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.

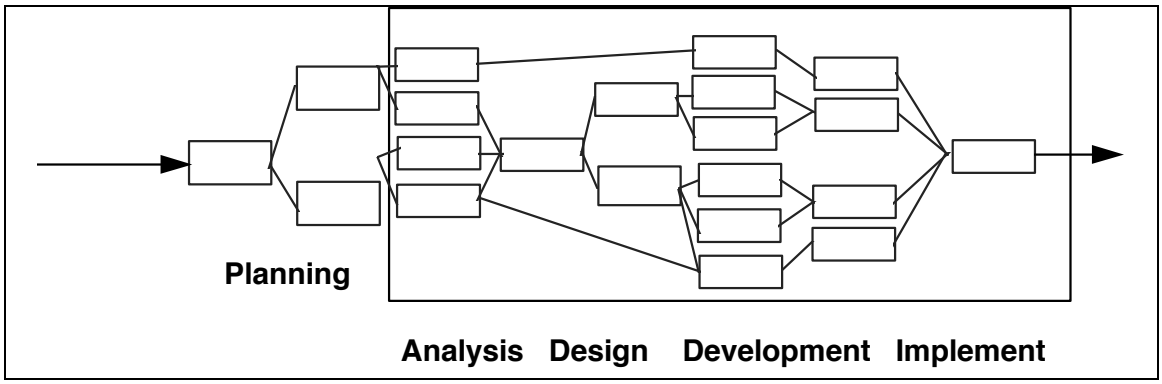

Figure 1 : Waterfall Methodology

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.

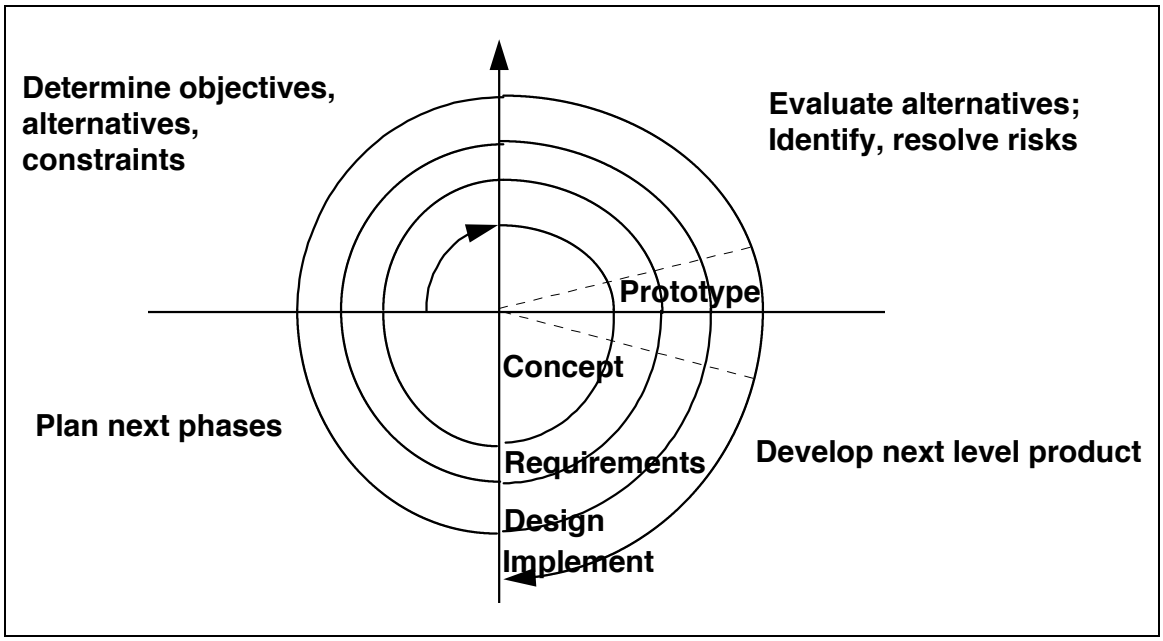

Figure 2 : Spiral Methodology

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.

