CONTENTS INCLUDE:

Scrum

By Michael James

ABOUT SCRUM

Scrum is a simple management framework for incremental product development using one or more cross-functional, self-organizing teams of about seven people each.

Scrum teams use fixed length iterations, called Sprints, typically two weeks or 30 days long. They attempt to build a potentially shippable (properly tested) product increment every iteration.

An Alternative to Waterfall

Scrum's incremental, iterative approach trades the traditional phases of "waterfall" development for the ability to develop a subset of high-business value features first, incorporating user feedback sooner.

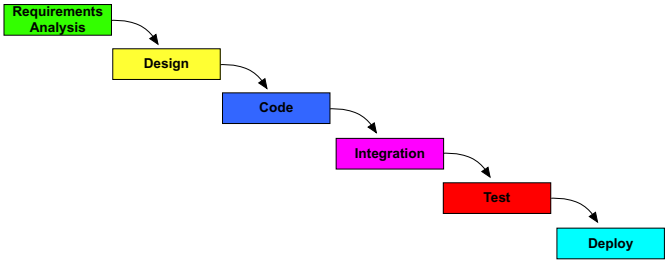

Figure 1: Traditional "waterfall" development assumes perfect understanding of requirements at outset.

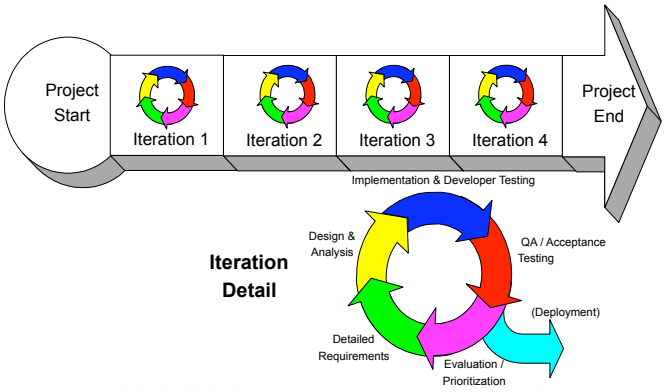

Figure 2: Scrum blends all development activities into every iteration, adapting to discovered realities at fixed intervals.

Scrum has been used for a variety of products, but has initially been most popular for software products using object-oriented technologies. It is particularly suited to high risk endeavors where traditional efficiency concerns are secondary to the ability to deliver the right product, or any product, by the required date.

A Disruptive Framework to Transform Organizations

The reality checks forced by the short feedback loops are intended to expose dysfunction at the individual, team, and organizational level. Rather than modify Scrum to mask these dysfunctions, organizations are encouraged to challenge these constraints and transform themselves.

Scrum is a framework, not a defined process or methodology. Scrum provides a simple structure of roles, meetings, rules, and artifacts[1]. Scrum teams are responsible for creating and adapting their processes within this framework. Scrum's management practices are similar to those of eXtreme Programming (XP), but, unlike XP, Scrum does not prescribe specific engineering practices.

SCRUM ROLES

Product Owner

The Product Owner is the single individual responsible for return on investment (ROI) of the product development effort. The Product Owner owns the product vision, constantly re-prioritizes the Product Backlog, and revises release plan expectations. The Product Owner is the final arbiter of requirements questions, including which items are considered "done" at the Sprint Review Meeting.

Scrum Development Team

The Team is a self-organizing/-managing group of about seven (give or take two) individuals. While the Team may contain specialists, collectively it is cross-functional, containing the range of skills (including testing) which were traditionally found in different departments. The Team is given autonomy regarding how to achieve the commitments they're held responsible for between the Sprint Planning and Sprint Review meetings.

A Scrum Development Team is most likely to succeed when members are co-located in a team room.

ScrumMaster

The ScrumMaster facilitates the Scrum process, keeps the Scrum artifacts visible, facilitates Team self-organization (keeping it in the "zone"), helps resolve impediments (at the team level and organizational level), shields the team from interference, and advocates improved engineering practices[2].

The ScrumMaster does these things (and more) without any authority on the Team. The ScrumMaster does not make business decisions or technical decisions, does not commit to work on behalf of the Team, etc.

SCRUM MEETINGS

All Scrum Meetings are facilitated by the ScrumMaster, though he has no decision-making authority at these meetings.

