® project, with a large distributed team, is an example in which management mandates the recording of all decisions in the project repository for future use in the project [2]. Consequently, online project repositories contain a wealth of requirements-related communication.

In these kinds of environments, however, the expected evolution of a requirement from an initial idea, through clarification, to design and full implementation, often stagnates. Often stakeholders continue to *clarify* the requirement because it is ambiguous, incomplete, or has frequent changes. As a result, its implementation can be delayed 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 and differentiating between healthy and problematic patterns of communication associated with an individual requirement. Our V:Issue:lizer helps managers to analyze the content of communication among stakeholders involved in the discussion of a particular requirement, identify specific instances of *clarifying* communication, and examine the trajectory of clarifications (i.e. amount and progression) throughout the lifetime of a requirement. In addition, V:Issue:lizer can visualize social networks that allow to assess the constellation of communication actors and to identify 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 confronted software managers with V:Issue:lizer 's visualizations 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.

II. Background and Related Work

Define "Online Communication" = ticket system, issue tracker, task management system

Use consistent terminology

Harvest RE paper

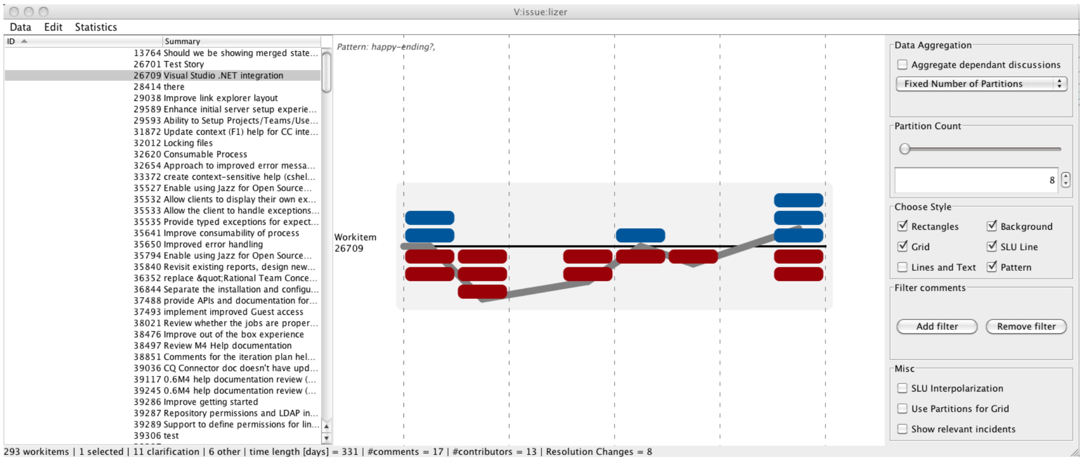
Since tasks crosscut both the technical aspects of collaboration an communication, finding the right tasks at the right time is crucial to the success of a project [5]. Treude and Storey identified the lack of visualizations as one of the most important short comings of today's task management systems [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.

Read this, change text

Ellis et al. [7] report results from interviews of how developers use Bugzilla, a popular task management system. The motivation for their study was the design of a visualization tool for tasks. One of their key findings was that Bugzilla played a key role in managing the project. Their visualization reveals social and historical patterns in tasks, but it does not focus on exploration activities.