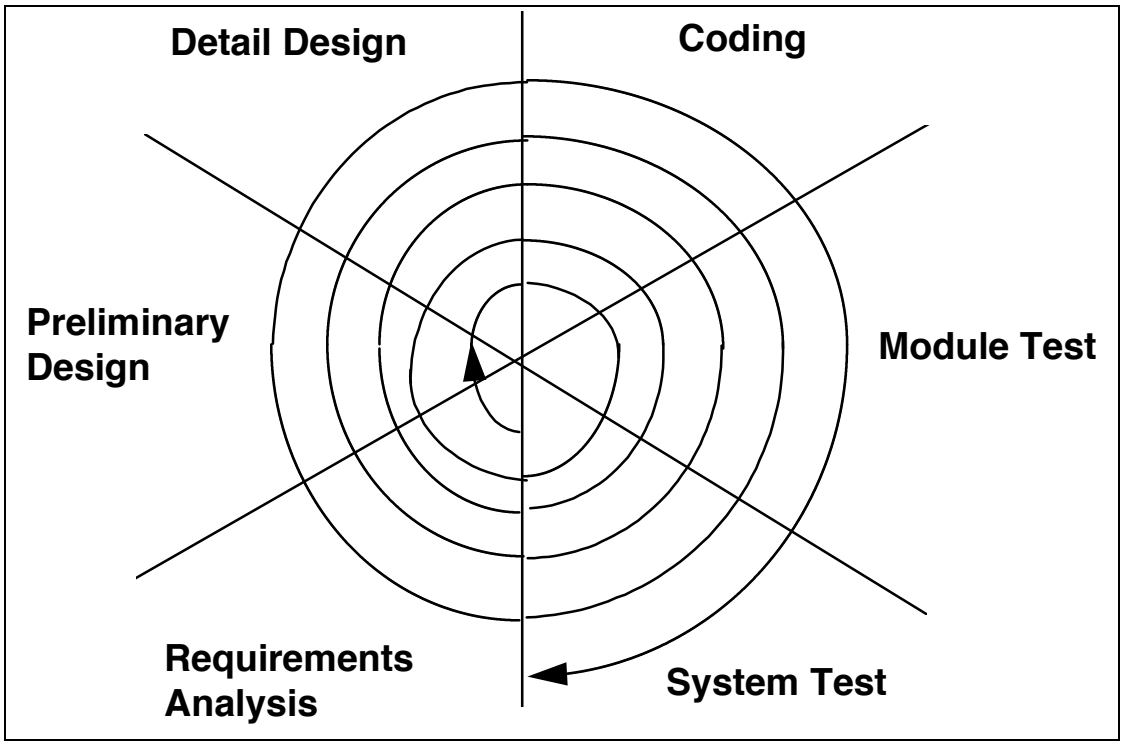

Figure 3 : Iterative Methodology

Given the complex environment and the increased reliance on new "state-of-the-art" systems, the risk endured by system development projects has increased and the search for mechanisms to handle this risk has intensified.

One can argue that current methodologies are better than nothing. Each improves on the other. The Spiral and Iterative approaches implant formal risk control mechanisms for dealing with unpredictable results. A framework for development is provided.

However, each rests on the fallacy that the development processes are defined, predictable processes. But unpredictable results occur throughout the projects. The rigor implied in the development processes stifles the flexibility needed to cope with the unpredictable results and respond to a complex environment.

Despite their widespread presence in the development community, our experience in the industry shows that people do not use the methodologies except as a macro process map, or for their detailed method descriptions.

The following graph demonstrates the current development environment, using any of the Waterfall, Spiral or Iterative processes. As the complexity of the variables increase even to a moderate level, the probability of a “successful” project quickly diminishes (a successful project is defined as a system that is useful when delivered). See Figure 4.

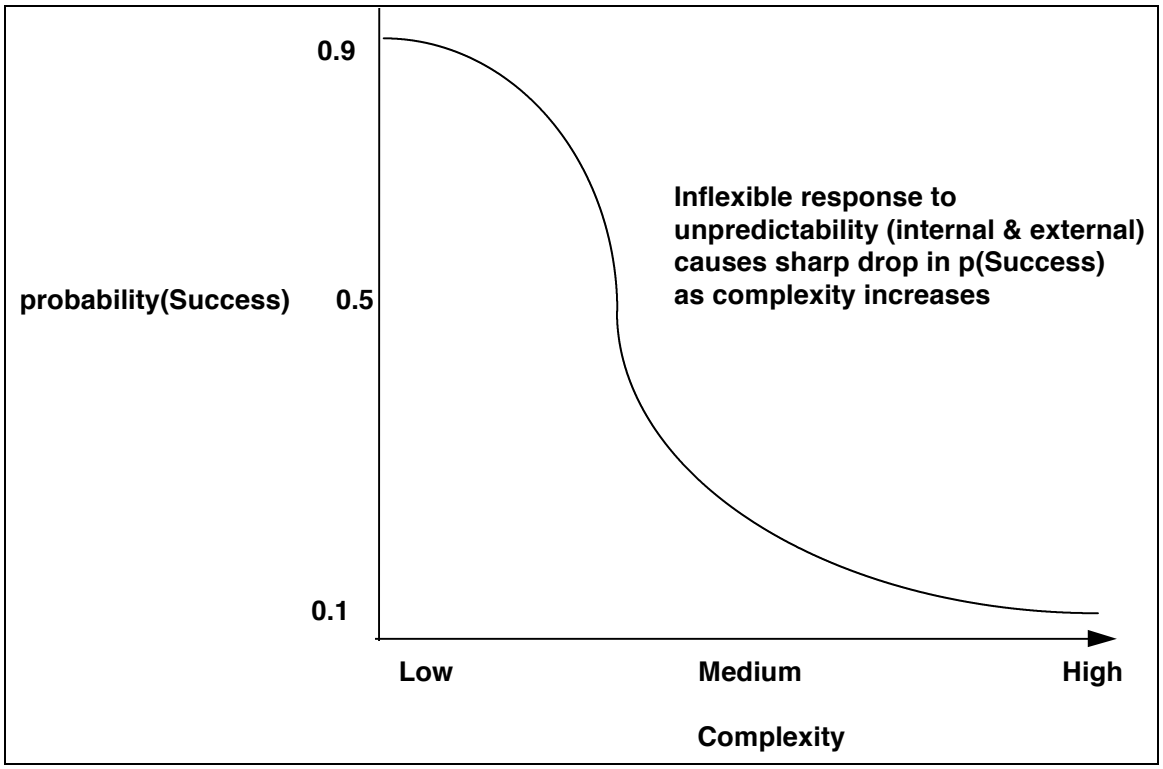

Figure 4
Defined Process Risk/Complexity Graph

4. SCRUM Methodology

The system development process is complicated and complex. Therefore maximum flexibility and appropriate control is required. Evolution favors those that operate with maximum exposure to environmental change and have optimised for flexible adaptation to change. Evolution deselects those who have insulated themselves from environmental change and have minimized chaos and complexity in their environment.

An approach is needed that enables development teams to operate adaptively within a complex environment using imprecise processes. Complex system development occurs under rapidly changing circumstances. Producing orderly systems under chaotic circumstances requires maximum flexibility. The closer the development team operates to the edge of chaos, while still maintaining order, the more competitive and useful the resulting system will be. Langton has modeled this effect in computer simulations[13] and his work has provided this as a fundamental theorem in complexity theory.

[13] Langton, Christopher. Artificial Life. InArtificial Life, Volume VI: SFI Studies in the Sciences of Complexity (Ed. C. Langton) Addison-Wesley, 1988.

Methodology may well be the most important factor in determining the probability of success. Methodologies that encourage and support flexibility have a high degree of tolerance for changes in other variables. With these methodologies, the development process is regarded as unpredictable at the onset, and control mechanisms are put in place to manage the unpredictability.

If we graph the relationship between environmental complexity and probability of success with a flexible methodology that incorporates controls and risk management, the tolerance for change is more durable. See Figure 5.

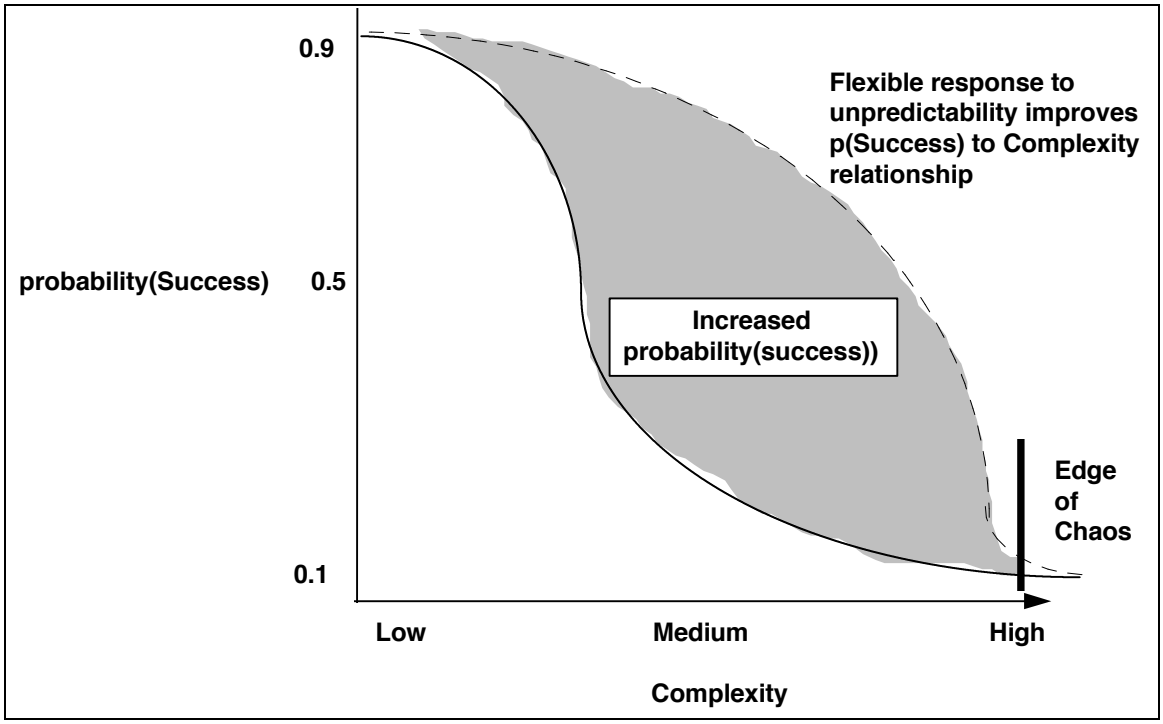

Figure 5 - Risk/Complexity Comparison Graph

Figures 4 and 5 reflect software development experiences at ADM, Easel, VMARK, Borland and virtually every other developer of "packaged" software. These organizations have embraced risk and environmental complexity during development projects. Increased product impact, successful projects, and productivity gains were experienced. The best possible software is built.

Waterfall and Spiral methodologies set the context and deliverable definition at the start of a project. SCRUM and Iterative methodologies initially plan the context and broad deliverable definition, and then evolve the deliverable during the project based on the environment. SCRUM acknowledges that the underlying development processes are incompletely defined and uses control mechanisms to improve flexibility.

The primary difference between the defined (waterfall, spiral and iterative) and empirical (SCRUM) approach is that The SCRUM approach assumes that the analysis, design, and development processes in the Sprint phase are unpredictable. A control mechanism is used to manage the unpredictability and control the risk. Flexibility, responsiveness, and reliability are the results. See Figure 6.

