Story-Wall: A Concept for Lightweight Requirements Management

Lorena Delgadillo and Orlena Gotel
Department of Computer Science, Pace University, New York
lorena.m.delgadillo@gmail.com; *ogotel@pace.edu*

Abstract

Most commercial requirements management tools are costly, document-driven and used by organizations undertaking traditional forms of software development. While not immediately in the spirit of agile, which advocates live dialogue over documentation and encourages developers to do the simplest thing possible, there are some practices supported by these tools that also play a role in more agile forms of software development. This work examines the requirements management needs that are common to software development of all flavors. The session will illustrate a concept designed to bring lightweight requirements management to the agile context and seek interactive feedback from RE'07 participants. The concept is based on experiences of agile development within ibm.com and on the transition from handling paper-based story cards to the use of first generation story management tools. The prototype and its validation is work in progress.

1. Introduction

Whether a software system is being built according to traditional software development processes (e.g. waterfall-based and plan-driven) or following an agile development process (e.g. eXtreme Programming [1]), we claim that the fundamental requirements engineering activities are basically the same. Stakeholders need to be identified, candidate requirements or stories need to be determined, discussion and negotiation has to take place to check, agree and prioritize these, and some mechanism needs to be established to monitor progress, ensure satisfaction and deal with changes. The major difference between paradigms is primarily the explicit emphasis and support given to each activity and the importance placed on the documentation of such.

2. Core requirements management needs

Requirements management is the process through which requirements, both technical and non-technical, are controlled throughout the project life cycle [2]. It provides the ability to allocate development tasks, see which requirements have been implemented and to assess the impact of changes.

A study was conducted of over 20 requirements and story management tools to determine the common underlying features. At a high-level these include: requirements/story storage, prioritization, change control, progress visibility and traceability. In consultation with practitioners to determine their needs, those tools specifically designed to support traditional practices were found to be heavyweight in their configuration and process requirements, a deterrent for agile teams requiring a low start-up cost and configurability. Further, those tools explicitly designed to support agility actually appeared to lose many of the benefits of this particular way of working.

3. Story-Wall concept

A concept was developed to investigate ways to provide lightweight requirements management support tailored to the agile (predominantly XP) context. In addition to the core requirements, it was to support the working practices of a team of developers working on a globally distributed project and focus on accessibility for a non-technical customer. The main features of Story-Wall are described below.

Story Card Simulation. Paper-based story cards are limited in size according to the dimensional constraints of the physical index cards used. Many agile tools, while preserving the concept of a card, allow the text within it to expand. This reduces some of the power of the metaphor since a small area forces the story to remain tentative and to act as a prompt for discussion, as opposed to conveying finality. In addition, both the front and back sides of the card get

used. Within the team targeted by this work, the details of the story are written on the front and the back is used to show the different tasks the story involves and/or the test cases for the story. These concepts were all preserved by Story-Wall.

Virtual Story Wall. Both the customers and developers of XP projects rely on story cards being placed on a physical wall to see what needs to be or what has been done on a project, and to easily move a story between stages of development. It provides for on-going situational awareness. Unfortunately, cards can fall off walls, get mislaid, and the ability to share and collaboratively manipulate cards on a physical wall does not translate across distances. The affordances of such a story-wall were lost by the tools we surveyed. Upon opening a project in Story-Wall, the status of the wall is displayed, the current iteration being centralized in the context of the end-to-end project timeline. Users are able to scroll to different iterations and view the wall at moments in time for traceability, acting as an index into more detail. Using drag and drop functionality, users can drag story cards from stage to stage until customer acceptance. Priority and sizing information is also visually represented.

Prioritizing Stories. A drag and drop interface was created to enable customers to sort their story cards into priority lists according to perceived business value. The idea of adding numbers to cards to depict priority, or levels such as 'high', 'medium' or 'low', did not always reflect what happened with the physical cards in practice. Story-Wall provides a way for customers to physically order their cards via direct manipulation into piles. Visual cues as to relative priority are then used when placed on the wall.

Sizing Stories. Sizing is the anticipated development effort to implement a story. Although practitioners write this estimate on an index card, it was remarked as sometimes easier to estimate stories relative to others rather than to give an absolute number. Story-Wall provides a mechanism to physically stretch a story card so that the area represents the estimate. The team can physically manipulate cards to assist consensus building.

Iteration Selection. Typically, the customer uses the priority and sizing information to select those stories to be developed per time-boxed iteration. The constraint is the velocity or total amount of effort that the developers have available. Stories are physically selected for iterations within Story-Wall by filling a container proportionally sized to reflect the velocity. This way it is possible to visually see whether the iteration is being filled with small effort tasks implementing features of low value or high value stories requiring large effort. Such cues can assist the customer in undertaking the task and the development team in organizing their subsequent work.

Lightweight Traceability. 'Traceability' is a little used term within some agile communities, even though some contexts appear to demand the need for a mechanism to support project longevity, scale and distribution. Lightweight traceability can be achieved within Story-Wall as a by-product of use. It keeps track of the changes made to a story by adding a simple history to the card. Annotation and comment facilities are provided, with font types differentiating contributors, as per a physically annotated card. This offers a visual way to see who has worked on what and provides a tracing dimension. Story details can be traced back to the associated discussions between customers and developers by recording a meeting or phone conversation, exploiting web 2.0 technologies.

4. Ongoing and future work

The concept is in prototype phase. While it has been validated with a small number of practitioners from IBM, it requires refinement and feedback from others. The platform on which the tool is being developed is wiki-based, and a number of web 2.0 technologies, AJAX and Ruby on Rails were chosen for implementation. The requirements management support is minimal, this work being an initial attempt to highlight the benefits it can bring to a targeted agile team. Ongoing work is examining how to maintain traceability between connected stories, to better support the story derivation and discussion process, and integration with subsequent development artifacts.

5. Acknowledgements

The authors would like to thank David Leip, the *ibm.com* Agile Methods Advocate, and Joe Krebs from the Rational brand. This undergraduate work is supported by a Pace University Eugene M. Lang Student Research Fellowship Presidential Grant.

6. References

[1] Beck, K. *Extreme Programming Explained: Embrace Change*, Addison-Wesley, 2001.

[2] Hull, M.E.C., Jackson, K. and Dick, J.J. *Requirements Engineering*, Springer-Verlag, 2002.