Read this, change text

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



V:issue:lizer

ID	Summary
13764	Should we be showing merged state...
26701	Test Story
26709	Visual Studio .NET integration
28414	there
29038	Improve link explorer layout
29589	Enhance initial server setup experie...
29593	Ability to Setup Projects/Teams/Use...
31872	Update context (F1) help for CC inte...
32012	Locking files
32620	Consumable Process
32654	Approach to improved error messa...
33372	create context-sensitive help (cshel...
35527	Enable using Jazz for Open Source...
35532	Allow clients to display their own ex...
35533	Allow the client to handle exceptions...
35535	Provide typed exceptions for expect...
35641	Improve consumability of process
35650	Improved error handling
35794	Enable using Jazz for Open Source...
35840	Revisit existing reports, design new...
36352	replace &quot;Rational Team Conce...
36844	Separate the installation and configu...
37488	provide APIs and documentation for...
37493	implement improved Guest access
38021	Review whether the jobs are proper...
38476	Improve out of the box experience
38497	Review M4 Help documentation
38851	Comments for the iteration plan hel...
39036	CQ Connector doc doesn't have upd...
39117	0.6M4 help documentation review (...
39245	0.6M4 help documentation review (...
39286	Improve getting started
39287	Repository permissions and LDAP in...
39289	Support to define permissions for lin...
39306	test

Pattern: happy-ending?,

Workitem
26709

Data Aggregation
☐ Aggregate dependant discussions
Fixed Number of Partitions

Partition Count
8

Choose Style
☑ Rectangles ☑ Background
☑ Grid ☑ SLU Line
☐ Lines and Text ☑ Pattern

Filter comments

Misc
☐ SLU Interpolarization
☐ Use Partitions for Grid
☐ Show relevant incidents

293 workitems | 1 selected | 11 clarification | 6 other | time length [days] = 331 | #comments = 17 | #contributors = 13 | Resolution Changes = 8

Figure 1: Screenshot: the main window of the V:ISSUE:LIZER shows a list of issues (e.g. user stories) and their clarification trajectories.

concerned with visualizing and exploring this information space. Treude et al. [8] present the workitemexplorer, a highly related tool that allows to explore the information stored in a task management system. Compared to this work, V:ISSUE:LIZER allows the analysis of a very specialized aspect, 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 users to dynamically explore data from a software development task management system with a focus on communication and clarification of requirements related issues. The main control element shows a list of tasks (e.g. workitems in jazz, items in jira, issues in other systems) (see Figure 1). The power of V:ISSUE:LIZER is to add visualizations to the selected issues that help to assess the communication in comments related to these issues.

V:ISSUE:LIZER currently supports two different visualizations:

- **Clarification Trajectories** show how the percentage of communication related to clarifying requirements changes over the lifetime of the issue.
- **Social Networks** show who is participating in a discussion related to (a set of) issue(s) and how the actors in the discussion are structured.

IV. EXAMPLE SCENARIOS

In this section we describe examples that highlight the functionality of the V:ISSUE:LIZER tool.

A. *Where are the hotspots in a set of issues?*

Often, a clear understanding of requirements only evolves during the development of software. This is especially true (but not limited to) agile software projects, where managers decide to frame only rudimentary requirements and refine the details on the go. For a manager, it is important to know when problematic requirements surface, because they can have a serious impact on the project. V:ISSUE:LIZER helps managers in this scenario as follows:

1) The manager loads a set of issues (e.g. user stories for the current iteration).
2) V:ISSUE:LIZER automatically analyzes the comments that are related to these issues and that are available in online communication.
3) V:ISSUE:LIZER creates a set of *clarification trajectories* (c.f. Figure 2), one for each issue. Comments that are concerned with clarifying requirements are depicted by red rectangles below the timeline, other comments are depicted by blue rectangles above the timeline.
4) V:ISSUE:LIZER also displays suggestive pattern names for distinctive trajectories (e.g. textbook-example, back-to-draft, procrastination).
5) The manager scrolls through the issues and associated trajectories and decides based on this rich information where to invest more resources.

Typically, there is a number of issues without pathological findings, e.g. user stories with some clarification in the beginning and other communication events later on that show progress. But there are also suspicious trajectories: a large amount of clarification late in the iteration, perhaps even after the issue seemed to be solved. Or no clarification at all, even though the issue seems to be complex.