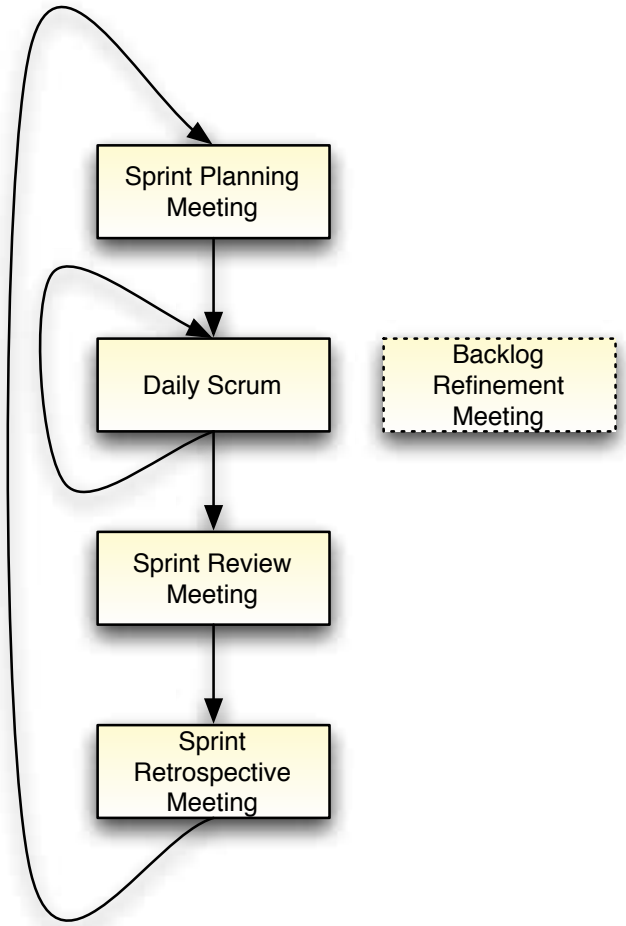

Figure 3: Scrum Meetings

Sprint Planning Meeting

Part 1: At the beginning of each iteration, the Product Owner and Team negotiate which Product Backlog Items the Team will attempt this Sprint. The Product Owner is responsible for declaring which items are the most important to the business, and the Team is responsible for committing to the amount of work they feel they can accomplish without accruing technical debt.

Part 2: The Team decomposes the selected Product Backlog Items into an initial list of Sprint Tasks and makes a final commitment to do the work. The Product Owner's full attendance is often not necessary during Part 2. The maximum time for planning a 30-day Sprint is 8 hours.

Daily Scrum

Every day, at the same time and place, the Scrum Development Team members spend 15 minutes reporting to each other. Each team member reports to the rest of the team what he did the previous day, what he will do today, and what impediments he has.

The team will typically examine the current Sprint Task list, Sprint Burndown Chart, and impediments list.

The Product Owner's attendance is often not necessary at the Daily Scrum and may actually impede team self-organization.

Topics that require additional discussion may be handled as optional sidebars after every team member has reported. It's a common practice to stand at this meeting to create a sense of urgency, so it's sometimes called the "standup meeting."

Reporting to an entire team, rather than to a boss, is one of the new habits Scrum team members learn.

Sprint Review Meeting

At the end of each Sprint execution, the Team demonstrates the actual working product increment they built to the Product Owner and other stakeholders. The Product Owner declares which committed items will be considered "done" according to the previously negotiated agreement. Incomplete items are returned to the Product Backlog as candidates for future Sprints. Feedback from stakeholders is converted to new Product Backlog Items.

The Sprint Review Meeting is an opportunity to inspect and adapt the product as it emerges and iteratively refine the understanding of the requirements. New products, particularly software products, are hard to visualize in a vacuum. Many customers need to be able to react to a piece of functioning software to discover what they actually want.

Sprint Retrospective

At the end of every Sprint, the team meets to reflect on its own process. They inspect their own behavior and take action to adapt it for future Sprints. This meeting provides an inspect-and-adapt mechanism for the team's process.

Techniques ScrumMasters can use to facilitate retrospectives[3] include silent writing, timelines, satisfaction histograms, and many others. Common topics include: "What went well?"; "What could be improved?"; "What did we learn?"; "What still puzzles us?"; "What actions will we take?"

