Whitepaper

SCRUM Common Sense in Action

A collection of Articles from Ken Schwaber

Version: 2005-00-01

teamresponz

Table of Contents

Philosophies of Software Development 3
- Micro and Macro components 4

The Philosophy of Scrum 5
- The approach is called Scrum. 5
- We can't do that! 6

Begin Using Scrum 6
- Start the Scrum Process 6
- Appoint Scrum Master 7
- Identify Backlog 7
- Establish and Conduct Daily Scrum Meeting 7
- Why Scrum Is Powerful 8

Bottom-Up Continuous Processing Improvement 8

Bottom-Up Continuous Processing Improvement 9
- Scrum's Secrets: Focus and Visibility 9
- Scrum's Three Tools 9
- Examples of Re-Engineering 10
- Rapid Bottom-Up Re-Engineering as Painful but Necessary 10
- Scrum Causes Change 11

Scrum Rules 11
- Backlog 12
- Sprint 12
- Scrum Meeting 12

Vocabulary 13

Frequently Asked Questions 13

References 16
- Web 16
- Books 16

Document ID	WP-SCRUM-011	**Filename**	SCRUM Common Sense in Action
Author	Kurt B. Nielsen	**Revised**	2005-00-01

Philosophies of Software Development

The emperor is not wearing any clothes. The SEI CMM Level 3 defined approach to systems development does not and cannot work. Research between ADM, Vmark, Borland, and DuPont Advanced Research Facility has proven that systems development is an empirical process that requires controls. The controls ensure adequate risk of management and adequate response to the unknown and unexpected.

A war is raging as some of the most respected names in the industry try to apply the rigid, stifling development of the past to OO. They are confronted in the marketplace and on the Internet by the successes and commentary of those who are building the best possible software using SCRUM like approaches.

More successful organizations use a revolutionary systems development process: **Scrum**. Scrum contains the essence of development at these and other successful organizations. SCRUM (the word comes from the game of rugby) enables OO through advanced iterative, incremental development within a controlled environment. It allows organizations to build the best possible OO systems possible.

OO development requires a different development process. SCRUM is not a step-by-step cookbook approach. SCRUM requires active, thoughtful development and management. Assessing, adjusting, thinking - these are the characteristics of a successful SCRUM.

Many new development approaches have failed. IS managers are so desperate for a "silver bullet" that the salesmen joke that IS management has to turn over every three years to support the "silver bullet" software industry. CASE is a great example.

Failed approaches have become emperors without any clothes: everyone knows that they don't work, but everyone is embarrassed to say so. One failure builds on another.

The Capability Maturity Model from the Software Engineering Institute represents the current fad. Not that the concept of improving processes is wrong. It's just that the systems development process is not defined past level two. That doesn't stop gaggles of consultants from selling "self-improvement" programs to get organizations to level 3. Too bad that it doesn't work. The whole philosophical basis of level 3 defined processes is wrong.

As organizations have succeeded with empirical, controlled approaches to systems development, the chasm between them and SEI proponents has become apparent. Now the two side's yodel at each other from either side, both in the press and the Internet.

Systems 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 meta model 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 meta process (a process for modeling processes).

Micro and Macro components

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.

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.

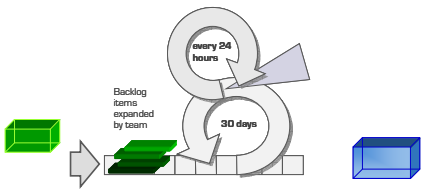
Models of theoretical processes are derived from first principles, using material and energy balances and fundamental laws to determine the model. For a systems development method to be categorized as theoretical, it must conform to this definition. Models of empirical processes are derived categorizing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

We assert that the systems development process is empirical:

- Applicable first principles are not present;
- The process is only beginning to be understood;
- The process is complex;
- The process is changing;

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.

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.

The Philosophy of Scrum

The core of the Scrum approach is the belief that most systems development has the wrong philosophical basis. The stated, accepted philosophy is that systems development process is a well understood approach that can be planned, estimated, and successfully completed.