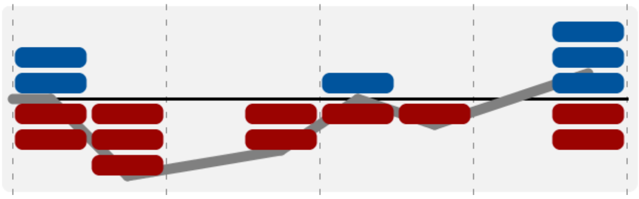


Figure 2: Example of a clarification trajectory

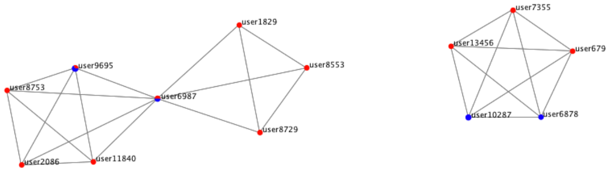


Figure 3: Example of a social network

B. *Are there any communication breakdowns?*

After identifying those hotspots, the manager most likely wants to continue with a closer investigation. Often, he or she will investigate who participates in a discussion of an issue and who is not. V:ISSUE:LIZER helps managers in this scenario by creating social networks for an issue or a set of issues on the fly. More over, V:ISSUE:LIZER also integrates information of the automatic analysis of online communication into these social networks, i.e. showing for each actor the percentage of clarification and other communication. Figure 3 shows an example of an social network for the issue presented in Figure 2. The developers are presented as nodes (here: anonymized), and connections between nodes are weighted by the amount of communication both developers share about a given issue in a specific time interval. In the example above, the manager might conclude that there is no single person who is coordinating the work around this issue, because there is no actor who participates in all relevant time intervals. A suitable action might be to assign the 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 communication events related to clarification and technical knowledge that shows in other communication events. In order to leverage the power of V:ISSUE:LIZER for this task, the manager selects a number of issues that are related to a given topic. V:ISSUE:LIZER integrates the social networks of these issues in a single large network (see Figure 4). The manager looks for candidates with a balanced percentage of clarification and other communication and a medium centrality, actors with high centrality might already have a very high workload.

V. PRELIMINARY EVALUATION

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.

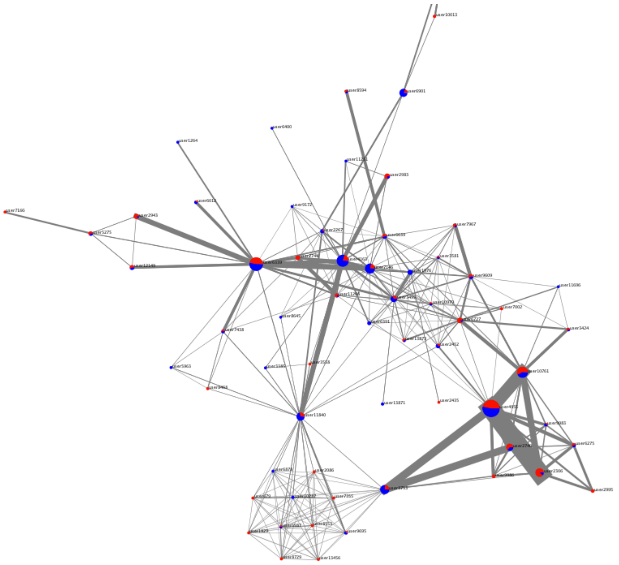


Figure 4: Example of a social network of a set of issues

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 just started to evaluate the ability to support software managers in decision making. For a preliminary evaluation, we presented V:ISSUE:LIZER to several software managers, followed by a semi-structured interview. First, 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 managers of the IBM RTC project and related projects for a second round of interviews. 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

With V:ISSUE:LIZER we introduced a visualization tool to analyze requirements clarification in online communication over time. Especially in 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 project manager's 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 out 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 or symmetry of ignorance.

Todo list

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

[3] C. N. Parkinson, *Parkinson's Law: The Pursuit of Progress*. John Murray, 1958.

[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.