SCRUM Development Process

Ken Schwaber

Advanced Development Methods
131 Middlesex Turnpike Burlington, MA 01803
email virman@aol.com Fax: (617) 272-0555

***ABSTRACT**. The stated, accepted philosophy for systems development is that the development process is a well understood approach that can be planned, estimated, and successfully completed. This has proven incorrect in practice. SCRUM assumes that the systems development process is an unpredictable, complicated process that can only be roughly described as an overall progression. SCRUM defines the systems development process as a loose set of activities that combines known, workable tools and techniques with the best that a development team can devise to build systems. Since these activities are loose, controls to manage the process and inherent risk are used. SCRUM is an enhancement of the commonly used iterative/incremental object-oriented development cycle.*

***KEY WORDS:** SCRUM SEI Capability-Maturity-Model Process Empirical*

---

1. Introduction

In this paper we introduce a development process, SCRUM, that treats major portions of systems development as a controlled black box. We relate this to complexity theory to show why this approach increases flexibility and produces a system that is responsive to both initial and additional requirements discovered during the ongoing development.

Numerous approaches to improving the systems development process have been tried. Each has been touted as providing "significant productivity improvements." All have failed to produce dramatic improvements.[1] As Grady Booch noted, "We often call this condition the software crisis, but frankly, a malady that has carried on this long must be called normal."[2]

Concepts from industrial process control are applied to the field of systems development in this paper. Industrial process control defines processes as either "theoretical" (fully defined) or "empirical" (black box). When a black box process is treated as a fully

[1] Brooks, F.P. "No silver bullet—essence and accidents of software engineering." Computer 20:4:10-19, April 1987.
[2] Object Oriented Analysis and Design with Applications, p. 8, Grady Booch, The Benjamin/Cummings Publishing Company, Inc., 1994

defined process, unpredictable results occur. A further treatment of this is provided in Appendix 1.

A significant number of systems development processes are not completely defined, but are treated as though they are. Unpredictability without control results. The SCRUM approach treats these systems development processes as a controlled black box.

Variants of the SCRUM approach for new product development with high performance small teams was first observed by Takeuchi and Nonaka[3] at Fuji-Xerox, Canon, Honda, NEC, Epson, Brother, 3M, Xerox, and Hewlett-Packard. A similar approach applied to software development at Borland was observed by Coplien[4] to be the highest productivity C++ development project ever documented. More recently, a refined approach to the SCRUM process has been applied by Sutherland[5] to Smalltalk development and Schwaber[6] to Delphi development.

The SCRUM approach is used at leading edge software companies with significant success. Industry analysts believe SCRUM may be appropriate for other software development organizations to realize the expected benefits from Object Oriented techniques and tools.[7]

2. Overview

Our new approach to systems development is based on both defined and black box process management. We call the approach the SCRUM methodology (see Takeuchi and Nonaka, 1986), after the SCRUM in rugby -- a tight formation of forwards who bind together in specific positions when a scrumdown is called.[8]

As will be discussed later, SCRUM is an enhancement of the iterative and incremental approach to delivering object-oriented software initially documented by Pittman[9] and later expanded upon by Booch.[10] It may use the same roles for project staff as outlined by Graham[11], for example, but it organizes and manages the team process in a new way.

---

[3] Takeuchi, Hirotaka and Nonaka, Ikujiro. January-February 1986. "The New New Product Development Game." Harvard Business Review.

[4] Coplien, J. "Borland Software Craftsmanship: A New Look at Process, Quality and Productivity." Proceedings of the 5th Annual Borland International Conference, June 5, 1994. Orlando, Florida.

[5] Sutherland, Jeff. ScrumWeb Home Page: A Guide to the SCRUM Development Process, Jeff Sutherland's Object Technology Web Page, 1996 <http://www.tiac.net/users/jsuth/scrum/index.html>

[6] Schwaber, Ken. "Controlled Chaos: Living on the Edge." American Programmer, April 1996.

[7] Aberdeen Group. Upgrading To ISV Methodology For Enterprise Application Development, Product Viewpoint 8:17, December 7, 1995.

