



NOKIA TEST

"aka the ScrumButt Test"

Updated 25 March 2009

With help from Google, Yahoo, Microsoft, IBM, Oracle, MySpace, Adobe, GE, Siemens, Disney Animation, BellSouth, Nortel, GSI Commerce, Ulticom, Palm, St. Jude Medical, DigiChart, RosettaStone, Healthwise, Sony/Ericson, Accenture, Trifork, Systematic Software Engineering, Exigen Services, SirsiDynix, Softhouse, Philips, Barclays Global Investors, Constant Contact, Wellogic, Inova Solutions, Medco, Saxo Bank, Xebia, Insight.com, SolutionsIQ, Crisp, Johns Hopkins Applied Physics Laboratory, Unitarian Universalist Association, Motley Fool, Planon, FinnTech, OpenView Venture Partners, Jyske Bank, BEC, Camp Scrum, DotWay AB, Ultimate Software, Scrum Training Institute, AtTask, Intronis, Version One, OpenView Labs, Central Desktop, Open-E, Zmags, eEye, Reality Digital, DST, Booz Allen Hamilton, Scrum Alliance, Fortis, DIPS, Program UtVikling



Avoiding ScrumButt - Nokia Test Origins

Nokia Siemens Networks

- **In 2005, Bas Vodde started training and coaching teams at Nokia Networks in Finland. The first Nokia test focused on Agile practices**

- jeffsutherland.com/scrum/basvodde2006_nokia_agile.pdf

- **By 2007, Siemens joined Nokia Networks to form Nokia Siemens Networks with over 60,000 employees and 15 billion Euro in revenue. Bas Vodde moved to China to train Nokia Siemens Networks staff on Scrum and updated the Nokia Test to include Scrum practices.**

- **In 2007, Jeff Sutherland tuned the Nokia Test for Scrum Certification and in 2008 developed a scoring system**

- agileconsortium.blogspot.com/2007/12/nokia-test.html

- jeffsutherland.com/scrum/Agile2008MoneyforNothing.pdf

- **Each person on the team takes a sheet of paper and prepares to score questions on a scale of 1-10.**

Question 1 - Iterations

- **No iterations - 0**
- **Iterations > 6 weeks - 1**
- **Variable length < 6 weeks - 2**
- **Fixed iteration length 6 weeks - 3**
- **Fixed iteration length 5 weeks - 4**
- **Fixed iteration 4 weeks or less - 10**

Question 2 - Testing within the Sprint

- **No dedicated QA - 0**
- **Unit tested - 1**
- **Feature tested - 5**
- **Features tested as soon as completed - 7**
- **Software passes acceptance testing - 8**
- **Software is deployed - 10**

Question 3 - Agile Specification

- 🎯 **No requirements - 0**
- 🎯 **Big requirements documents - 1**
- 🎯 **Poor user stories - 4**
- 🎯 **Good requirements - 5**
- 🎯 **Good user stories - 7**
- 🎯 **Just enough, just in time specifications - 8**
- 🎯 **Good user stories tied to specifications as needed - 10**

Question 4 - Product Owner

- **No Product Owner - 0**
- **Product Owner who doesn't understand Scrum - 1**
- **Product Owner who disrupts team - 2**
- **Product Owner not involved with team - 2**
- **Product owner with clear product backlog estimated by team before Sprint Planning meeting (READY) - 5**
- **Product owner with release roadmap with dates based on team velocity - 8**
- **Product owner who motivates team - 10**

Question 5 - Product Backlog

- **No Product Backlog - 0**
- **Multiple Product Backlogs - 1**
- **Single Product Backlog - 3**
- **Product Backlog clearly specified and prioritized by ROI before Sprint Planning (READY) - 5**
- **Product Owner has release burndown with release date based on velocity - 7**
- **Product Owner can measure ROI based on real revenue, cost per story point, or other metrics - 10**

Question 6 - Estimates

- 🎱 **Product Backlog not estimated - 0**
- 🎱 **Estimates not produced by team - 1**
- 🎱 **Estimates not produced by planning poker - 5**
- 🎱 **Estimates produced by planning poker by team - 8**
- 🎱 **Estimate error < 10% - 10**

Question 7 - Sprint Burndown Chart

- **No burndown chart - 0**
- **Burndown chart not updated by team - 1**
- **Burndown chart in hours/days not accounting for work in progress (partial tasks burn down) - 2**
- **Burndown chart only burns down when task is done (TrackDone pattern) - 4**
- **Burndown only burns down when story is done - 5**
- **Add 3 points if team knows velocity**
- **Add two point if Product Owner release plan based on known velocity**

Track Done - Scrum pattern by Jim Coplien

... you have a burn-down chart that you are using to track remaining work. The burn-down chart is a visible picture of the project state, and serves as a team motivator and sanity check.

It is easy to interpret the burn-down chart as a good portrayal of estimated remaining time, and to use that portrayal to develop confidence in meeting the Sprint's actual business goals of *done* functionality.

