Enterprise SoBA:
Large-scale Implementation of Acceptance Test Driven Story Cards

Patrick Onions, Chetankumar Patel
Leeds Metropolitan University's Centre for Project Management and School of Computing
p.onions@leedsmet.ac.uk, c.patel@leedsmet.ac.uk

Abstract

Agile software development has proven to be an effective alternative to the regimentation of waterfall-based approaches. Recent research by Patel and Ramachandran [17] has led to the innovation of Acceptance Test Driven Story Cards (SoBA). Implemented using Extreme Programming, preliminary findings show SoBA retains essential Agile attributes like feedback, flow and universal responsibility for quality [3] whilst delivering improvements to estimation, communication, requirements gathering and testing.

Original SoBA theory was exploratory, and its validation was experimental systems development. Enterprise software on the other hand is more complex and architectural in its design. This paper suggests integration with architecture, project management and business analysis practices in order to expand SoBA's potential and address criticisms of Agile methods and Extreme Programming; particularly instability of requirements, user conflicts and scalability.

Enterprise SoBA is untried. This paper proposes high-level abstract concepts and integration, and these will be tested during the course of ongoing research.

Keywords

Extreme programming, story card based agile software development, acceptance test driven story cards, SoBA, enterprise software development, project management

1. Introduction

Preliminary research into the Acceptance Test Driven Story Card (SoBA - Story card Based Agile software development) [17] has revealed this concept to be highly effective in supporting requirements elicitation. SoBA enhanced the traditional story card [2] through introducing a focus on stakeholder involvement, adding acceptance testing and other details to the story card, and modifying the requirements elicitation process. These innovations and enhancements were intended to improve estimation, improve communication between customer and developer, identify verifiable functional and non-functional requirements, support gap analysis, enhance the consistency of documentation, and integrate testing into requirements gathering. Initial empirical experiments have revealed this to function correctly and deliver intended benefits.

Initial empirical experiments have revealed the original SoBA concept to function correctly and deliver intended benefits. Preliminary work focused on theory development and empirical testing in three small-scale software development projects.

Enterprise software development however tends to display complexities that defy conceptually solid methods. Analysis reveals further enhancements could improve SoBA's suitability for large-scale systems whilst retaining its key attributes and benefits. This paper proposes to address scalability, flexibility, management and control, concern for all stakeholders, integration with development and implementation processes, version control, configuration or change management, and intellectual asset management. Areas of integration and implementation will be considered but not prescribed since this is ongoing research and so as to remain flexible and relevant to a broader base of applications.

2. Agile requirements engineering practice

This paper will treat Extreme Programming (XP) concepts and practices as illustrative of Agile methods. It will span the breadth of requirements engineering practice, which SoBA has been designed to effectively encompass.

Requirements engineering is divided into three practices in XP: the planning game, release planning, and iteration planning [2]. Usually projects comprise several iterations for each release, but iterations can also be released individually depending on project scope [2]. Figure 1 below summarises XP's requirements engineering practice.

Requirements elicitation is undertaken during the planning game, and responsibility for this activity lies largely with the client. System requirements are collected using story cards written by the customer, not by a programmer. The customer defines the value with the intention of the programmer building that value.

978-1-4244-4116-7/09/$25.00 ©2009 IEEE

Requirements analysis is divided between the planning game and iteration planning. Negotiation of requirements is mainly undertaken in the planning game, by prioritising requirements with customers aiming to maximise their value for every function [11]. This typically involves iterating through the customer defining the values, programmer estimating, customer selecting value based on business priority, and programmer builds.

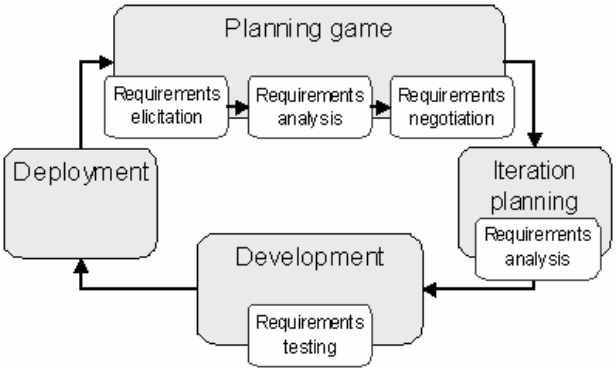

Figure 1. Agile requirements engineering

Requirements validation is not formally included in XP practice according to Kolehmainen [14]. Onsite customers do however provide insurance by allowing missing or contradicting requirements to be swiftly analysed and the iteration steered accordingly ([1, 5, 12, 15]).

