V:issue:lizer
Analyze Requirements Clarification over Time

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

***Abstract*—In current project environments, requirements often evolve throughout the project and are worked on by stakeholders in large and distributed teams. Such teams often use online tools such as mailing lists, bug tracking systems or online discussion forums to communicate, clarify or coordinate work on requirements. In this kind of environment, the expected evolution from initial idea, through clarification, to a stable requirement, often stagnates. When project managers are not aware of underlying problems, development may proceed before requirements are fully understood and stabilized, leading to numerous implementation issues and often resulting in the need for early redesign and modification.**

In this paper, we present the prototype of a tool for analyzing online requirements communication and for supporting our method for the detection and classification of clarification events in requirement discussions. We believe that our prototype will be inspire both researchers and practitioners. Our v:issue:lizer tool enables researchers to analyze data from our previous studies, relate this data to their research, and use our tool to investigate requirements clarification in other research projects. Since a predominant amount of clarifications through the lifetime of a requirement is often indicative of problematic requirements, our v:issue:lizer tool supports project managers to assess, in real-time, the state of discussions around a requirement and promptly react to requirements problems.

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

I. Introduction

In large software projects stakeholders often need to collaborate across geographically distributed sites and to rely upon online communication to perform requirements related activities. Agile software teams in particular implement iterative processes to requirements discovery and use 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]. Whether the time zone differences between stakeholders are too great to allow for frequent synchronous interaction, or the project mandates the recording of project communication online [2], online project repositories contain a wealth of requirements-related communication, making them valuable for research on communication in requirements engineering. 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].

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. Bikeshedding, also known as the Parkinson's law of triviality [3] is another common situation in which developers give disproportionate weight and time to solving trivial issues and delay development. An example from the RTC project (jazz.net) is the ongoing discussion of a large number of developers over the text required in a UI element, and which blocked the development of this user story. Late in the project a manager intervenes and makes a decision "we go with [...] for this iteration", after which development of the story is completed. Although with a happy ending, many situations like this go unnoticed by managers. 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.

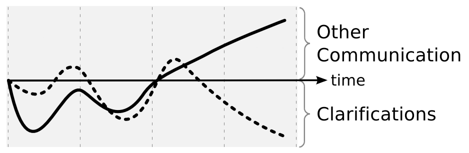

Figure 1: Two different trajectories of reqts. communication

Studying recorded online communication fills this gap by offering the potential to reveal patterns of communication that correlate to problematic situations around requirements development. In this paper we describe a novel approach for differentiating between healthy and problematic patterns of communication associated with an individual requirement. We 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. Figure 1 shows two distinct and quite different possible trajectories of *clarifying* communication in the lifetime of a requirement. While one would expect clarification to diminish as development of a requirement nears the end (solid line trajectory), its predominance throughout the requirement's life may be indicative of problematic requirements (dashed line trajectory). The method proposed in this paper aims to identify these patterns automatically so that managers or involved stakeholders can be made aware of requirements that should be closely investigated.

The contribution of this paper is the method for the detection and classification of clarification communication patterns, as well as a set of six communication patterns that we identified by applying our method in a large industrial project. The remainder of the paper is structured as follows: Section **??** surveys related work in the study of communication in requirements engineering (RE), as well as related to information retrieval and automated classification techniques in RE. Section **??** introduces our research approach in studying requirements communication and the set of our techniques for the detection and classification of clarification patterns. We then describe the details of the industrial case study in which we applied our techniques in Sections **??**, **??**, **??** and **??**, and conclude with future research steps in Section **??**.

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.