[8] Gartner, Lisa. The Rookie Primer. Radcliffe Rugby Football Club, 1996 <http://vail.al.arizona.edu/rugby/rad/rookie_primer.html>

[9] Pittman, Matthew. Lessons Learned in Managing Object-Oriented Development, IEEE Software, January, 1993, pp. 43-53.

[10] Booch, Grady. Object Solutions: Managing the Object-Oriented Project, Addison-Wesley, 1995.

[11] Graham, Ian. Migrating to Object Technology. Addison-Wesley, 1994.

SCRUM is a management, enhancement and maintenance methodology for an existing system or production prototype. It assumes existing design and code which is virtually always the case in object-oriented development due to the presence of class libraries. SCRUM will address totally new or re-engineered legacy systems development efforts at a later date.

Software product releases are planned based on the following variables :

- Customer requirements - how the current system needs enhancing.
- Time pressure - what time frame is required to gain a competitive advantage.
- Competition - what is the competition up to, and what is required to best them.
- Quality - What is the required quality, given the above variables.
- Vision - what changes are required at this stage to fulfill the system vision.
- Resource - what staff and funding are available.

These variables form the initial plan for a software enhancement project. However, these variables also change during the project. A successful development methodology must take these variables and their evolutionary nature into account.

3. Current Development Situation

Systems are developed in a highly complicated environment. The complexity is both within the development environment and the target environment. For example, when the air traffic control system development was initiated, three-tier client server systems and airline deregulation did not have to be considered. Yet, these environmental and technical changes occurred during the project and had to be taken into account within the system being built.

Environmental variables include:

- Availability of skilled professionals - the newer the technology, tools, methods, and domain, the smaller the pool of skilled professionals.

- Stability of implementation technology - the newer the technology, the lower the stability and the greater the need to balance the technology with other technologies and manual procedures.

- Stability and power of tools - the newer and more powerful the development tool, the smaller the pool of skilled professionals and the more unstable the tool functionality.

- Effectiveness of methods - what modeling, testing, version control, and design methods are going to be used, and how effective, efficient, and proven are they.

- Domain expertise - are skilled professionals available in the various domains, including business and technology.

- New features - what entirely new features are going to be added, and to what degree will these fit with current functionality.

- Methodology - does the overall approach to developing systems and using the selected methods promote flexibility, or is this a rigid, detailed approach that restricts flexibility.

- Competition - what will the competition do during the project? What new functionality will be announced or released.

- Time/Funding - how much time is available initially and as the project progresses? How much development funding is available.

- Other variables - any other factors that must be responded to during the project to ensure the success of the resulting, delivered system, such as reorganizations.

The overall complexity is a function of these variables :

complexity = f(development environment variables + target environment variables)

where these variables may and do change during the course of the project.

As the complexity of the project increases, the greater the need for controls, particularly the ongoing assessment and response to risk.

Attempts to model this development process have encountered the following problems:

- Many of the development processes are uncontrolled. The inputs and outputs are either unknown or loosely defined, the transformation process lacks necessary precision, and quality control is not defined. Testing processes are an example.

- An unknown number of development processes that bridge known but uncontrolled processes are unidentified. Detailed processes to ensure that a logical model contains adequate content to lead to a successful physical model is one such process.

- Environmental input (requirements) can only be taken into consideration at the beginning of the process. Complex change management procedures are required thereafter.

Attempts to impose a micro, or detailed, methodology model on the development process have not worked because the development process is still not completely defined. Acting

as though the development process is defined and predictable results in being unprepared for the unpredictable results.

Although the development process is incompletely defined and dynamic, numerous organizations have developed detailed development methodologies that include current development methods (structured, OO, etc.). The Waterfall methodology was one of the first such defined system development processes. A picture of the Waterfall methodology is shown in Figure 1.

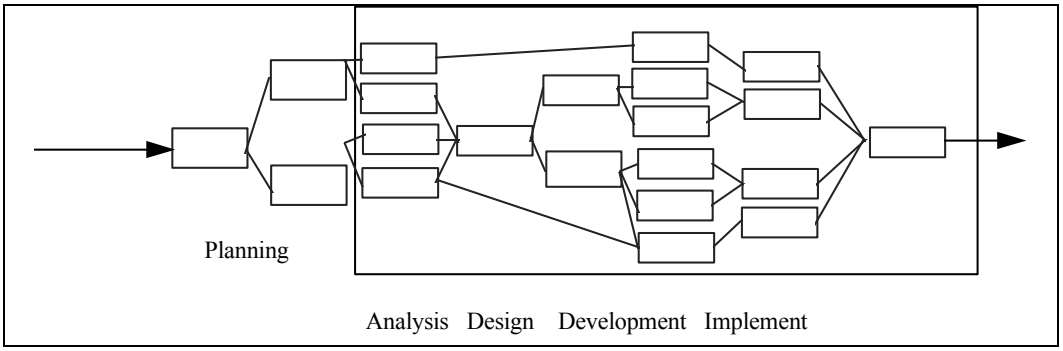

Figure 1 : Waterfall Methodology

Although the waterfall approach mandates the use of undefined processes, its linear nature has been its largest problem. The process does not define how to respond to unexpected output from any of the intermediate process.

Barry Boehm[12] introduced a Spiral methodology to address this issue. Each of the waterfall phases is ended with a risk assessment and prototyping activity. The Spiral methodology is shown in Figure 2.

The Spiral methodology "peels the onion", progressing through "layers" of the development process. A prototype lets users determine if the project is on track, should be sent back to prior phases, or should be ended. However, the phases and phase processes are still linear. Requirements work is still performed in the requirements phase, design work in the design phase, and so forth, with each of the phases consisting of linear, explicitly defined processes.

[12] Boehm, B.W. 1985. "A Spiral Model of Software Development and Enhancement," from Proceedings of an International Workshop on Software Process and Software Environments, Coto de Caza, Trabuco Canyon, California, March 27-29, 1985.

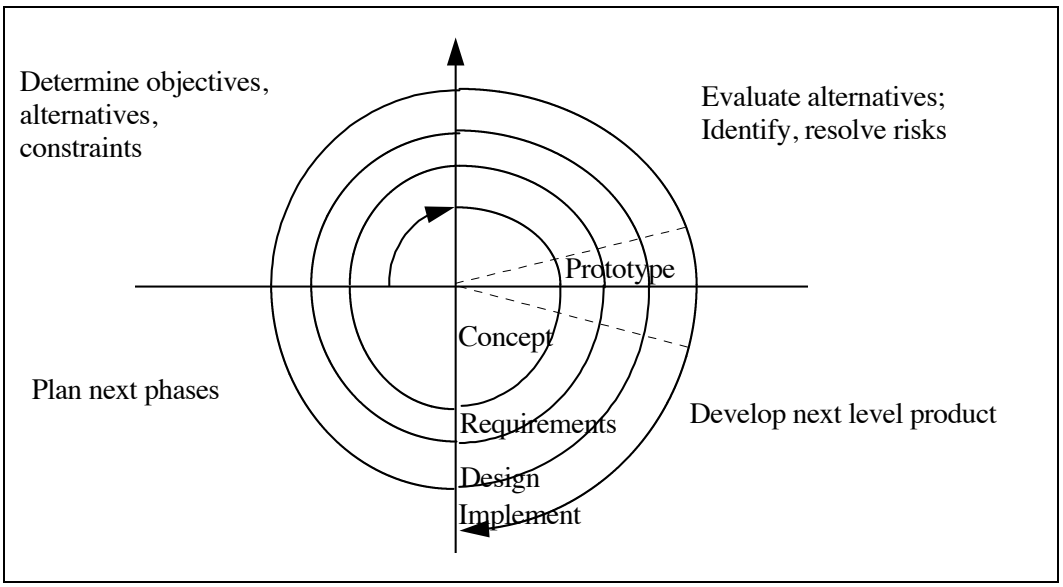

Figure 2 : Spiral Methodology

The Iterative methodology improves on the Spiral methodology. Each iteration consists of all of the standard Waterfall phases, but each iteration only addresses one set of parsed functionality. The overall project deliverable has been partitioned into prioritized subsystems, each with clean interfaces. Using this approach, one can test the feasibility of a subsystem and technology in the initial iterations. Further iterations can add resources to the project while ramping up the speed of delivery. This approach improves cost control, ensures delivery of systems (albeit subsystems), and improves overall flexibility. However, the Iterative approach still expects that the underlying development processes are defined and linear. See Figure 3.