3. Story Card based development

The concepts of the user story and story cards require definition for the requirements elicitation process to be understood; and prioritisation, arbitration, iteration and estimation concepts need to be embraced.

A story refers to the user requirements, and is both a promise to be converted and a unit of development and deployment [2]. A story should be understandable to customers and developers, testable, valuable to the customer, and small enough for programmers to build perhaps half a dozen per iteration ([2, 6, 7]).

Users have a right to maximum value from every programming moment, and the programmer has the right to know what is needed. These two rights come together in the user story [11], a unit of functionality in an XP project defined by the user (customer). Each user story is a short description of the behaviour of that chunk of functionality ([2, 7, 11]), and a medium of analysis and medium of communication between customer and programmer [11].

Stories are interrelated and display hierarchies. Dependencies in user stories or story cards should be avoided as they may lead to prioritization and planning problems [5]. Whilst choice of story implementation depends on the customer, inconsistencies may emerge if high priority stories are dependent on low priority stories. Arbitration may also be necessary where more than one stakeholder has an interest in the functionality.

The story card is the document on which stories are recorded. Cohn [5] suggests that story cards need to be independent, valuable, testable, negotiable and small enough to fit into the iteration. Story cards are divided into task cards that represent discrete chunks of functionality. The pair programmer signs off task cards during test driven development, and the functionality is continuously deployed (integrated) following unit testing (Figure 2).

Story cards are crucial in supporting each and every XP practice either directly or indirectly. They are an element of the planning game, supporting the building of a release plan based on the length and number of stories. Story cards provide proof of a conversation over requirements, thereby increasing the communication among the team. Simple communication systems provide quick feedback from customers and this increases productivity.

Story Card Based Agile Software Development (SoBA) has made a number of unique contributions. These include a standardised formal template that was previously lacking, and adding testing to requirements gathering by incorporating acceptance test information to the story card.

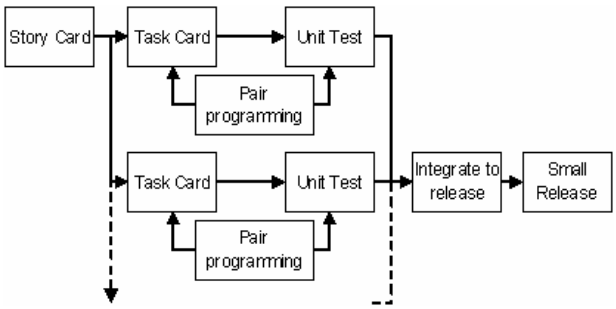

Figure 2. Story card support for XP

A range of project relevant factors would determine iteration size or duration, including business requirements, complexity, release frequency, uncertainty, performance measurement and release overhead [6]. Progress is demonstrated by the development team delivering tested, integrated code that implements a story.

Estimation techniques are largely experience based, with attendant problems that plague the waterfall model of development, although experience can be supplemented by using a spike. The spike is simply a quick, throw away proof of concept to allow the developers to gain an insight

into what's really required to implement a story and hence provide a more accurate estimate for the story. Implementers should however acknowledge and resist the temptation to allow the spike to become the solution.

4. Enterprise development requirements

Enterprise software development can involve hundreds of people with a variety of roles, engage with thousands of users, deliver millions of lines of code, and entail man-decades of effort and years of elapsed time. Failure to address such demands of this environment can lead to 'cowboy' practices or narrow viewpoints and unresolved progress [13]. SoBA enhancements should therefore address scalability whilst retaining those qualities that make the method so effective. These enhancements will be considered in terms of implementation and integration.