As with the Daily Scrum, the team may choose to invite the Product Owner. Candid communication will help the team gain common understanding of multiple perspectives and come up with actions that will take the team to the next level.

Backlog Refinement Meeting

The team estimates the effort of items in the Product Backlog so the Product Owner can prioritize them before the next Sprint Planning Meeting. Large, vague items are split and clarified. New stories might be written by a subset of the team, in conjunction with the Product Owner and other stakeholders, before involving the entire team in estimation.

This meeting lacks an official name, thus may also be called "Backlog Maintenance," "Backlog Grooming," "Backlog Estimation Meeting," etc.

SCRUM ARTIFACTS

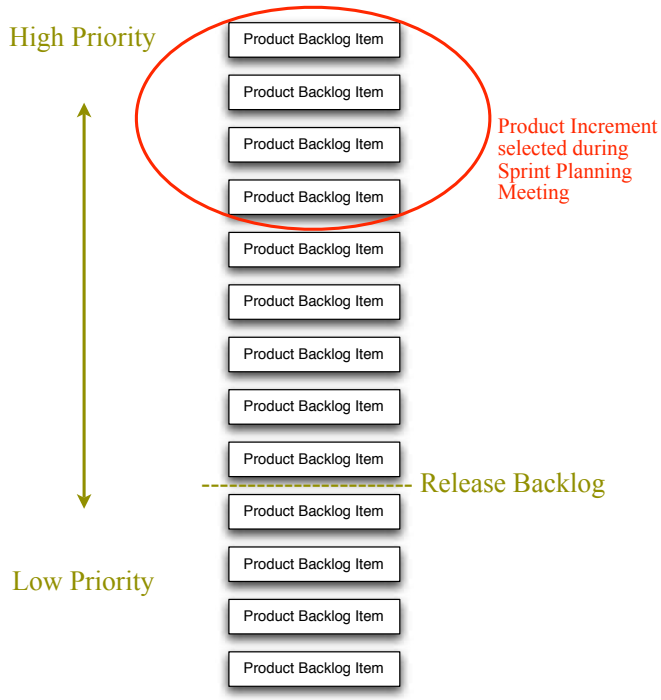

Figure 4: Product Backlog

Product Backlog

- Force-ranked list of desired functionality
- Visible to all stakeholders
- Any stakeholder (including team) can add items
- Constantly re-prioritized by Product Owner
- Items at top are more granular than items at bottom
- Maintained during Backlog Refinement Meeting

Product Backlog Item (PBI)

- Specifies the WHAT, not the HOW, of a customer-centric feature.
- Often written in "User Story" form
- Has acceptance criteria (and/or product-wide definition of "done") to prevent technical debt
- Effort is estimated by Team, ideally in relative units (e.g. story points)
- Business value is estimated by Product Owner

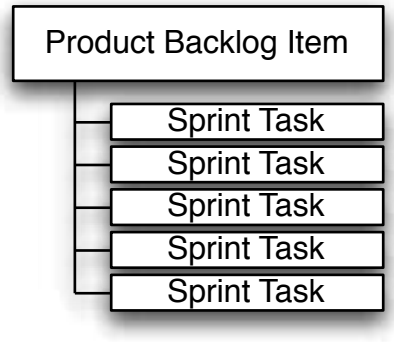

Figure 5: Each PBI represents a customer-centric feature, usually requiring several tasks to achieve definition of done.

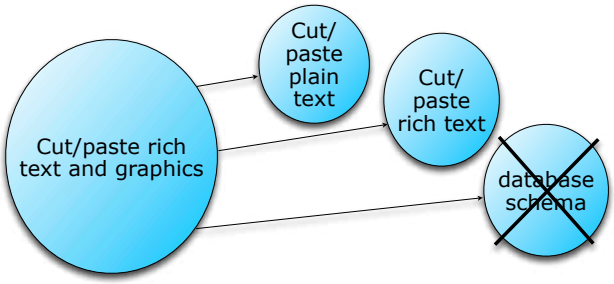

Figure 6: Large PBIs (often called "epics") split into thin vertical feature slices ("stories"), not horizontal implementation phases, when they rise toward the top of the Product Backlog.

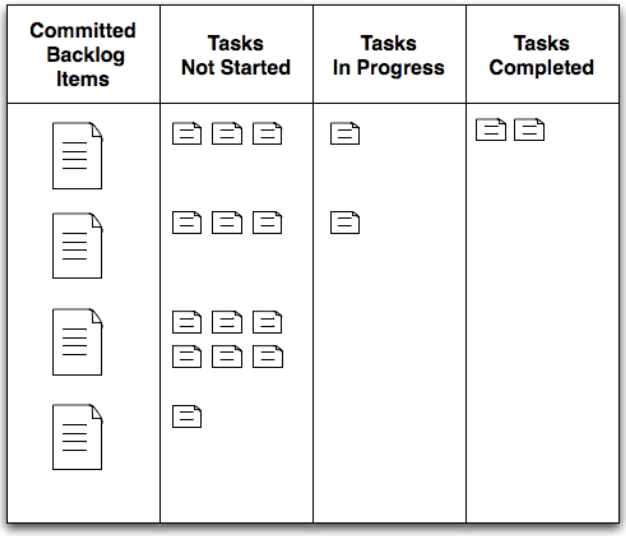

Figure 7: The Sprint Backlog is often represented on an "information radiator" such as a physical task board.

Sprint Backlog

- Committed PBIs negotiated between Team and Product Owner during Sprint Planning Meeting
- Scope Commitment is fixed during Sprint Execution
- Initial tasks created by Team during Sprint Planning Meeting, and expected to change during Sprint Execution
- Visible to Team (primarily)
- Referenced during Daily Scrum Meeting

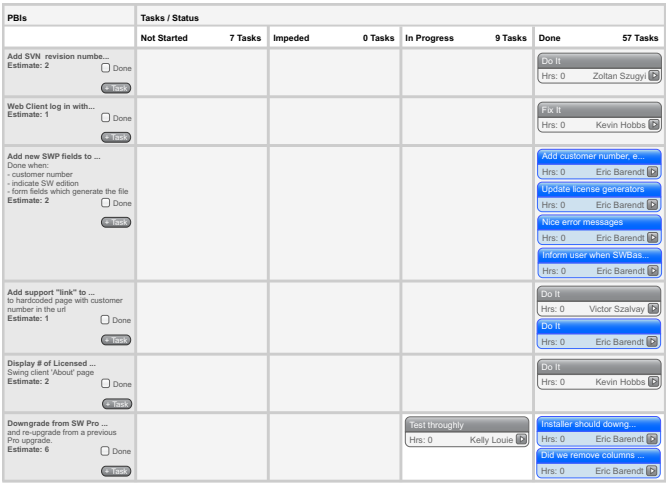

Figure 8: The Sprint Backlog may also be represented electronically in a collaboration tool such as ScrumWorks® Pro. This tool's electronic task board mimics the cards of a physical task board.

Sprint Task

- Specifies "how" to achieve the PBIs' "what"
- About one day or less of work
- Remainig effort re-estimated daily, typically in hours
- Task point person volunteers to see that it gets done, but commitment is owned by entire team and collaboration is expected

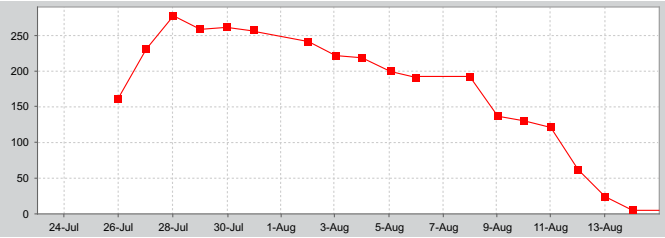

Figure 9: Sprint Burndown Chart

Sprint Burndown Chart

- Total remaining team task hours within one Sprint
- Re-estimated daily, thus may go up before going down
- Intended to facilitate team self-organization, not a management report
- Fancy variations, such as itemizing by point person, tend to reduce effectiveness at encouraging collaboration

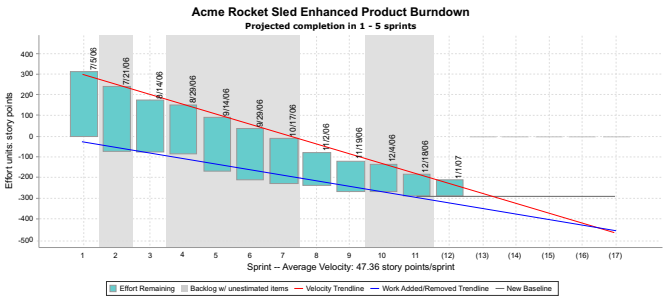

Figure 10: A Release Burndown Chart variation popularized by Mike Cohn[4]. The red line tracks PBIs completed over time (velocity), while the blue line tracks new PBIs added (new scope discovery). The intersection projects release completion date from empirical data[5].

Product/Release Burndown Chart

- Tracks remaining Product Backlog effort from one Sprint to the next
- May use relative units such as "Story Points" for Y axis
- Depicts empirical trends, introducing reality check to Product Owner's release plan

SCALING

Scrum addresses uncertain requirements and technology risks by grouping people from multiple disciplines into one team, ideally in one team room, to maximize communication bandwidth, visibility, and trust.

When requirements are uncertain and technology risks are high, adding too many people makes things worse.

Traditional practices of grouping people by specialty or architectural component can also make things worse. Typical problems include messy team interdependencies, late discovery of integration risks (being "90% done 50% of the time"), and poor alignment of effort with business value.

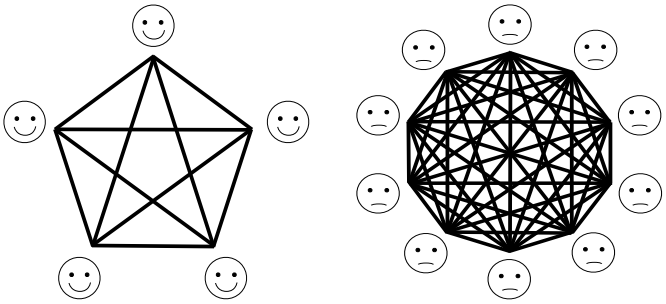

Figure 11: Communicaton pathways increase geometrically with team size.

A more successful approach has been fully cross-functional "feature teams," able to operate at all layers of the architecture, and across components if necessary[6].

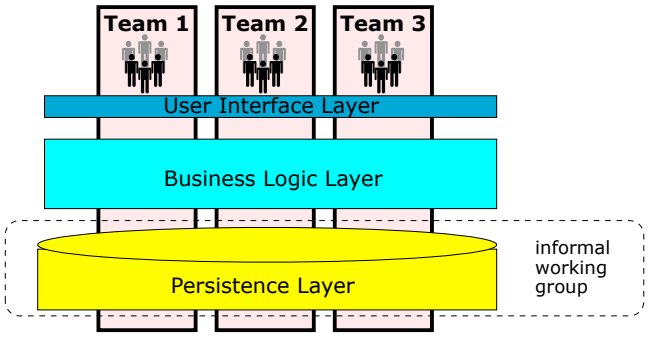

Figure 12: Cross-functional teams organized around related features.

Rather than get all aspects of a component working first, a feature team focuses on thin user-centric slices of functionality that cut through multiple architectural layers or physical components.

Since multiple feature teams risk stepping on each other's work, it's wise to get practices of continuous integration (with robust test coverage enforced through a product-wide definition of "done") established by one feature team before adding other teams.

Multiple teams coordinate with each other in a variety of ways, including sending delegates to each other's meetings or to central "Scrum of Scrums" meetings. Individuals working on particular components may form informal working groups with their counterparts on other feature teams. They are primarily responsible to their feature teams, however.

Organizations seeking to scale Scrum are advised to pursue training, coaching, and to examine previous case studies.

RELATED PRACTICES

Scrum is a general management framework coinciding with the agile movement in software development, which is partly inspired by Lean manufacturing approaches such as the Toyota Production System. Scrum has been popularized by people like Ken Schwaber, organizations like the Scrum Alliance, and companies like Danube Technologies, Inc.

Unlike eXtreme Programming (XP), Scrum does not prescribe specific engineering practices.

Scrum focuses on incrementally improving the definition of "done" (particularly around testing) before work is demonstrated. This can motivate the team to adopt engineering practices associated with XP and now proven to reduce technical debt: Continuous Integration (continuous automated testing), Test Driven Development (TDD), constant refactoring, pair programming, frequent check-ins, etc.

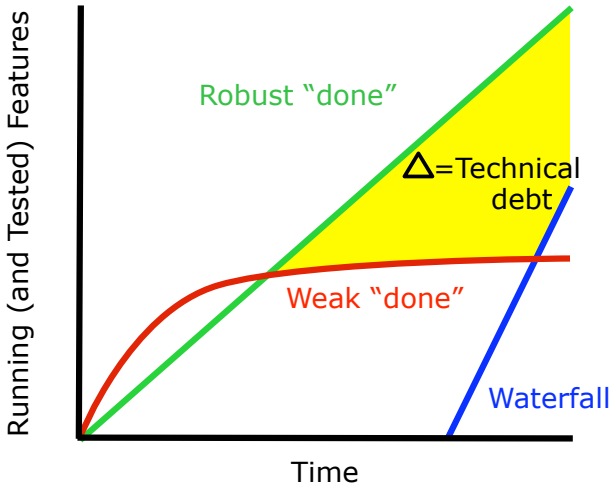

Figure 13: The green line represents the general goal of agile methods. Doing Scrum properly entails incrementally improving the definition of "done" to prevent technical debt. [7]

WHEN IS SCRUM APPROPRIATE?

Defined Processes vs. an Empirical Framework

Teams applying Scrum rigorously (as intended by this author) may find themselves questioning traditional "best practices."

The expectation of a working product demonstrated early and often, combined with frequent retrospection, leads teams to challenge assumptions from respected sources such as the Project Management Institute's Project Management Body of Knowledge (PMBOK®), existing waterfall habits, and more prescriptive processes for iterative development such as IBM's Rational Unified Process (RUP). Scrum teams may even question agile practices such as eXtreme Programming (XP).

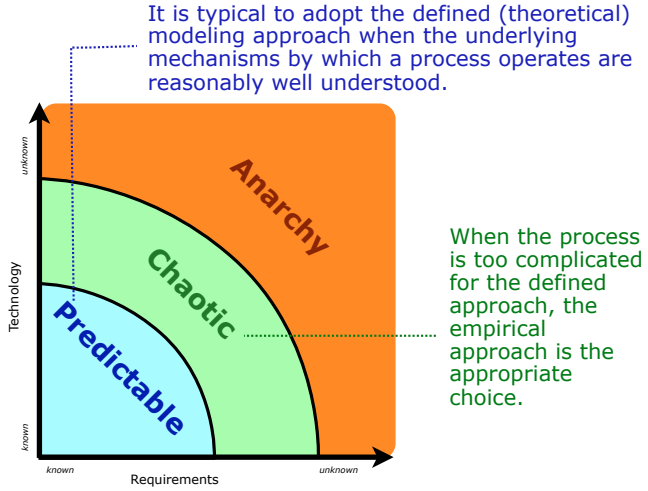

Figure 14: Scrum is intended for the green space labeled as "Chaotic" above. [8 9]

The disruption of introducing Scrum is not always advisable when defined processes could meet the needs. Ken Schwaber has said, "If waterfall suits current needs, continue using it." Consider whether the underlying mechanisms are well understood. Scrum was not originally intended for repeatable types of production and services.

Scrum is intended for the kinds of work defined processes have often failed to manage: uncertain requirements combined with unpredictable technology implementation risks. These conditions usually exist during new product development.

Challenges and Opportunities of Team Self-Organization

This author has seen self-organizing teams radically outperform larger, more experienced, traditionally managed teams. Family-sized groups naturally self-organize when they are committed to clear, short-term goals, all members can gauge the group's progress, and all members can observe each other's contribution.

Psychologist Bruce Tuckman describes stages of group development as "forming, storming, norming, performing."[10] Optimal self-organization takes time, introducing a reasonable risk the team will perform worse during early iterations than it might have as a traditionally managed working group.

Research suggests heterogeneous teams outperform homogeneous teams when doing complex work, and they experience more conflict[11]. Disagreements are to be expected on a motivated team -- team performance will be determined by how well the team handles these conflicts.

"Bad apple theory" suggests a single negative individual ("withholding effort from the group, expressing negative affect, or violating important interpersonal norms"[12]) can disproportionately reduce performance of an entire group. Such individuals are rare, but their impact is magnified by a team's reluctance to remove them. This can be partly mitigated by giving teams greater influence over who joins them.

Other individuals who underperform in a boss/worker situation (due to being under-challenged or micromanaged) will shine on a Scrum team.

Self-organization is hampered by conditions such as geographic distribution, boss/worker dynamics, part-time team members, and interruptions unrelated to sprint goals. Most teams will benefit from a full-time ScrumMaster, whose responsibilities include helping mitigate these kinds of impediments.

SCRUM EDUCATION AND CERTIFICATES

Scrum is mainly an oral tradition conveyed through Certified ScrumMaster (CSM) courses. These are typically two-day events led by trainers who have been vetted by the Scrum Alliance[13]. The CSM credential does not prove proficiency. It is intended as a stepping stone toward Certified Scrum Practitioner (CSP), an indication of at least one year of experience doing Scrum. The Scrum Alliance also certifies Scrum Product Owners, coaches, and trainers.

REFERENCES

[1] Agile Project Management with Scrum, Schwaber, Microsoft Press, 2004. See Appendix A "Rules."

[2] http://danube.com/blog/michaeljames/a_scrummasters_checklist

[3] Agile Retrospectives: Making Good Teams Great, Derby/Larsen, Pragmatic Bookshelf, 2006

[4] Agile Estimating and Planning, Cohn, Prentice Hall PTR, 2005.

[5] Example generated by ScrumWorks® Basic, a free tool.

[6] Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum, Larman/Vodde, Addison-Wesley Professional, 2008.

[7] Graph inspired by Ron Jeffries lecture at http://www.infoq.com/news/2006/12/jeffries-running-tested-features

[8] Extensively modified version of a graph in Strategic Management and Organizational Dynamics, Stacey, 1993, referenced in Agile Software Development with Scrum, Schwaber/Beedle, 2001.

[9] Quote is from Process Dynamics, Modeling, and Control, Ogunnaike, Oxford University Press, 1992.

[10] "Developmental Sequence in Small Groups." Psychological Bulletin, 63 (6): 384-99 Tuckman, referenced by Schwaber.

[11] Group Genius: The Creative Power of Collaboration, Sawyer, Basic Books, 2007.

[12] "How, when, and why bad apples spoil the barrel: Negative group members and dysfunctional groups." Research in Organizational Behavior, Volume 27, 181–230, Felps/Mitchell/Byington, 2006.

[13] Scrum training/coaching also available from http://danube.com/

ABOUT THE AUTHOR

Michael James, Danube Technologies, Inc.

Michael is a software process mentor, team coach, and Scrum trainer with a focus on the engineering practices (TDD, refactoring, continuous integration, pair programming) that allow agile project management practices. He is also a recovering "software architect" who still loves good design. Many other articles from Michael can be found at http://danube.com/blog/michaeljames

Danube Technologies, Inc. is a Scrum consulting and tools company. Danube hosts hundreds of Certified ScrumMaster courses yearly around the world. To find out more about ScrumMaster Certification see http://danube.com/courses. Danube produces ScrumWorks Pro, the first commercial tool designed specifically for Scrum. With more than 105,000 active users worldwide, ScrumWorks is used by more than half of the Fortune 100 and boasts the largest market share of any agile management tool. Learn more at http://danube.com/scrumworks

RECOMMENDED CLASS

danube SCRUM CORE™ TRAINING

Danube's ScrumMaster Certification Course

The best way to get started with Scrum is through hands-on experience in a Danube ScrumCORE CSM course. Taught by Michael James and other experts with deep experience working in Scrum environments, Danube's two-day course covers the fundamental principles, mechanics, and values of Scrum through a combination of simulated sprint activities and examples of real projects.

For more information, visit:
http://www.danube.com/courses

DZone

DZone communities deliver over 4 million pages each month to more than 2 million software developers, architects and decision makers. DZone offers something for everyone, including news, tutorials, cheatsheets, blogs, feature articles, source code and more. **"DZone is a developer's dream,"** says PC Magazine.

DZone, Inc.
1251 NW Maynard
Cary, NC 27513

888.678.0399
919.678.0300

Refcardz Feedback Welcome
refcardz@dzone.com

Sponsorship Opportunities
sales@dzone.com

ISBN-13: 978-1-934238-53-0
ISBN-10: 1-934238-53-8
50795
9 781934 238530
$7.95