The failed projects, inappropriate systems, and ineffective productivity tools are seen as proof that the development process needs more rigor, that if we can get those unruly developers to actually follow it, these maladies will go away.

Scrum states that the systems development process is an unpredictable, complicated process that can only be roughly described as an overall progression. Cookbook, step-by-step approaches do not work because they aren't adequately defined and don't cope with the unpredictability of systems development.

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.

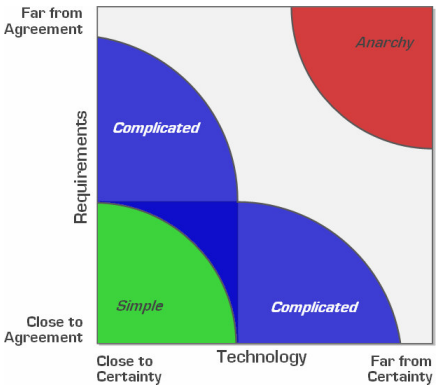
The following maladies occur as a result applying the wrong philosophy and techniques to systems development:

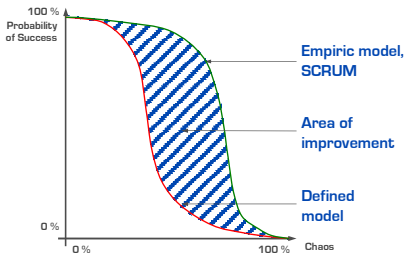
- By the time the system is delivered, it is often irrelevant or requires significant change. This occurs because environmental inputs are used only during planning.
- Management actually believes that it can predict the cost, delivery schedule, and functionality that will be delivered.
- Developers and project managers are forced to live a lie...they have to pretend that they can plan, predict and deliver, and then work the best way that they know how to deliver a system. They build one way, pretend that they build another way, and are without controls.

Scrum is applicable to internal IS Organizations. Scrum will relieve IS Organizations of these terrible symptoms.

The approach is called Scrum.

It starts with the acceptance that this is a complicated, unpredictable world and development environment.





- Whitepaper – SCRUM Common Sense in Action

It also starts with the premise that you can't predict or definitively plan what you will deliver, when you will deliver it, and what the quality and cost will be.

It starts with the assumption that you can estimate these, and then negotiate them according to various risks as you proceed.

It is understood at the start that you will deliver the best possible software given the circumstances, and that following any cookbook approach won't improve the definition of "best", and will only hinder appropriate responsiveness to the complexity and unpredictability.

We can't do that!

The reaction of many IS organizations to this approach includes "we can't do that, everything will be out of control..." and "our board of directors will never approve funding to build an undefined product." However, that is exactly the way it is right now: out of control because of the inability to respond to unpredictability, and the funding is for products whose definition changes from the first day of the project.

As the divergence between those who are succeeding with a Scrum-like development process and those who are failing with SEI-CMM type processes increases, the heat of the argument is rising.

ADM and VMark have formalized the Scrum process, based on packaged software vendor experiences, because they feel that OO will suffer and possibly fail using the current development philosophy.

Scrum is applicable to new or existing systems that use clean interface or objects and components. Scrum enables component based development. Scrum tolerates on the job learning. As a matter of fact, Scrum is a knowledge creating process, where tacit knowledge is created and shared as work progresses. Collaborative teamwork ensures knowledge sharing and creation.

Begin Using Scrum

Scrum is application of common-sense to work that uses productivity producing techniques applied to software engineering. Scrum places highest priority on doing work and producing releases.

To implement the Scrum process for a specific area of work:

- Start the Scrum Process
- Appoint a Scrum Master
- Identify Backlog
- Establish and Conduct Daily Scrum Meeting

Start the Scrum Process

Define the team consisting of pigs (people who are assigned work) and chickens (people who are interested, but are not working). Identify pigs that will compose the Scrum team:

- No more than 6-9 members per team

- If more members than manageable, break into multiple Scrums
- Each Scrum focuses on one, self-contained area of work
- All staff performing work in this area

Appoint Scrum Master

The Scrum Master is the person who conducts the Scrum meetings, empirically measures progress, makes decisions, and gets impledíments out of the way of slowing or stopping work. This is often the engineering or marketing manager for this product or system area.