Successful implementation of new concepts like Agile and SoBA into an organisation requires attention to be paid to the design of the innovation and to the transfer of knowledge and skill into the organisation. Froese [9] found that successful adoption requires an awareness of the innovation and an understanding of it and how it may be operated. Adoption can lead to upheaval and a decline in performance [4]. New technologies can challenge the organisation, and success depends on rapid integration into existing structures [20]. Peterson [18] finds that innovations should be tailored to suit the organisation and its users, whilst Rogers [19] is of the opinion that the organisation should adapt to the innovation.

5. Enterprise enhancements

Although SoBA has been designed for requirements elicitation, it has application and impact throughout the software development lifecycle. Since enterprise scale software requires meticulous design, its development should involve meticulous planning and control. Design of enhancements has therefore focused on integrating and playing a fundamental supportive role in three areas: architecture, project management and business analysis.

5.1. Architectural enhancements

The interdependence of a multitude of systems within an enterprise imposes constraints and opportunities on the software development process that cannot be ignored. One means of addressing these obligations is through integrating with enterprise systems architecture efforts (this paper loosely adopts the ANSI/IEEE 1471-2000 view of enterprise architecture: the strategic and conceptual design of systems components, inter-relationships and guiding principles). Four suggestions are offered here to integrate with architecture: taxonomies, domain analysis, information resource management (IRM) and indexing of stories in an enterprise repository.

Enterprise SoBA should be underpinned by a robust taxonomy of concepts. Taxonomies ensure consistency of terminology and clarity of meaning across systems and between people, and variation in customer or business terminology can either be interpreted correctly or educated out in both users and programmers. Typical sources of taxonomy data include an enterprise dictionary, meta-data or data derived from sources such as data warehousing.

Enterprise SoBA should also be underpinned by domain analysis. This activity would identify systems, common and variable components, as well as provide a framework by which to solicit and structure requirements elicitation and development. The relationship between story cards and domain analysis is described in Figure 3 below. SoBA would be compatible with a range of domain analysis methodologies, such as those outlined by Frakes and Kang [8].

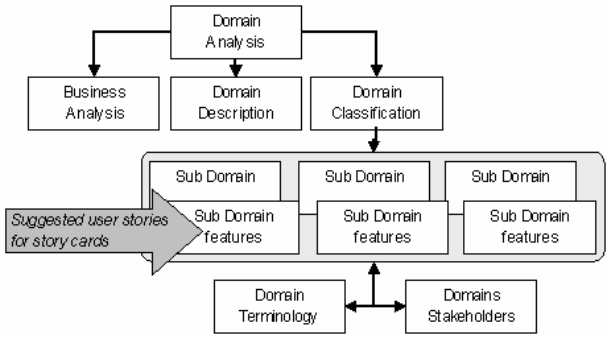

Figure 3. Relationship with domain analysis

Large information systems are usually integrated with other systems and functionality, even to the extent of sharing components such as through Service Oriented Architecture (SOA). The ability to locate a particular story quickly and across project and business boundaries will be valuable in reducing reinvention and improving reuse. This would entail a single common enterprise repository for all users; where stories can be easily added and retrieved even amongst thousands of stories and even if descriptions and terminology differ from the search.

In practice this would entail changes to analysis and story cards. Analysts should be informed by the taxonomy, and try to map their work to it. They could work in or across domain areas, but always with an awareness of where the story card would be positioned and where the functionality may describe reusable services. Story cards should be numbered to reflect the overall enterprise architecture, with indexes linked to domain models, meta-data, thesauri and other information resource management (IRM) knowledge bases.

5.2. Project management enhancements

Enhancements are required to improve the control of large scale projects and mitigate the causes of some criticisms of Agile and XP practices, particularly scalability, cowboy coding, progress tracking and risk management [13, 21].

