What Is Scrum?

by Ken Schwaber

Software development is a complex endeavor. Its results are ephemeral, consisting of signals that control machines. The process is entirely intellectual, with all intermediate products being marginal representations of the thoughts involved. The materials used to create the end product are extremely volatile: user requirements of what the users have yet to see, the interoperation of other program's signals with our programs, and the interactions of the most complex processes yet – a team of people working together. Such a complex process requires empirical, rather than defined, process control. Scrum is a simple set of practices and rules that encompasses the transparency, inspection, and adaptation requirements inherent in empirical process control.

Empirical Process Control

Laying out a process that will produce acceptable quality output over and over again is called *defined* process control. We try to use defined processes whenever possible because with them we can crank up unattended production to such a quantity that the output can be priced as a commodity. However, if the commodity is of such unacceptable quality as to be unusable, the rework too great to make the price acceptable, or the cost of unacceptably low yields is too high, we have to turn to and accept the higher costs of *empirical* process control. In the long run, making successful products the first time using empirical process control has turned out to be much cheaper than reworking many unsuccessful products using defined process control.

Empirical process control has three legs underlying all of its implementations: transparency, inspection, and adaptation. Transparency means that those aspects of the process that affect the outcome must be visible and known to those controlling the process. Inspection requires that various aspects of the process be inspected frequently enough so that unacceptable variances in the process can be detected. This frequency must take into consideration the fact that all processes are changed by the act of inspection. Additionally, the inspector must possess the skills to assess what he or she is inspecting. The third leg of empirical process control is adaptation. If the inspector determines from the inspection that one or more aspects of the process are outside acceptable limits, and that the resulting product will be of unacceptable, the inspector must adjust the process or the material being processed. The adjustment must be made as quickly as possible to minimize further deviation.

An example of an empirical process control in software development is a code review. The code is reviewed against coding standards and industry best practices. Everyone involved in the review fully and mutually understands these standards and best practices. The code review occurs whenever someone feels that a section of code is complete. The most experienced developers review the code, and their comments and suggestions lead to the developer adjusting his or her code.

Scrum: Skeleton and Heart

Scrum addresses the complexity of software development projects by implementing the inspection, adaptation, and visibility requirements of empirical process control in a set of simple practices and rules. When I say control, I don't mean control to create what we predict. I mean that we will control the process to guide the work toward the most valuable outcome possible.

Scrum employs an iterative, incremental process skeleton on which hang all of its practices. The skeleton operates this way: At the start of each iteration, the team reviews what it must do. Then, it selects what it believes it can turn into an increment of potentially shippable functionality by the end of the iteration. The team is then left alone to make its best effort for the rest of the iteration. At the end of the iteration, the team presents the increment of functionality that it built so that the stakeholders can inspect it and make timely adaptations to the project.

The heart of Scrum occurs within the iteration. The team takes a look at the requirements, the technology, and evaluates each other's skills and capabilities. The team then collectively devises the best way it knows to build the

functionality, modifying the approach daily as it encounters new complexities, difficulties, and surprises. The team figures out what needs to be done, and determines the best way to do it. This creative process is the heart of the Scrum's productivity.

Scrum: Roles

Scrum implements this iterative, incremental skeleton through three roles: the Product Owner, the Team, and the ScrumMaster. All management responsibilities in a project are divided between these three roles.

The Product Owner is responsible for representing the interests of everyone with a stake in the project and its resulting product. The Product Owner achieves initial and on-going funding for the project by creating the project's initial overall requirements, return on investment objectives, and release plans. The list of requirements is called the Product Backlog. The Product Owner is responsible for using the Product Backlog to ensure that the most valuable functionality is produced first and built upon; this is achieved by frequently prioritizing the Product Backlog to queue up the most valuable requirements for the next iteration. The Product Owner is responsible for the success of the product.

The Team is responsible for developing functionality. Teams are self-managing, self-organizing, and cross-functional. A Team is responsible for figuring out how to turn the Product Backlog into an increment of functionality within an iteration, and managing its own work to do so. The Team members are collectively responsible for the success of each iteration.

The ScrumMaster is responsible for the Scrum process, for teaching it to everyone involved in the project, for implementing it so it fits within an organization's culture and still delivers the expected benefits, and for ensuring that everyone follows its rules and practices.

Scrum: Flow

A Scrum project starts with a vision of the system and a simple, baseline plan of cost and timeframes. The vision may be vague at first, stated in market terms rather than product terms. The vision will become clearer as the project moves forward. The Product Owner is responsible to those funding the project to deliver the vision in a manner that maximizes their return on investment. The Product Owner formulates a plan for doing so which includes a Product Backlog. The Product Backlog is a list of functional and non-functional requirements that, when turned into functionality, will deliver this vision. The Product Backlog is prioritized so that the items most likely to generate value are top priority. The Product Backlog is divided into proposed releases. This is a starting point, and the contents, priorities, and grouping of the Product Backlog into releases is expected to and usually does change the moment the project starts. Changes in the Product Backlog reflect changing business requirements and how quickly or slowly the Team can transform the Product Backlog into functionality.

All work is done in Sprints. Each Sprint is an iteration of one month. Each Sprint is initiated with a Sprint Planning meeting, where the Product Owner and Team get together to collaborate about what will be done for the next Sprint. The Sprint Planning meeting has two parts. The first four hours are spent with the Product Owner presenting the highest priority Product Backlog to the Team. The Team questions him or her about the content, purpose, meaning, and intentions of the Product Backlog. When the Team knows enough, but before the first four hours elapses, the Team selects as much Product Backlog as it believes that it can turn into a completed increment of potentially shippable product functionality by the end of the Sprint. The Team commits to do its best to do so to the Product Owner.

During the second four-hours of the Sprint Planning meeting, the Team plans out the Sprint. It creates a design within which the work can be done. Scrum requires Teams to build an increment of product functionality every Sprint. This increment must be potentially shippable, for the Product Owner may choose to immediately implement the functionality. Each increment must consist of thoroughly tested, well-structured and written code that has been built into an executable. The user operation of the functionality must be documented, either in Help files or user documentation. This is the definition of a "done" increment and it should factor into how much work a team can take on in a Sprint. It takes some development organizations awhile to be capable of building something this "done."

Since the Team is responsible for managing its own work, it needs a tentative plan to start the Sprint. The tasks that comprise this plan

are placed in a Sprint Backlog; the tasks in the Sprint Backlog emerge as the Sprint evolves. At the start of the second four-hour period of the Sprint Planning meeting, the Sprint has started and the clock is ticking toward the month-long Sprint time-box. Note that Sprint Planning meetings cannot last longer than eight hours. They are timeboxed to avoid too much hand-wringing about what is possible. The goal is to get to work, not to think about working.

Every day the team gets together for a fifteen minute meeting called a Daily Scrum. At the Daily Scrum, each Team member answers three questions: What have you done on this project since the last Daily Scrum meeting? What do you plan on doing on this project between now and the next Daily Scrum meeting? And, what impediments are in the way of you meeting your commitments toward this Sprint and this project? The purpose of the meeting is to synchronize the work of all team members daily and to schedule any meetings that the Team needs to forward its progress. The team members are inspecting each others work in light of the team's commitments, and making adaptations to optimize their chance of meeting those commitments.

At the end of the Sprint, a Sprint Review meeting is held. This is a four-hour timeboxed meeting where the Team presents what was developed during the Sprint to the Product Owner and any other stakeholders that wish to attend. This is an informal meeting, with the presentation of the functionality intended to foster collaboration about what to do next based on what the Team just completed. The Product Owner and stake holder inspect the Team's work in light of project goals, and make adaptations to optimize their chance of reaching those goals.

After the Sprint Review and prior to the next Sprint Planning meeting, the ScrumMaster holds a Sprint Retrospective meeting with the Team. At this three hour, timeboxed meeting the ScrumMaster encourages the team to revise, within the Scrum process framework and practices, its develop-

Backlog Description	Initial Estimate	Adjustment Factor	Adjusted Estimate	f work remaining until completion						
				1	2	3	4	5	6	7
Title Import				256	209	193	140	140	140	140
Project selection or new	3	0.2	3.6	3.6	0	0	0	0	0	0
Template backlog for new projects	2	0.2	2.4	2.4	0	0	0	0	0	0
Create product backlog worksheet with formatting	3	0.2	3.6	3.6	0	0	0	0	0	0
Create sprint backlog worksheet with formatting	3	0.2	3.6	3.6	0	0	0	0	0	0
Display tree view of product backlog, releases, sprints	2	0.2	2.4	2.4	0	0	0	0	0	0
Sprint-1	13	0.2	15.6	16	0	0	0	0	0	0
Create a new window containing product backlog template	3	0.2	3.6	3.6	3.6	0	0	0	0	0
Create a new window containing sprint backlog template	2	0.2	2.4	2.4	2.4	0	0	0	0	0
Burndown window of product backlog	5	0.2	6	6	6	0	0	0	0	0
Burndown window of sprint backlog	1	0.2	1.2	1.2	1.2	0	0	0	0	0
Display tree view of product backlog, releases, sprints	2	0.2	2.4	2.4	2.4	0	0	0	0	0
Display burndown for selected sprint or release	3	0.2	3.6	3.6	3.6	0	0	0	0	0
Sprint-2	16	0.2	19.2	19	19	1.2	0	0	0	0
Automatic recalculating of values and totals	3	0.2	3.6	3.6	3.6	3.6	0	0	0	0
As changes are made to backlog in secondary window, update burndown graph on main page	2	0.2	2.4	2.4	2.4	2.4	0	0	0	0
Hide/automatic redisplay of burndown window	3	0.2	3.6	3.6	3.6	3.6	0	0	0	0
Insert Sprint capability ... adds summing Sprint row	2	0.2	2.4	2.4	2.4	2.4	0	0	0	0
Insert Release capability adds summary row for backlog in Sprint	1	0.2	1.2	1.2	1.2	1.2	0	0	0	0
Owner/assigned capability and columns optional	2	0.2	2.4	2.4	2.4	2.4	0	0	0	0
Print burndown graphs	1	0.2	1.2	1.2	1.2	1.2	0	0	0	0
Sprint-3	14	0.2	16.8	17	17	17	0	0	0	0
Duplicate incomplete backlog without affecting totals	5	0.2	6	6	6	6	6	6	6	6
Note capability	6	0.2	7.2	7.2	7.2	7.2	7.2	7.2	7.2	7.2
What-if release capability on burndown graph	15	0.2	18	18	18	18	18	18	18	18
Trend capability on burndown window	2	0.2	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4
Publish facility for entire project, publishing it as HTML web pages	11	0.2	13.2	0	0	13	13	13	13	13
Future Sprints	39	0.2	46.8	34	34	47	47	47	47	47
Release-1				85	70	65	47	47	47	47

Figure 1. Sample Product Backlog

ment process to make it more effective and enjoyable for the next Sprint.

Collectively, the Sprint Planning meeting, the Daily Scrum meeting, the Sprint Review meeting, and the Sprint Retrospective meeting implement the empirical inspection and adaptation practices within Scrum.

Scrum: Artifacts

Product Backlog

The requirements for the product being developed by the project(s) are listed in the Product Backlog. The Product Owner is responsible for the Product Backlog, its contents, its availability, and its prioritization. The Product Backlog is never complete, and the Product Backlog in the project plan only lays out the initially known and best-understood requirements. The Product Backlog evolves as the product and the environment in which it will be used emerge. The Product Backlog is dynamic, in that management constantly changes it to identify what the product needs to be appropriate, competitive, and useful. As long as a product exists, the Product Backlog also exists. An example of a Product Backlog maintained on the Scrum Product Management tool, based in a spreadsheet, is shown in Figure 1, previous page.

This spreadsheet is the Product Backlog in March 2003, from a project for developing the Scrum Project Management software. The rows are the Product Backlog items, interspersed by Sprint and Release dividers. For instance, all of the rows above Sprint 1 were worked on in that Sprint. The rows between Sprint 1 and Sprint 2 rows were done in Sprint 2. Notice that the row "Display tree view of product backlog, releases, sprints" is duplicated in Sprint 1 and Sprint2. This is because row 10 wasn't completed in Sprint 1, so it was moved down to the Sprint 2 for completion.

The first four columns are the Product Backlog item name, the initial estimate, the complexity factor, and the adjusted estimate. The complexity factor increases the estimate due to project characteristics that reduce the productivity of the Team. The next columns are the Sprints during which the Product Backlog is developed. When the Product Backlog is first thought of and entered, its estimated work is placed into the column of the Sprint that is going on at that time. The developers and product owner devised most of the backlog items shown before starting this project. The sole exception is row 31 (Publish facility for entire project, publishing it as HTML web pages),

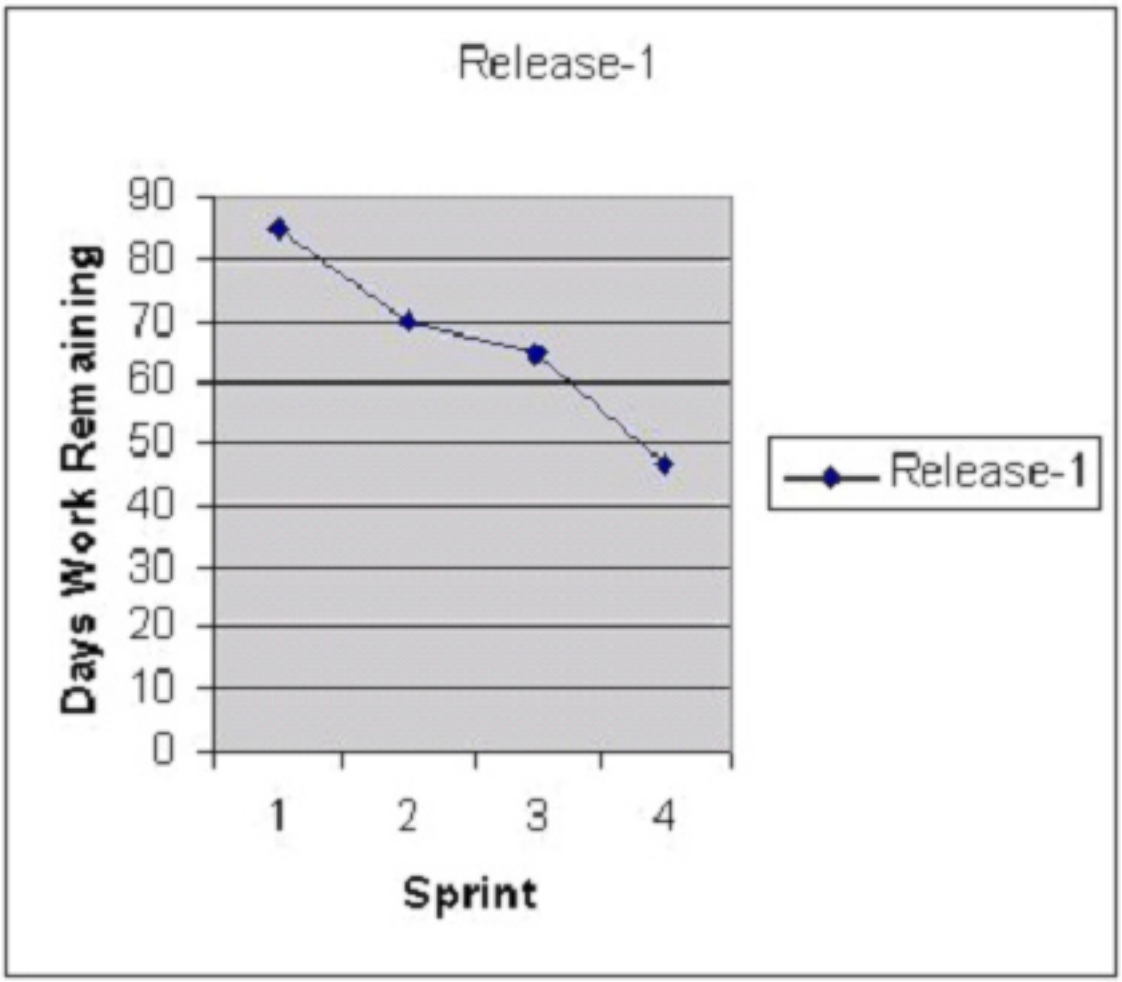

Figure 2. Sample Burndown Chart

which wasn't thought of until during Sprint 3.

Burndown Chart

A burndown chart shows the amount of work remaining across time. The burndown chart is an excellent way of visualizing the correlation between the amount of work remaining at any point in time and the progress of the project team(s) in reducing this work. The intersection of a trend line for work remaining and the horizontal axis indicates the most probable completion of work at that point in time. A burndown chart reflecting this is in Figure 2. The burndown chart helps me to "what if" the project by adding and removing functionality from the release to get a more acceptable date, or extending the date to include more functionality. The burndown chart is the collision of reality (work done and how fast it's being done) with what is planned, or hoped for.

Sprint Backlog

The Sprint Backlog defines the work, or tasks, that a Team defines for turning the Product Backlog it selected for that Sprint into an increment of potentially shippable product functionality. The Team compiles an initial list of these tasks in the second part of the Sprint Planning meeting. Tasks should have enough detail so that each task takes roughly four to sixteen hours to finish. Tasks that are of longer estimated time are used as placeholders for tasks that haven't been finely defined. Only the team can change the Sprint Backlog. The Sprint Backlog is a highly visible, real time picture of the work that the team plans to accomplish during the Sprint. An example Sprint Backlog is in Figure A-3. The rows represent Sprint Backlog tasks. The columns represent the days in the month of the Sprint. Once a task is defined, the estimated number of hours remaining to complete the task is place in the intersection of the task and the Sprint day by the person working on the task.

Final Thoughts

There is no panacea for the complexities of software development. Scrum is devised specifically to wrest usable products from complex problems. It has been used successfully on thousands of projects in hundreds of organizations over the last sixteen years. Scrum is not for those who seek easy answers and simple solutions to complex problems; it is for those who understand that complex problems can only be met head on with determination and wit.

Task Description	Originator	Responsible	Status (Not Started / In Progress / Completed)	Hours of work remaining unt											
				1	2	3	4	5	6	7	8	9	10	11	12
Meet to discuss the goals and features for Sprint 3-6	Danielle	Danielle/Sue	Completed	20	0	0	0	0	0	0	0	0	0	0	0
Move Calculations out of Crystal Reports	Jim	Allen	Not Started	8	8	8	8	8	8	8	8	8	8	8	8
Get KEG Data		Tom	Completed	12	0	0	0	0	0	0	0	0	0	0	0
Analyse KEG Data - Title		George	In Progress	24	24	24	24	12	10	10	10	10	10	10	10
Analyse KEG Data - Parcel		Tim	Completed	12	12	12	12	12	4	4	4	0	0	0	0
Analyse KEG Data - Encumbrance		Josh	InProgress							12	10	10	10	10	10
Analyse KEG Data - Contact		Danielle	In Progress	24	24	24	24	12	10	8	6	6	6	6	6
Analyse KEG Data - Facilities		Allen	In Progress	24	24	24	24	12	10	10	10	10	10	10	10
Define & build Database		Barry/Dave	In Progress	80	80	80	80	80	60	60	60	60	60	60	60
Validate the size of the KEG database		Tim	Not Started												
Look at KEG Data on the G:\		Dave	In Progress	3	3	3	3	3	3	3	3	3	3	3	3
Confirm Agreement with KEG		Sue	Not Started												
Confirm KEG Staff Availability		Tom	Not Started	1	1	1	1	1	1	1	1	1	1	1	1
Switch JDK to 1.3.1. Run all tests.		Allen	Not Started	8	8	8	8	8	8	8	8	8	8	8	8
Store PDF files in a structure		Jacquie	Completed	8	0	0	0	0	0	0	0	0	0	0	0
TopLink: Cannot get rid of netscape parser		Richard	Completed	4	0	0	0	0	0	0	0	0	0	0	0
Build test data repository		Barry	In Progress	10	10	10	10	10	10	10	10	8	8	8	8
Move application and database to Qual (incl Crystal)		Richard	Completed	4	4	4	4	4	4	4	0	0	0	0	0
Set up Crystal environment		Josh	Completed	2	2	2	2	1	1	1	0	0	0	0	0
Test App in Qual		Sue	In Progress												20
Defining sprint goal required for solution in 2002		Lynne	In Progress	40	40	40	40	40	40	40	38	38	38	38	38
Reference tables for import process		Josh	In Progress												
Build standard import exception process		Josh	InProgress									12	12	12	10
Handle multiple file imports on same page		Jacquie	Disregarded	20	15	15	15	12	12	12	12	9	0	0	0
Migrate CruiseControl Servlet to IWS 6.0 (landcc_7101) server		Allen	Not Started	4	4	4	4	4	4	4	4	4	4	4	4
Create web server for Qual on PF1D8		Allen	Completed	1	0	0	0	0	0	0	0	0	0	0	0
LTCS Disk		Danielle/George	In Progress	12	12	12	12	8	8	8	8	8	8	8	8
Follow thru with questions about KEG data to Sue/Tom, re: Keg, LTO	Jacquie	Danielle	Completed	10	10	10	10	10	8	8	0	0	0	0	0
Map KEG data to Active Tables - see also #14	Jacquie	Jacquie/Allen	In Progress	50	50	50	50	50	50	50	50	50	50	50	50
Prepare SQL to import from KEG tables to Active Tables	Jacquie	George	In Progress	25	25	25	25	25	25	25	25	25	24	23	22

Figure 3. Sample Scrum Backlog