Usually, team members update the burn-down chart daily to reflect adjustments to the amount of remaining work. Such updates reflect a desire to have as good knowledge as is possible about the effort remaining. These estimates are made in mid-stream and reflect increases that arise from emergent requirements. However, given that one emergent requirement has been discovered in a task doesn't imply that no others remain. While the confidence in an estimate *usually* improves with each revision and with continued work on the task, unusually wicked problems seem never to converge.

On the other hand the Product Owner is not centrally interested in partially completed work, only in items that are *done* and potentially shippable. Since the goal of Scrum is to achieve the Sprint target agreed with the Product Owner, and to reduce risk, the focus should be on *done*. Emergent requirements increase risk, and the Product Owner is certainly interested if estimates expand. Because there may always be emergent requirements, any estimate of remaining time based on work mid-stream in a task has a higher degree of uncertainty than the relatively risk-free estimate of zero remaining time for *done* items.

In theory, it is possible for the remaining time on a burn-down chart to be quite near zero, yet to have few (or perhaps zero!) tasks in the *done* state.

Therefore:

Update the Product Backlog in only two cases: *reducing* the amount of remaining known work if the task is *done*; and *increasing* the amount of known work if the task grows in size due to emergent requirements or other insights gained during the Sprint. Do not reduce the amount of remaining work that arises from progress on partially completed tasks.

* * *

The team and Product Owner have a better picture of the state of the Sprint with respect to the Sprint's business goals of delivering *done* functionality. The team can revise estimations in the middle of a Sprint with more confidence because they are not dependent on the unknown remaining time for partially completed tasks. Yet, the risks incurred by the "surprises" of emergent requirements are embraced and made visible.

It is impossible, using this approach, to come near the end of a Sprint with a burn-down chart that projects success even if the Sprint only ends with 90% of the tasks 90% done.

* * *

There is a chance that a completed task can become "un-completed" by emerging requirements in some other task during the sprint. For such cases, see the pattern Domino Effect.

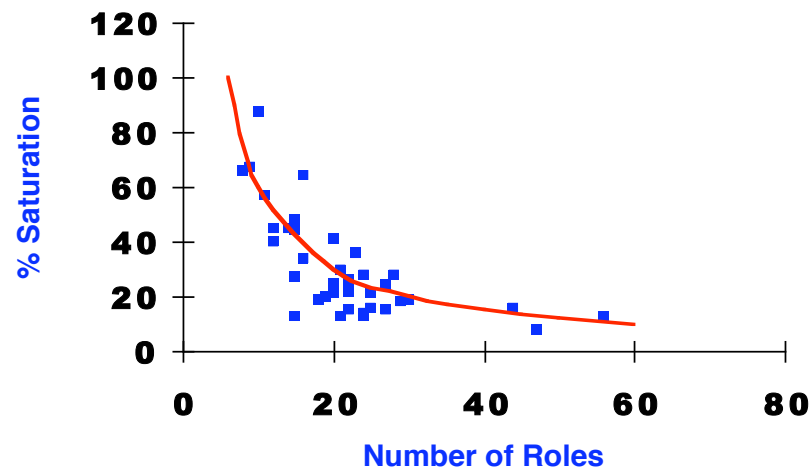
This pattern was suggested by Jeff Sutherland, co-founder of Scrum, and he reports that it is widely used by his clients.

James O. Coplien

CSM v9.6 © Jeff Sutherland 1993-2009

Question 8 - Team Disruption

- ❶ **Manager or Project Leader disrupts team - 0**
- ❷ **Product Owner disrupts team - 1**
- ❸ **Managers, Project Leaders or Team leaders telling people what to do - 3**
- ❹ **Have Project Leader and Scrum roles - 5**
- ❺ **No one disrupting team, only Scrum roles - 10**



Organizational Patterns of Agile Software Development by Coplien and Harrison (2004)

CSM v9.6 © Jeff Sutherland 1993-2009

Question 9 - Team

- **Tasks assigned to individuals during Sprint Planning – 0**
- **Team members do not have any overlap in their area of expertise – 0**
- **No emergent leadership - one or more team members designated as a directive authority -1**
- **Team does not have the necessary competency - 2**
- **Team commits collectively to Sprint goal and backlog - 7**
- **Team members collectively fight impediments during the sprint - 9**
- **Team is in hyperproductive state - 10**

Typical Nokia Test Scores

- **CSM classes start out at average score of 4.0**
- **By end of class, individuals think they can raise their teams to 6.0 by the end of one month**
- **Conservatively this will raise velocity by 20%.**
- **One month for one team costs about 100000 Euro. Cost reduction of 20%. Earlier time to market should generate revenue multiplier.**
- **Minimum return first year is 220000 and cost of Scrum Certification is less than 2000 Euro.**
- **ROI > 11000% first year**

Results in many OpenView portfolio companies

- **Score begins at 4.0**
- **Within six months score is over 7.0**
- **Velocity is 300% of initial velocity**

- **Product Owner now has too much work to do and Product Owner team must be formed.**
- **Company reaches parity with competition and begins to focus on product vision that goes beyond current market.**
- **Assisted by strong management involvement all the way to Board level.**



CSM v9.6 © Jeff Sutherland 1993-2009

Questions?



Emergent architecture



CSM v9.6 © Jeff Sutherland 1993-2009