Most software development is planned and controlled using project management. Irrespective of whether SCRUM, spiral, V or waterfall lifecycles are used, management is reliant on the breakdown of work and on attributing time, cost and quality dimensions to that decomposition. Story cards naturally record a decomposition of requirements, so they are ideally suited to supporting project management through integration with several key project management tools:

- The project work breakdown structure (WBS) decomposes activities into manageable, measurable and discrete parcels of work. If development treats each story card or task card as a parcel of work, then the WBS simply maps the story card hierarchy and eases the project planning process.
- Adding names of stakeholders, contributors, programmers and testers to each story card allows a detailed project organisation breakdown structure (OBS) to be constructed. Resourcing of the project is then improved when this information is combined with the WBS information.
- The encapsulated and granular nature of story cards improves estimation. Aggregation of estimates is simplified, and scheduling would benefit from using the WBS and OBS information to reduce contention for resources.
- As long as development follows the story cards, progress tracking becomes a matter of confirming the completion of development and testing of each story card or its task cards and unit tests. Story card functionality is customer functionality, so progress would also be visible to both customer and supplier.
- Progress may now be plotted on an earned value curve based on quality, traditional cost based measurement like S-curves, or other techniques.
- Risks noted on each story card may be transcribed or automatically extracted into the risk register. This may provide the project manager with an unusual and actionable level of risk detail.
- Test plans may be informed by and constructed from acceptance test criteria recorded on the story card. This supports very detailed management and crystallises the requirements capture.

5.3. Business analysis enhancements

The discrete nature of story cards is a primary cause of many criticisms of agile methods and XP, particularly instability, user conflicts and scalability. A number of enhancements to story cards are proposed here to improve adoption, integration, management and control, and stability of requirements.

Standardisation is an effective means of driving process efficiency, information reuse and understanding. Stakeholders from across the organisation inform functionality, and systems and processes impose on the design. Elicitation should be specific and accurate, yet it should also be tailored to the needs of the organisation and the system's context. A choice of story templates may be offered to guide the analyst and subject expert.

Stories must be managed, in their writing, storage and recollection. A broad range of users such as business and systems analysts, architects, developers, testers, trainers, project managers and even auditors may need to access requirements information at various times during and after the project. Story cards therefore need to include the project, technical and business context of the functionality on the card. The enterprise repository system discussed above should allow users to index story cards with appropriate security and privileges to create, retrieval, update and delete information.

Scale and complexity of enterprise systems presents problems for most requirements elicitation techniques. Increasing detail is encountered as the story emerges, and the project may also be viewed or filtered at different levels of detail. A hierarchy of story cards should emerge, and links with domain analysis maps recorded. The relationships between story cards should also be mapped.

Requirements emerge, need refinement and change over time and the story card is seen as a record of that evolution. Story cards should be subject to approval mechanisms. They should also be version controlled, using a version number that is recorded in a configuration management register or change control system.

Tests have already shown that the story card is compatible with diagramming and modelling techniques (i.e. UML). Organisations should choose an appropriate language by which to document requirements, to support rather than solely rely on natural language.

Externalising and socialising knowledge [16] may be difficult or even be impossible in the case of global organisations. A personalisation knowledge management strategy [10] may be adopted to treat people rather than documents as the repositories of knowledge. This would overcome communication and interpretation barriers involved with complex problems, and support future communication and collaboration. Personalisation may be implemented by recording those participants who were involved in preparing each story card.

6. The way forward

This paper has identified three areas in which SoBA concept could be enhanced to make it more suitable for large scale and enterprise information systems development, outlined in Figure 4 below.

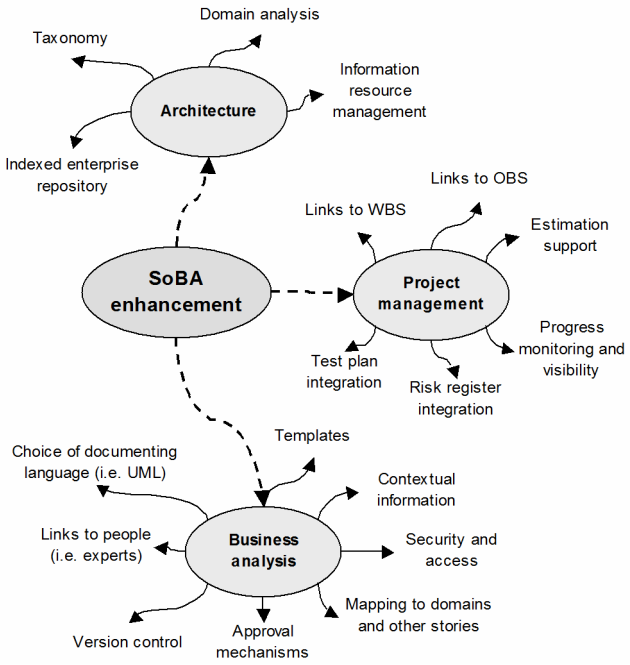

Figure 4. Outline of proposed enhancements

The SoBA enhancements are intended to improve communications, management and practice. Implementation has purposefully not been specified for two reasons. Firstly, this is preliminary research. Secondly, experience shows that practices and tools vary widely between organisations; so prescribing implementation mechanisms may limit or deter potential applications.