- Person who asks all pigs three questions (should also be a pig)
  - What did you do since last Scrum
  - What got in your way of doing work
  - What will you do before the next Scrum
- Must be able to make immediate decisions
- Better to ask forgiveness than ask permission
- Must resolve work impediments ASAP
- Identifies initial backlog

Identify Backlog

Backlog is all of the work that is outstanding for a product area, both immediate and well-defined, and long terms and visionary.

- List the known work to be done
- Group it into increments that should take no more than 30 days
- In areas where work is volatile or cannot be fully defined for up to 30 days, establish an increment for known horizon
- Lists all outstanding work to be done
- Only one person in charge of backlog prioritization
- Team chooses backlog for Sprint
  - Sprint is an increment of work that can be completed in less than 30 days and is well enough defined that it can be accurately estimated
  - Can be completed in less than 30 days
  - Can be reasonable estimated by backlog item
- Backlog is signed up for by team members
- Only this backlog is worked on during this Sprint for this area

Establish and Conduct Daily Scrum Meeting

The daily Scrum meeting is a status check where the team meets and updates each other about what's going on. It provides a daily focus on the work being done:

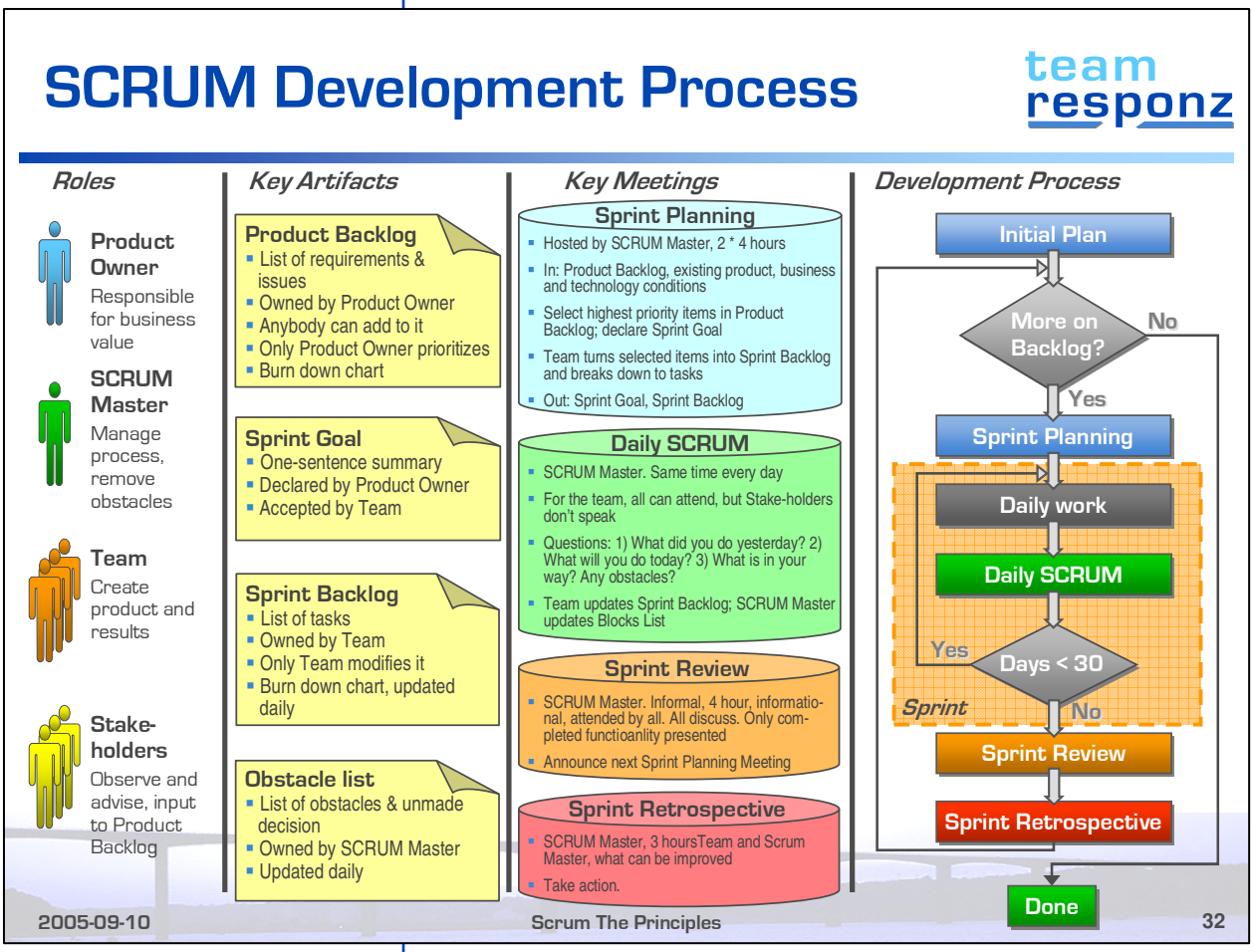
- Same time and place

- Avoids overhead of finding a place daily
- Avoids overhead of team trying to figure out where and when is today's meeting

- Let's chicken's know where and when
- No more than 15 minutes
- Daily Scrum Meeting agenda: Scrum master asks pigs the 3 questions:
- Scrum master is responsible for making decisions
- Scrum master is responsible for noting and resolving work impediments
- All discussion other than replies to 3 questions deferred to later meetings

Why Scrum Is Powerful

- Focus is on team's work, and team's work only
- Daily communication of status occurs
- Enables low-overhead empirical management
- Makes impediments visible
- Someone is willing to make decisions and remove impedi-ments in real-time



- Whitepaper – SCRUM Common Sense in Action

Bottom-Up Continuous Processing Improvement