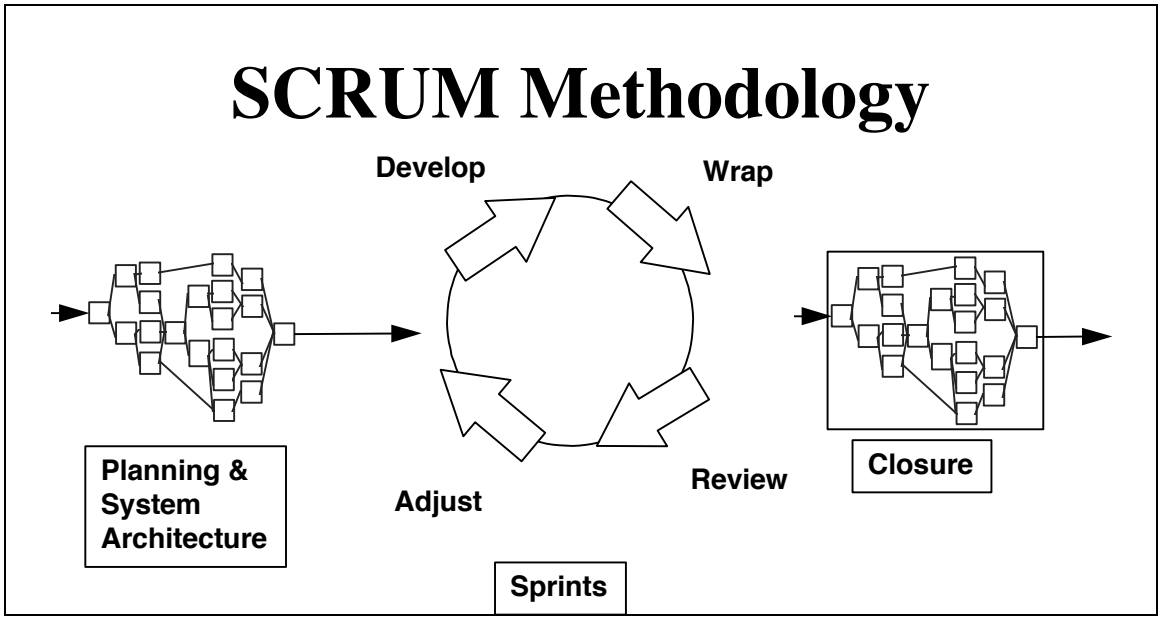

Figure 6 : SCRUM Methodology

Characteristics of SCRUM methodology are :

- The first and last phases (Planning and Closure) consist of defined processes, where all processes, inputs and outputs are well defined. The knowledge of how to do these processes is explicit. The flow is linear, with some iterations in the planning phase.

- The Sprint phase is an empirical process. Many of the processes in the sprint phase are unidentified or uncontrolled. It is treated as a black box that requires external controls. Accordingly, controls, including risk management, are put on each iteration of the Sprint phase to avoid chaos while maximizing flexibility.

- Sprints are nonlinear and flexible. Where available, explicit process knowledge is used; otherwise tacit knowledge and trial and error is used to build process knowledge. Sprints are used to evolve the final product.

- The project is open to the environment until the Closure phase. The deliverable can be changed at any time during the Planning and Sprint phases of the project. The project remains open to environmental complexity, including competitive, time, quality, and financial pressures, throughout these phases.

- The deliverable is determined during the project based on the environment.

Figure 7 compares the primary SCRUM characteristics to those of other methodologies.

	Waterfall	**Spiral**	**Iterative**	**SCRUM**
Defined processes	Required	Required	Required	Planning & Closure only
Final product	Determined during planning	Determined during planning	Set during project	Set during project
Project cost	Determined during planning	Partially variable	Set during project	Set during project
Completion date	Determined during planning	Partially variable	Set during project	Set during project
Responsiveness to environment	Planning only	Planning primarily	At end of each iteration	**Throughout**
Team flexibility, creativity	Limited - cookbook approach	Limited - cookbook approach	Limited - cookbook approach	**Unlimited during iterations**
Knowledge transfer	Training prior to project	Training prior to project	Training prior to project	**Teamwork during project**
Probability of success	Low	Medium low	Medium	**High**

Figure 7 : Methodology Comparison

4.1 SCRUM Phases

SCRUM has the following groups of phases:

4.1.1. Pregame

- Planning : Definition of a new release based on currently known backlog, along with an estimate of its schedule and cost. If a new system is being developed, this phase consists of both conceptualization and analysis. If an existing system is being enhanced, this phase consists of limited analysis.
- Architecture : Design how the backlog items will be implemented. This phase includes system architecture modification and high level design.

4.1.2. Game

- Development Sprints : Development of new release functionality, with constant respect to the variables of time, requirements, quality, cost, and competition. Interaction with these variables defines the end of this phase. There are multiple, iterative development sprints, or cycles, that are used to evolve the system.

4.1.3. Postgame

Closure : Preparation for release, including final documentation, pre-release staged testing, and release.

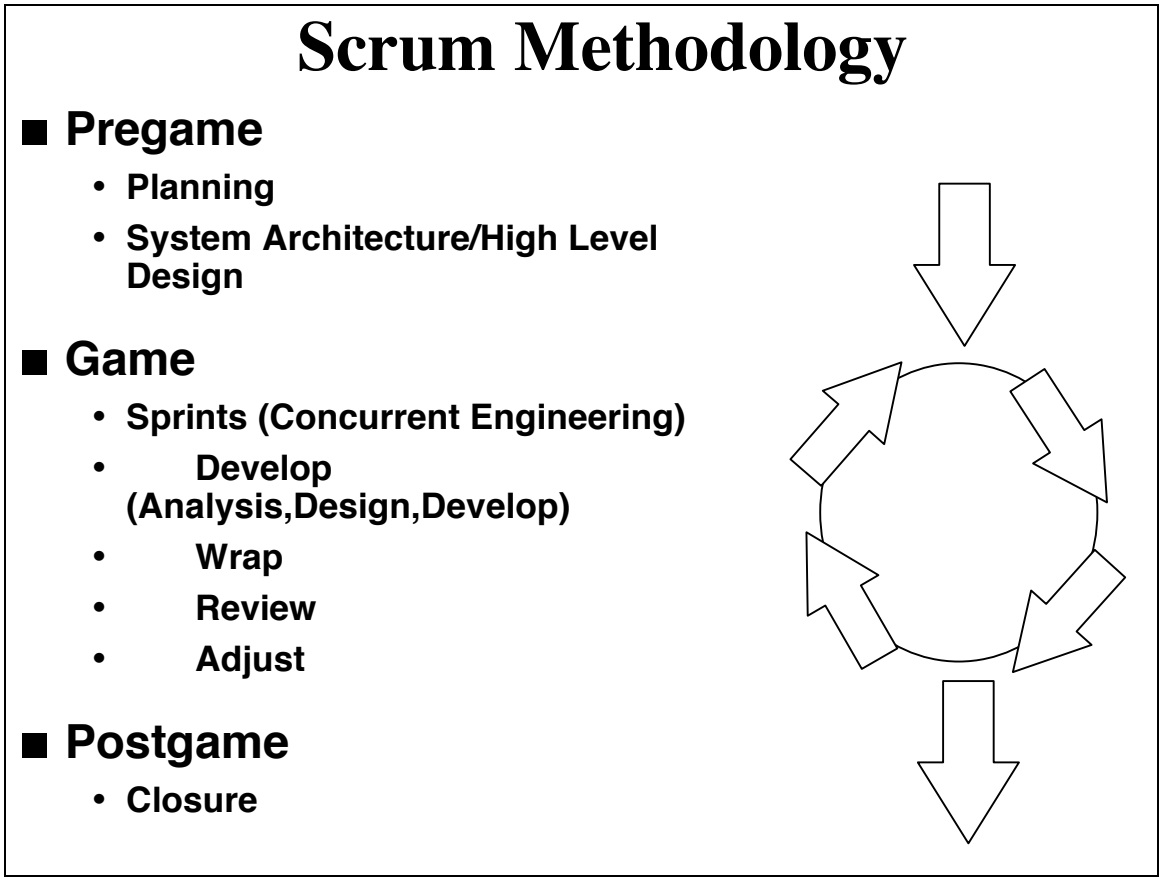

Figure 9

4.2 Phase Steps

Each of the phases has the following steps:

4.2.1. Planning

- Development of a comprehensive backlog list.
- Definition of the delivery date and functionality of one or more releases.
- Selection of the release most appropriate for immediate development.
- Mapping of product packets (objects) for backlog items in the selected release.
- Definition of project team(s) for the building of the new release.
- Assessment of risk and appropriate risk controls.
- Review and possible adjustment of backlog items and packets.
- Validation or reselection of development tools and infrastructure.
- Estimation of release cost, including development, collateral material, marketing, training, and rollout.
- Verification of management approval and funding.

4.2.2. Architecture/High Level Design

- Review assigned backlog items.
- Identify changes necessary to implement backlog items.
- Perform domain analysis to the extent required to build, enhance, or update the domain models to reflect the new system context and requirements.
- Refine the system architecture to support the new context and requirements.
- Identify any problems or issues in developing or implementing the changes
- Design review meeting, each team presenting approach and changes to implement each backlog item. Reassign changes as required.

4.2.3. Development (Sprint)

The Development phase is an iterative cycle of development work. The management determines that time, competition, quality, or functionality are met, iterations are completed and the closure phase occurs. This approach is also known as Concurrent Engineering. Development consists of the following macro processes :

- Meeting with teams to review release plans.
- Distribution, review and adjustment of the standards with which the product will conform.
- Iterative Sprints, until the product is deemed ready for distribution.

A Sprint is a set of development activities conducted over a pre-defined period, usually one to four weeks. The interval is based on product complexity, risk assessment, and degree of oversight desired. Sprint speed and intensity are driven by the selected duration of the Sprint. Risk is assessed continuously and adequate risk controls and responses put in place. Each Sprint consists of one or more teams performing the following:

- Develop: Defining changes needed for the implementation of backlog requirements into packets, opening the packets, performing domain analysis, designing, developing, implementing, testing, and documenting the changes. Development consists of the micro process of discovery, invention, and implementation.
- Wrap: Closing the packets, creating a executable version of changes and how they implement backlog requirements.
- Review: All teams meeting to present work and review progress, raising and resolving issues and problems, adding new backlog items. Risk is reviewed and appropriate responses defined.
- Adjust: Consolidating the information gathered from the review meeting into affected packets, including different look and feel and new properties.

Each Sprint is followed by a review, whose characteristics are :

- The whole team and product management are present and participate.
- The review can include customers, sales, marketing and others.
- Review covers functional, executable systems that encompass the objects assigned to that team and include the changes made to implement the backlog items.
- The way backlog items are implemented by changes may be changed based on the review.
- New backlog items may be introduced and assigned to teams as part of the review, changing the content and direction of deliverables.
- The time of the next review is determined based on progress and complexity. The Sprints usually have a duration of 1 to 4 weeks.

4.2.4. Closure

When the management team feels that the variables of time, competition, requirements, cost, and quality concur for a new release to occur, they declare the release "closed" and enter this phase. This phase prepares the developed product for general release. Integration, system test, user documentation, training material preparation, and marketing material preparation are among closure tasks.

4.3. SCRUM Controls

Operating at the edge of chaos (unpredictability and complexity) requires management controls to avoid falling into chaos. The SCRUM methodology embodies these general, loose controls, using OO techniques for the actual construction of deliverables.

Risk is the primary control. Risk assessment leads to changes in other controls and responses by the team.

Controls in the SCRUM methodology are :

- Backlog: Product functionality requirements that are not adequately addressed by the current product release. Bugs, defects, customer requested enhancements, competitive product functionality, competitive edge functionality, and technology upgrades are backlog items.

- Release/Enhancement: backlog items that at a point in time represent a viable release based on the variables of requirements, time, quality, and competition.

- Packets: Product components or objects that must be changed to implement a backlog item into a new release.

- Changes: Changes that must occur to a packet to implement a backlog item.

- Problems: Technical problems that occur and must be solved to implement a change.

- Risks: risks that effect the success of the project are continuously assessed and responses planned. Other controls are affected as a result of risk assessment.

- Solutions: solutions to the problems and risks, often resulting in changes.

- Issues: Overall project and project issues that are not defined in terms of packets, changes and problems.

These controls are used in the various phases of SCRUM. Management uses these controls to manage backlog. Teams use these controls to manage changes, problems. Both management and teams jointly manage issues, risks, and solutions. These controls are reviewed, modified, and reconciled at every Sprint review meeting.

4.4 SCRUM Deliverables

The delivered product is flexible. Its content is determined by environment variables, including time, competition, cost, or functionality. The deliverable determinants are market intelligence, customer contact, and the skill of developers. Frequent adjustments to deliverable content occur during the project in response to environment. The deliverable can be determined anytime during the project.

4.5 SCRUM Project Team

The team that works on the new release includes full time developers and external parties who will be affected by the new release, such as marketing, sales, and customers. In traditional release processes, these latter groups are kept away from development teams for fear of over-complicating the process and providing "unnecessary" interference. The SCRUM approach, however, welcomes and facilitates their controlled involvement at set intervals, as this increases the probability that release content and timing will be appropriate, useful, and marketable.

The following teams are formed for each new release:

Management: Led by the Product Manager, it defines initial content and timing of the release, then manages their evolution as the project progresses and variables change. Management deals with backlog, risk, and release content.

Development teams: Development teams are small, with each containing developers, documenters and quality control staff. One or more teams of between three and six people each are used. Each is assigned a set of packets (or objects), including all backlog items related to each packet. The team defines changes required to implement the backlog item in the packets, and manages all problems regarding the changes. Teams can be either functionally derived (assigned those packets that address specific sets of product functionality) or system derived (assigned unique layers of the system). The members of each team are selected based on their knowledge and expertise regarding sets of packets, or domain expertise.

4.6 SCRUM Characteristics

The SCRUM methodology is a metaphor for the game of Rugby. Rugby evolved from English football (soccer) under the intense pressure of the game :

> Rugby student William Webb Ellis, 17, inaugurates a new game whose rules will be codified in 1839. Playing soccer for the 256-year-old college in East Warwickshire, *Ellis sees that the clock is running out with his team behind so he scoops up the ball and runs with it in defiance of the rules*.
> **The People's Chronology**, Henry Holt and Company, Inc. Copyright © 1992.

SCRUM projects have the following characteristics :

- Flexible deliverable - the content of the deliverable is dictated by the environment.

- Flexible schedule - the deliverable may be required sooner or later than initially planned.

- Small teams - each team has no more than 6 members. There may be multiple teams within a project.

- Frequent reviews - team progress is reviewed as frequently as environmental complexity and risk dictates (usually 1 to 4 week cycles). A functional executable must be prepared by each team for each review.

- Collaboration - intra and inter-collaboration is expected during the project.

- Object Oriented - each team will address a set of related objects, with clear interfaces and behavior.

The SCRUM methodology shares many characteristics with the sport of Rugby :

- The context is set by playing field (environment) and rugby rules (controls).

- The primary cycle is moving the ball forward.

- Rugby evolved from breaking soccer rules - adapting to the environment.

- The game does not end until environment dictates (business need, competition, functionality, timetable).

5. Advantages of the SCRUM Methodology

Traditional development methodologies are designed only to respond to the unpredictability of the external and development environments at the start of an enhancement cycle. Such newer approaches as the Boehm spiral methodology and its variants are still limited in their ability to respond to changing requirements once the project has started.

The SCRUM methodology, on the other hand, is designed to be quite flexible throughout. It provides control mechanisms for planning a product release and then managing variables as the project progresses. This enables organizations to change the project and deliverables at any point in time, delivering the most appropriate release.

The SCRUM methodology frees developers to devise the most ingenious solutions throughout the project, as learning occurs and the environment changes.

Small, collaborative teams of developers are able to share tacit knowledge about development processes. An excellent training environment for all parties is provided.

Object Oriented technology provides the basis for the SCRUM methodology. Objects, or product features, offer a discrete and manageable environment. Procedural code, with its many and intertwined interfaces, is inappropriate for the SCRUM methodology. SCRUM may be selectively applied to procedural systems with clean interfaces and strong data orientation.

6. SCRUM Project Estimating

SCRUM projects can be estimated using standard function point estimating. However, it is advisable to estimate productivity at approximately twice the current metric. The estimate is only for starting purposes, however, since the overall timetable and cost are determined dynamically in response to the environmental factors.

Our observations have led us to conclude that SCRUM projects have both velocity and acceleration. In terms of functions delivered, or backlog items completed :

- initial velocity and acceleration are low as infrastructure is built/modified
- as base functionality is put into objects, acceleration increases
- acceleration decreases and velocity remains sustainably high

Further development in metrics for empirical processes is required**.**

7. System Development Methodologies : Defined or Empirical

System development is the act of creating a logical construct that is implemented as logic and data on computers. The logical construct consists of inputs, processes, and outputs, both macro (whole construct) and micro (intermediate steps within whole construct). The whole is known as an implemented system.

Many artifacts are created while building the system. Artifacts may be used to guide thinking, check completeness, and create an audit trail. The artifacts consist of documents, models, programs, test cases, and other deliverables created prior to creating the implemented system. When available, a *metamodel* defines the semantic content of model artifacts. *Notation* describes the graphing and documentation conventions that are used to build the models.

The approach used to develop a system is known as a *method*. A *method* describes the activities involved in defining, building, and implementing a system; a *method* is a framework. Since a *method* is a logical process for constructing systems (process), it is known as a *metaprocess* (a process for modeling processes).

A *method* has micro and macro components. The macro components define the overall flow and time-sequenced framework for performing work. The micro components include *general design rules*, *patterns* and *rules of thumb*. *General design rules* state properties to achieve or to avoid in the design or general approaches to take while building a system. *Patterns* are solutions that can be applied to a type of development activity; they are solutions waiting for problems that occur during an activity in a method. *Rules of thumb* consist of a general body of hints and tips.

Figure 1 visualizes the relationship between the Method, the Artifacts, and the System.

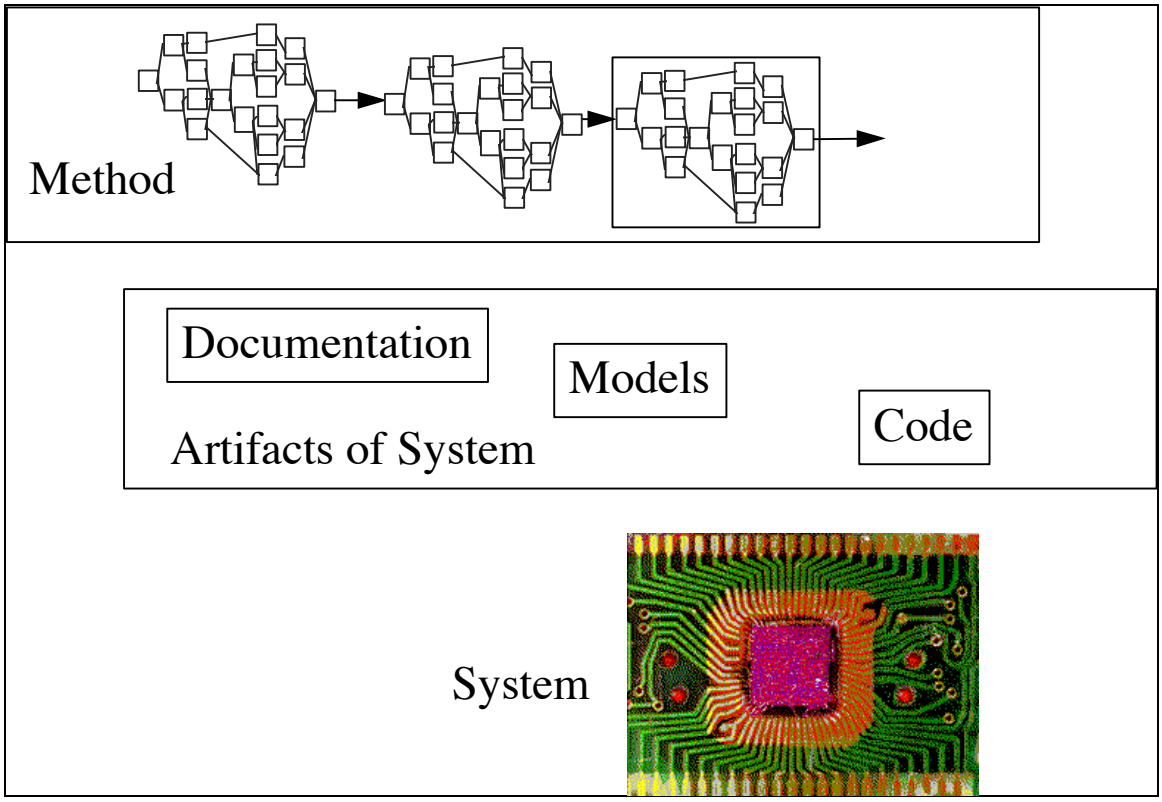

Figure 1

Applying concepts from industrial process control to the field of systems development, *methods* can be categorized as either "theoretical" (fully defined) or "empirical" (black box).

