Scrum Software Development Methodology

Ruth Guthrie
California Polytechnic State University, Pomona, USA

INTRODUCTION

An important decision in developing software is what methodology to use. Scrum, a very popular technique, is an agile software development methodology that started the early 1990s (Schwaber, Business Object Design Implementation Workshop, 1995). Agile and scrum methodologies are growing popular with software developers because they are seeking ways to reduce cost and delivery times, while maintaining software quality and improving customer satisfaction.

The methods used in scrum were developed to solve problems that existed with other development methodologies, which were seen as slower and inflexible. A particular emphasis was given to meeting user goals and allowing users to change their minds during the development process. By using an approach to software development that creates several small, operational deliverables and allowing requirements more flexibility, scrum promises lower software development costs and shorter development times. With scrum methodologies, flexibility towards changes in user requirements purports to deliver products that achieve high user satisfaction. Scrum is designed to accommodate environments where requirements are unstable and ill defined.

BACKGROUND

In the history of software development, programmers have pursued approaches to development that improve quality, shorten time to market and reduce cost. Major methodologies in software development include waterfall, iterative design and agile development.

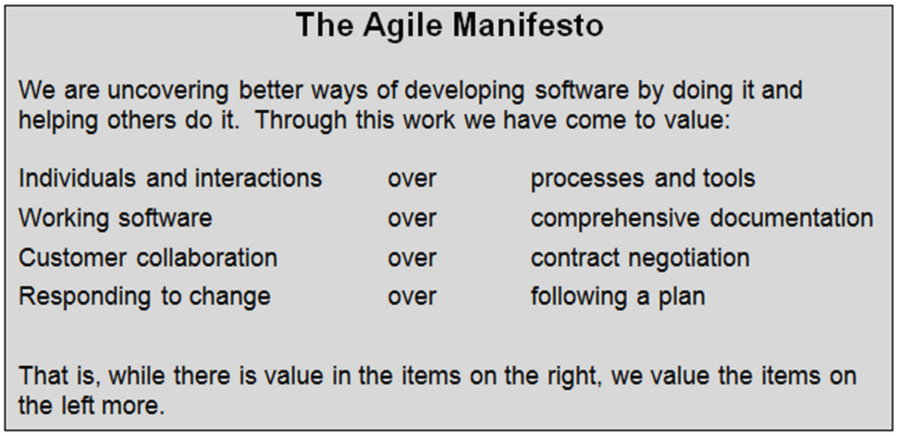
The earliest computer programs followed no methodology, resulting in 'spaghetti code' and unmaintainable systems. As development projects became larger and more complex, the waterfall methodology was introduced (Royce, 1970) to manage large-scale software development projects. Waterfall methodology has an approach to developing software that consists of several stages: requirements, design, implementation, test and maintenance. The process is known for being very rigid and not accommodating to changes that may evolve, or be requested during the project. This development process has many variations but is commonly known as the software development life cycle (SDLC). Requirements are developed and agreed upon at the beginning of the life cycle and at the end of development, teams test to see if they met the requirements. The requirements document represents a contract between the developer and client on what specifically will be delivered. When a change in the software is needed, a requirements change is made, triggering many more changes in code, test and documentation. Controlling the changes and adequately testing the entire system, helps to maintain quality as the software changes. The inflexibility with the waterfall methodology made the importance of correctly defining the requirements clear. It was much more expensive to fix a requirement or add a feature to a large scale program at the end of the life cycle.

Large software projects developed a reputation of poor user satisfaction, late delivery and cost overruns (Yourdon, 2003). The 1995 Chaos Report from the Standish Group reported that approximately 31% of software projects end in failure, and 50% of the projects that succeeded were 189% over their original cost estimates (The Standish Group, 1995). People began to look at alternative development methods to improve on cost, schedule and satisfaction.

Barry Boehm proposed a more iterative development methodology known as the spiral development process (Boehm, 1986). Phases of the spiral method are: determine objectives, risk mitigation, development and test, and planning. The spiral method breaks development down into several iterations, where issues of planning and risk are continually addressed. The Spiral method allows for requirements changes

DOI: 10.4018/978-1-4666-5888-2.ch719

Figure 1. Agile Manifesto (http://www.agilealliance.org)



to occur more frequently, resulting in lower cost and higher user satisfaction. One disadvantage of the spiral methodology was that is complex and difficult for people to understand.

Agile methodologies arose from iterative strategies, like spiral development, in the early 2000s. The focus of agile methods is to allow for more fluid, rapid requirements changes, continuous user involvement and the use of smaller, self-organizing teams to develop software.

Seeing older methodologies as too rigid, a group of software developers met in Snowbird, Utah, in 2001 to develop concepts for agile development methods (The Agile Manifesto, 2013). They formed the Agile Alliance and defined their goals in what is known as the Agile Manifesto.

In the manifesto (Figure 1), developers focus on meeting user needs through collaboration and change are evident. Their belief was, if a change is needed, it should be made in the interest of developing the best possible product. The focus is on making software work well over following a restrictive contract or adhering to a massive design document.

Among different agile techniques are Agile Unified Process (AUP), Extreme Programming (XP) and Scrum. Scrum's name is derived from a Harvard Business Review article (Takeuchi & Nonaka, 1986) that looked at the success of teams. Takeuchi compared the outstanding manufacturing teams to a scrum in the sport of rugby. Ken Schwaber adapted the ideas from Takeuchi's article to software development (Schwaber & Gladwell, SCRUM Development Process, 1996). Scrum is the most popular of the agile techniques and is rapidly growing. Of agile methods practiced in industry, 32% of them use scrum (Forrester Research, 2009).

Advocates of scrum development claim that it delivers:

- Lower overall cost
- Shorter time to market
- Adaptability to user changes
- Higher quality, operational products
- High user satisfaction

Scrum methodology has a specific set of roles, techniques and tools that help teams accurately cost and manage their software development projects.

SCRUM METHODOLOGY

Overview of the Scrum Process

Scrum projects are typified by small development teams working on incremental functional deliveries. Each functional delivery is a working software product, with the highest priority functionality developed first. After each delivery, the user gives feedback to the development team. The team modifies the requirements and continues another iteration, until the software is complete.

Schwaber and Gladwell (1996) defined three phases of scrum: