Agile Scrum

Kenneth R. Walsh
University of New Orleans, USA

Sathiadev Mahesh
University of New Orleans, USA

INTRODUCTION

Agile Scrum is an iterative, incremental way to develop software. It was developed as an alternative to more traditional methods, such as waterfall development, which have in some cases been viewed as time consuming and fraught with delays. With Agile Scrum, development periods are short and iterative, allowing the customer to see the product as it grows, instead of having to wait until all development is completed (Schwaber, 2009). Agile development of software follows the principles laid out in the Agile manifesto (Agile Manifesto, 2012; Stewart, Chapple, & Gibson, 2012) which are, (1) customer satisfaction through early and continuous delivery, (2) frequent face-to-face conversation between developers and customers, (3) willingness to adjust designs based on customer's changing needs, (4) simple and uncluttered designs, (5) a healthy and sustainable pace of work during the entire development process, and (6) periodic episodes of deep reflection to make the team more effective. The approach focuses on transparency that results from customer knowledge of the true state of software development, and adaptation to changes based on dynamic market needs and cost and time influences. Scrum succeeds only when customers are actively involved with the project during its development.

Scrum was developed at Easel Corporation by Jeff Sutherland, John Scumniotales, and Jeff McKenna and is named after the process used in rugby involving close-knit team formation when a game is restarted after an accidental infringement (Rising & Janoff, 2000; Ambler, 2008; McKenna, 2013). One view of scrum is that frequent and intense feedback impact the team and improve productivity favorably, moving the development team and customers from their normal state of inertia to a high velocity operation where the eco-system settles into a "hyperproductive" state (Downey & Sutherland, 2013). The charts developed in scrum to show the rate at which tasks have been completed visually display the velocity of production. Members of a team have mental models about the status of the project and their work environment. Performance is improved when all members of a team share the same mental models about the project and workspace. Scrum improves collaboration and communication among team members and customers and helps alleviate bottlenecks in production (Santos & Moura, 2012).

BACKGROUND

Scrum is suitable in turbulent business situations where change must be embraced rather than rejected. Scrum is beneficial in situations where initial cost and time estimates are unreliable. The project manager can dynamically adapt to changes in cost and time using scrum (Chen, van den Akker, Brinkkemper, & Diepen, 2010). While scrum is commonly recommended for smaller projects, it can be integrated with approaches used for large projects such as the Capability Maturity Model (CMM) (Sutherland, Jakobsen, & Johnson, 2007; Marcal, de Freitas, Furtado-Soares, & Belchior, 2007). Roadmaps to link scrum practices to the CMMI model and provide a Mature Scrum approach are presented by Lucasiewicz and Miler (2012) and Moksen and Mensely (2012). Since CMMI enforces discipline in software development throughout the organization, and institutionalizes processes in the business, scrum benefits from a workforce that can adapt to and abide by the rules of the system. It is important to understand that scrum is not undisciplined hacking, but is

DOI: 10.4018/978-1-4666-5888-2.ch691

systematic development of a product within a dynamic environment.

Scrum has been combined with extreme programming (XP) and the Unified Software Development Process (Zhang & Patel, 2011). While standard scrum has sandboxed sprints, with one sprint completed and reviewed before the next sprint, an overlapping sprint approach (Sutherland J., 2005) has been proposed by Jeff Sutherland, one of the developers of the original scrum approach. Sutherland describes type B and type C scrums where each sprint has tasks that stage work for a subsequent sprint. This reduces the time used to update and prepare the newly prioritized product backlog at the start of each sprint.

Waterfall development, using the Systems Development Life Cycle (SDLC) is a popular approach for building large software projects. Waterfall development begins with the concept, moves on to design, implementation, testing, installation, and troubleshooting, and ends with operation and maintenance. The Customer, who has a great deal of input at the beginning of a waterfall development, has no further input until the product is released. One rule of waterfall development is to freeze changes at the design phase when specifications for an entire system have been completed. After that, the working software is only released when all specifications have been met, delaying the time when the customer will see the product. In Agile Scrum development, the customer is involved throughout the process and can suggest changes at the end of each short sprint, often every two to six weeks. In addition, the customer can interact with a usable product at the end of each sprint, and can choose to continue or end development.

Scrum is based on close team interaction as well as repeated user-developer interaction. Most scrum implementations have been in small collocated groups. However, with the rapid growth of global software development using distributed teams, there has been a lot of interest in implementing scrum in globally distributed teams. Face-to-face interactions offer rich communication support and virtual work spaces do not present the degree of richness necessary for scrum development. The major challenges of implementing scrum in virtual meeting environments are information transfer between the product owner and the scrum team in Scrum Planning and Scrum Review Meetings, challenges in whiteboard technology used for interaction when building and discussing scrum visual elements such as scrum Burndown charts, and knowledge transfer during peer communication between team members (Luz, Gazineu, & Teofilo, 2009). Lightweight Scrum (Lwei-Scrum) has been shown to enhance productivity significantly, with a six fold increase in productivity measured in function points developed per person-month, and a similar reduction in acceptance testing defects in a comparison with waterfall development in the same organization (Zafar, Sattar, & Mustafa, 2010). However, it is important to note that the improvement did not compare work on the same project, but on completely different projects.

SCRUM PROCESS

The scrum process is initiated by an executive level decision to engage in the development of a new product. Executive management identifies the project and assigns a Product Owner, the person responsible for both the technical and business success of the project. The product owner works with a customer liaison group in a series of meetings to create a project plan containing a description of the product to be developed along with a product vision and product personas (user groups with goals, concerns, skills, and usage patterns). After a review and approval of the project plan by executive management, the product owner works with the customer liaison group to develop user stories that describe how the product will be used. Issue tracking software is often used to organize the product owner's work.

Product Owner

The Product Owner must fully understand the critical needs of the business unit and be aware of the value addition to the business process from the project (Saddington, 2012). The project features are prioritized in terms of the value added to the business process. The product owner creates the overall project plan for the product as well as milestones and deliverables. The product owner is responsible for the profitability of the product, often measured in terms of its return on investment. Scrum can succeed only when this job is handled by one individual; and not a team. The product owner servers as the primary link between the stakeholders and the developers and authorizes necessary adjustments to features and priorities after every