Correctly categorizing systems development methods is critical. The appropriate structure of a *method* for building a particular type of system depends on whether the method is theoretical or empirical.

Models of *theoretical processes* are derived from *first principles*, using material and energy balances and fundamental laws to determine the model. For a systems development *method* to be categorized as theoretical, it must conform to this definition.

Models of ***empirical processes*** are derived categorizing observed inputs and outputs, and defining controls that cause them to occur within prescribed bounds. Empirical process modeling involves constructing a process model strictly from experimentally obtained input/output data, with no recourse to any laws concerning the fundamental nature and properties of the system. No *a priori* knowledge about the process is necessary (although it can be helpful); a system is treated like a black box.

Primary characteristics of both theoretical and empirical modeling are detailed in Table 1.

Theoretical Modeling	Empirical Modeling
1. Usually involves fewer measurements; requires experimentation only for the estimation of unknown model parameters.	Requires extensive measurements, because it relies entirely on experimentation for the model development.
2. Provides information about the internal state of the process.	Provides information only about that portion of the process which can be influenced by control action.
3. Promotes fundamental understanding of the internal workings of the process.	Treats the process like a "black box."
4. Requires fairly accurate and complete process knowledge.	Requires no such detailed knowledge; only that output data be obtainable in response to input changes.
5. Not particularly useful for poorly understood and/or complex processes.	Quite often proves to be the only alternative for modeling the behavior of poorly understood and/or complex processes.
6. Naturally produces both linear and nonlinear process models.	Requires special methods to produce nonlinear models.

Table 1

Upon inspection, we assert that the systems development process is empirical:

1. Applicable first principles are not present
2. The process is only beginning to be understood
3. The process is complex
4. The process is changing

Most methodologists agree with this assertion; "...you can't expect a *method* to tell you everything to do. Writing software is a creative process, like painting or writing or architecture... ... (a *method*) supplies a framework that tells how to go about it and identifies the places where creativity is needed. But you still have to supply the creativity...."[14]

Categorizing the systems development methods as empirical is critical to the effective management of the systems development process.

If systems development *methods* are categorized as *empirical*, measurements and controls are required because it is understood that the inner workings of the *method* are so loosely defined that they cannot be counted on to operate predictably.

In the past, *methods* have been provided and applied as though they were theoretical. As a consequence, measurements were not relied upon and controls dependent upon the measurements weren't used.

Many of the problems in developing systems have occurred because of this incorrect categorization. When a black box process is treated as a fully defined process, unpredictable results occur. Also, the controls are not in place to measure and respond to the unpredictability.

In practice, at multiple companies over multiple projects within some companies, SCRUM has been observed to provide a viable solution these problems. Projects are delivered on time and often exceed the expectations of both users and management. While working on a SCRUM development team is intense, developers are rewarded by high team spirit, a deep sense of accomplishment, and a feeling that development can be an enjoyable and satisfying experience.

References

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

2. Bach, James. "Process Evolution in a Mad World." Borland International, Scotts Valley, CA.

3. Bach, James. "The Challenge of "Good Enough" Software", American Programmer, October 1995.

4. 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.

[14] James Rumbaugh, October 1995, "What Is a Method"., Object Journal

5. Booch, Grady. Object Oriented Analysis and Design with Applications. The Benjamin/Cummings Publishing Company, Inc., 1994, p. 8

6. Booch, Grady. Object Solutions: Managing the Object-Oriented Project. Addison-Wesley, 1995.

7. Brooks, F.P. "No silver bullet—essence and accidents of software engineering." Computer 20:4:10-19, April 1987.

8. 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.

9. DeGrace, P. and Hulet Stahl, L. 1990. *Wicked Problems, Righteous Solutions*. Yourdon Press

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

11. Gleick, J. 1987. *Chaos, Making A New Science*. Penguin Books.

12. Graham, Ian. Migrating to Object Technology. Addison-Wesley, 1994.

13. Kahn, D. and Sutherland, J. March-April 1994. "Object Insider: Let's start under-promising and over-delivering on OT." Object Magazine.

14. Langton, Christopher. Artificial Life. In Artificial Life, Volume VI: SFI Studies in the Sciences of Complexity (Ed. C. Langton) Addison-Wesley, 1988.

15. Nonaka, Ikujiro and Takeuchi, Hirotaka. 1995. *The Knowledge Creating Company: How Japanese Companies Create the Dynamics of Innovation*, Oxford University Press.

16. Ogunnaike, B. 1994. *Process Dynamics, Modeling, and Control*. Oxford University Press.

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

18. Rumbaugh, October 1995, "What Is a Method". Journal of Object Oriented Programming.

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

20. 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>

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