Research now intends to find suitable projects to pilot Enterprise SoBA. This empirical work will evaluate efficiency, effectiveness and best practices for implementation; and illustrative examples and case studies describing integration will be published.

7. References

[1] K. Beck, "Embracing change with extreme programming", *IEEE Computer*, 1999, pp. 70-77

[2] K. Beck, *Extreme Programming Explained: Embrace Change*, Addison-Wesley, 2000

[3] K. Beck, "Tools for Agility", *Microsoft White Papers*, Microsoft Corporation, available online at www.microsoft.com, June 2008

[4] D. Briscoe, "Laying the Groundwork for Strategic Technology Deployment", *Report on Legal Management*, Altman Weil, 1999

[5] M. Cohn, *User Stories Applied: For Agile Software Development*, Addison-Wesley Professional, 2004

[6] M. Cohn, "Selecting the Right Iteration Length for Your Software Development Process", *InformIT Network*, available online at http://www.mountaingoatsoftware.com/, March 2006

[7] M. Fowler, "Put Your Process on a Diet", *Software Development*, Vol. 8 No. 12, December 2006, p.32-36.

[8] W.B. Frakes and K. Kang, "Software Reuse Research: Status and Future", *IEEE Transactions on Software Engineering*, Vol. 31 No. 7, July 2005, pp. 529-536

[9] T. Froese, "Vera - Information Networking in the Construction Process: A TEKES technology programme", *Dept. of Civil Engineering*, University of British Columbia, Vancouver, Canada, 2002

[10] M.T. Hansen, N. Nohria and T. Tierney, "What's your strategy for managing knowledge?" *Harvard Business Review*, Vol. 77 No. 2, 1999, pp. 108-116

[11] R. Jeffries, A. Anderson, and Hendrickson, *Extreme Programming Installed*, Addison-Wesley, 2000

[12] J. Kääriäinen, J. Koskela, P. Abrahamsson and J. Takalo, "Improving Requirements Management in Extreme Programming with Tool Support - an Improvement Attempt That Failed", "Software Process and Product Improvement" track, *Euromicro 2004*, Rennes, France

[13] C. Keith, and M. Cohn, "How to Fail with Agile: 20 Tips to help you Avoid Success", *Better Software*, July/August 2008

[14] C. Patel and M. Ramachandran, "Story Card's Values Oriented Prioritization Matrix for XP (Agile Software Development)", *International Conference on Software Engineering Theory and Practice* (SETP, 2008), Orlando, FL

[15] C. Patel and M. Ramachandran, "Bridging Best Traditional SWD Practices with XP to Improve the Quality of XP Projects", *International Symposium on Computer Science and its Applications*, Australia, 2008

[16] I. Nonaka and H. Takeuchi, *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*, Oxford University Press, New York, 1995

[17] C. Patel and M. Ramachandran, "Acceptance Test Driven Story Card Development for XP (Agile Software Development)", *Proceedings of the International Computer Science and Technology Conference* (ICSTC 2008), San Diego, California, 1-3 April 2008

[18] D. Peterson, „Mobila system kraver mobilt tankande", *Computer Sweden*, 30 April 2003

[19] E.M. Rogers, *Diffusion of Innovation*, The Free Press, 1995

[20] N. Saranummi, I. Korhonen, M. van Gils and S. Kivisaari, "Barriers limiting the diffusion of ICT for proactive and pervasive health care", *VTT Information Technology*, Tampere, Finland, 2001

[21] M. Stephens and D. Rosenberg, *Extreme Programming Refactored: The Case Against XP*, Apress LP, Berkeley, California, 2003