Using Scrum causes Bottom-up Continuous Process Improvement as impediments to progress are identified during the daily Scrum meetings and removed. The highest priority and focus of the organization is to generate quality, profitable, competitive software -- all other processes are either supportive or require re-engineering:

Any development organization taking on the Scrum process, expecting the benefits of greater productivity and regular, competitive release, also takes on the work of incrementally reengineering the organization --Scrum Causes Change.

Scrum as Continuous Bottom-Up Development Process Re-Engineering

Scrum, is a bottom-up, continuous business process and reengineering process for software development organizations. Scrum focuses on maximizing software development productivity, and uses the following assumptions:

- The organization is devoted primarily to producing competitive, profitable software products, and that all process are subservient to this goal;
- The organization will do the best that it can to correct anything that is in the way of the goal;
- The organization realizes that re-engineering any process is difficult, and that the speed at which Scrum uncovers productivity impediments are requires their correction also causes pain and commitment.

Scrum can be implemented anywhere in the development process -- strategic planning, product planning, design, development, implementation. All other processes that reduce the effectiveness of that area are progressively identified.

Scrum's Secrets: Focus and Visibility

For any particular area where Scrum is implemented, focus and visibility occur in daily Scrum meetings.

- **Focus**: The work being performed right then is identified and assessed.
- **Visibility**: The work being performed and anything getting in the way of its completion is made visible every day.

Scrum's Three Tools

- **Backlog**: Overall, by product line, by product, and by system an organization identifies all outstanding work and prioritizes it. This prioritized backlog list changes continuously, and is updated and re-prioitized continuously.
- **Sprints**: Sprints are work increments, where a team works on completing an identified, self-contained group of prioritized work. During the sprint, the work is not changed from outside the sprint, although as work occurs in the sprint, additional work may be uncovered.

- **Scrums**: Daily meetings where a Sprint team meets to identify what work was just done, what work will be done next, and what is impeding work.

Examples of Re-Engineering

Implementing Scrum anywhere in a development organization causes process re-engineering. For instance:

- Identifying the backlog for a planning session may point out that management doesn't have clear marketplace vision and can't clearly identify what they want to do.
- Identifying backlog for a product may identify that the competition is now well understood.
- Identifying backlog for design may point out that adequate design techniques are not being used.
- Grouping backlog for a Sprint may point out that priorities are unclear, or that too many people are trying to control priorities.
- Conducting Sprints with an identified backlog may point out that worker are constantly being interrupted by interests outside the Sprint and cannot complete top priority assignments.
- Daily Scrums may point out the absence of version control mechanisms as daily builds don't occur because of overlaid or unknown code.
- Daily Scrums may point out that people have far less time to devote to the work than management had thought.
- Daily Scrums may point out that workers are far less competent than previously assumed, or are in power struggles that stop progress.
- Daily Scrums may point out that staff spends hours per day getting coffee across the street because none is supplied internally.

All of these impediments and far more are identified as Scrum is implemented. The simple implementation process causes all of these symptoms to be identified. The challenge then is for management to address them ... both individually and by identifying the underlying causes and re-engineering those processes.

Rapid Bottom-Up Re-Engineering as Painful but Necessary

Although process engineering groups are useful for recording and tracking the resolution of impediments, the re-engineering work belongs to management. This causes management pain, because it puts vested interests at odds with the overall good of the organization. Only the assumption that the organization's top goal is to remain in business, and that all else is subordinate, is adequate for these vested interests and eccentricities to be given up.

Scrum does not imply that there is only one way to run a software development organization. There are many ways. Continuous identification of impediments provides management with daily opportunity to re-engineer, and to track the effectiveness of

the re-engineering - as work attempts to progress within the re-engineered environment.

Some organizations may choose to be less than efficient. High margins and unique products allow organizations to luxuriate in such practices as excessive meetings, offsites, hierarchical organizations, excessive and conflicting management, incompetence, etc. However, the market is unforgiving. Competition ensures that sooner or later, the profits will be noticed by another more efficient organization. Or technology will pass the organization by, and it will be unable to respond.

Scrum Causes Change

Scrum clarifies the roles of management and workers, helping each focus on their work and not waste time on the other's work:

- The role of management is to assess the marketplace, set direction and priorities, and to provide resources.
- The role of workers (engineers) is to build the best possible products that they know how.
- The assumption of workers is that management is competent at their job and their work will sell and provide money that they need to live.
- The assumption of management is that the workers will apply their skills as best they know how to build these products.

Given these assigned roles, management and workers can use the following contracts. The contract from management to worker is:

- We know the marketplace.
- We know what to build to successfully compete.
- We have provided adequate resources to build product that will generate enough revenue for us to be at least self-sustaining.
- We will provide a workplace in which your productivity is maximized.
- We will pay you a fair salary.

The contract from worker to management is:

- We understand how to do our work.
- We will apply our skills as best we can to build the products you identify.
- We will constantly assess our industry to ensure that our skills are the best possible and that we use the best technology possible.

Scrum Rules

Scrum is an iterative, incremental process for developing software in chaotic environments. Scrum consists of a series of 30 day sprints, each sprint producing an executable. Between sprints, all interested parties evaluate progress and reevaluate technical and business requirements. Work is reestablished and the team enters into another sprint.

■ Whitepaper – SCRUM Common Sense in Action

The pulse of Scrum is the key to its success ... management determines what should be done prior to every sprint, their determination influenced by prior deliverables and requirements. During the sprint, the team is left alone and produces the best software possible: Let in chaos, keep out chaos, let in chaos, keep out chaos, let in chaos, keep out chaos ... etc.

Backlog

A prioritized list of all work to be completed prior to releasing a product:

- Only one person maintains and prioritizes the backlog list.
- Any interested party can request that backlog be put on the list.

Between sprints, all involved parties and the engineering team meet to determine which work can be completed in the next sprint, and what the executable will be.

Sprint

A short burst of work lasting approximately 30 days during which an executable and other deliverables are built by an engineering team, as indicated by the assigned backlog.

- A sprint lasts no more than 30 days.
- A sprint is undertaken by a cross functional team consisting of no more than 9 members.
- Every sprint has a specific goal.
- An executable demonstrating the goal will be completed by the team during the sprint.
- The sprint team has final say in estimating and determining what they can accomplish during the sprint.
- Once the sprint is underway, new backlog cannot be added to the sprint except that, if the scrum master determines that a new backlog item will enhance the viability of the product, is in alignment with the sprint, builds on the sprint's executable, and can be completed within the sprint's time frame, the backlog item can be added. Examples are building a demonstration of the executable for a specific purpose, such as a trade show or prospect.
- If external forces determine that the sprint is working on the wrong thing, a sprint can be halted and restarted with new backlog and purpose.

Scrum Meeting

A Scrum meeting is a short daily meeting where the team shares status.

- During the sprint, the team conducts daily scrum meetings.
- The meetings are held in the same place at the same time every work day.
- The meetings don't last for more than 30 minutes.
- A scrum master is appointed. He is responsible for asking every team member the following three questions:

- What have you done since the last scrum meeting?
- What has impeded your work?
- What do you plan on doing between now and the next scrum meeting?

- Conversation is restricted to the team members answering the above questions.
- Meetings can be established for immediately after the scrum meeting based on answers to the above questions.
- The scrum master is responsible for making decisions immediately, if required to remove impediments to progress.
- The scrum master is responsible for noting impediments that must be resolved external to the meeting and causing them to be removed.

Vocabulary

Scrum provides a language for this common sense way of organizing, performing and managing work:

Backlog All work to be performed in the forseeable future, both well defined and requiring further definition.

Sprint A period of 30 days of less where a set of work will be performed to create a deliverable.

Sprint Backlog That work that is well-enough defined that it can be worked on with relatively little change over a period of 30 days or less and will result in a tangible, incremental deliverable.

Scrum A daily meeting at which progress and impediments to progress are reviewed.

Scrum Meeting Rules Protocol for effective Scrum daily meetings.

Scrum Team The cross-functional team working on the sprint's backlog.

Frequently Asked Questions

1. Is Scrum a methodology?

No – Scrum is a framework within which management and teams can do the best that they can to build any software in any environment. A number of common sense practices, processes, and techniques used within the software industry are combined into this framework.

2. When is Scrum appropriate?

When a project is important and no one has confidence that any existing approach will work.

3. What type of projects have used Scrum and failed?

Projects where something other than getting the software produced is more important. For instance, financial constraints may constrain the development effort to such a degree that the software cannot be delivered in time to be relevant. Or, organizational

politics cause usual problems and are not removed when escalated as impediments.

4. What is important to the success of Scrum?

Common sense, good engineering practices, good engineers and management, and at least one person to whom the project is so important that they are willing to be fired rather than have the project fail.

5. How do you handle geographically dispersed teams using Scrum?

Daily Scrum meetings are conference called to a common, central location with speaker phone facilities. All team members attend and respond. At the end of each Sprint, the entire team meets in one location for demonstrating the increment's results and planning the next increment.

6. How do I track a project's progress when Scrum is used?

Assess the number of product features that have been completed and are part of a release. Features that were part of a incomplete sprint, or features that are completed but not used in a release are not counted. For detailed sizing, count the function points of the released features.

The most meaningful way of tracking a team's progress, attend each end-of-Sprint demonstration. Cumulative increments of software are demonstrated and the team responds to detailed questions about its operation, architecture, and stability.

7. How do I track a team's progress during a Sprint?

During a Sprint, a team updates the estimated number of hours to finish a task. When the Sprint starts, this number of hours is the initially estimated hours. As work progresses, the remaining hours should decrease as work is applied, unless work is not being applied or, as work is being applied, tasks are larger than expected. A "burndown" graph with days on the Y axis and number of hours remaining on all tasks in the Sprint on the X axis provides a visual trend indicating the probability that work will be completed by the end of the Sprint.

An empirical way of determining a team's progress is to listen during the daily Scrum meeting. The tone of voice, the way work is being described, and the impediments give an empirical feel for a team's progress.

8. When and how do I use pert charts?

Pert charts are good for plotting overall project dependencies. However, pert charts are not used for tracking tasks during Sprints. A Sprint is sized (30 days, less than 8 workers) so that tasks can be kept track of by the team without any complicated mechanisms such as pert charts.

9. How does our standard methodology work with Scrum?

Scrum is an iterative, incremental approach for building project deliverables. The sequence in which deliverables are built is still dictated by good engineering practices - requirements, architecture, design, and code. With Scrum, testing and documentation are performed in parallel with development, and tangible software that can be incrementally be built upon is produced by each Sprint.

10. Can I only use the daily Scrum meetings initially and move on to the other parts of Scrum later?

Scrum is intended to improve productivity by letting teams and management be as responsive as possible to development conditions. The daily meetings are great for reducing the need for other meetings, for quick daily status, and keeping management up to speed, but without the other mechanisms the spirit and productivity of Scrum have been removed.

11. Do I need daily Scrum meetings, or can we use 3 Scrum meetings every week?

Daily, so that management can remove impediments as soon as they occur and everyone knows what's going on. We've heard suggestions about team's emailing impediments, but that misses the point – Scrum changes management's role from being remote, to being an immediate source of aid. The team doesn't have to hunt for help, help is immediately available.

12. How does Scrum compare to the CMM process?

Scrum is a framework within which an institution's regular processes occur. Such CMM processes as testing, inspections, and requirements analysis may be used within a Scrum project. The difference, however, is that the use of these processes is determined empirically by need ... if the team feels that they are needed, they are used; otherwise, they are not used.

13. What customer involvement would Scrum require?

Because Scrum is designed to be responsive to the changing dynamics surrounding a project, the customer determines the prioritization of features at the beginning of a sprint. The customer continually provides feedback on product design throughout a sprint and, more specifically, at the end of the sprint when the implemented features are demonstrated. Optionally, the customer can sit in the daily 15-minute-(or less) Scrum meeting for project updates.

14. How is the daily Scrum meeting different from the conventional daily status meeting?

It is kept to a minimum in consideration of people's tight schedule. Participants answer 3 questions:

- What I did in the last 24 hours?
- What I plan to do in the next 24 hours?
- What obstacles are standing in my way?

One key element that makes the daily Scrum meeting different from other conventional daily meeting is the face-to-face hand-off of obstacles to management. The management will then remove the obstacles standing in the way of the team and product delivery.

15. How does Scrum handle complex projects made up of multiple interdependent teams running concurrently?

Individual teams have their daily Scrum meeting, where core members provide updates and members from other teams can sit in to hear the progress and obstacles as they arise. Depending on the degree of interdependency among the teams, an additional Scrum meeting by designated members from each team can provide a fast and structured forum to monitor progress on cross-team requirements and issues.

16. How does Scrum reduce development cycle time?

One fundamental attribute of Scrum is the formation of a cross-functional team where each member develops their aspect of the selected features concurrently. For example, members representing Testing and Documentation participate in sprints together with developers, and they produce their deliverables incrementally and iteratively in support of the sprint goals. This approach avoids the big bang that typically happens near the end of a project, and facilitates communication across all development disciplines throughout the project.

References

Web

www.controlchaos.com

www.agilealliance.org

www.scrum.dk

www.scrumeducation.com

www.estherderby.com

www.xprogramming.com

www.mountaingoatsoftware.com

www.jeffsutherland.com

www.martinfowler.com

Books

Agile Software Development with Scrum – Ken Schwaber, Mike Beedle

Agile Project Management – Ken Schwaber