

CSM Practicing Certification Renewal Assessment

Name: Jim Schiel

email: james.schiel@siemens.com

date: June 13, 2005

Scrum depends on the “inspect and adapt” mechanisms of process control to manage the complexity of projects. For inspection to work, everyone must know what is being made visible. To implement the Scrum process, such regulating mechanisms as defined roles, involvement versus commitment, time-boxes, and regular cycles are used.

1. Describe one project on which you have used Scrum over the last twelve months.

The project I worked on consisted of multiple semi-independent middleware components developed over a period of 6 months that provided common functionality to a number of products built upon a similar architecture.

The project was budgeted at US\$2.5million and was completed with an actual cost of US\$1.75million.

The value add of these components is to improve the interoperability of the various products by giving them a common approach to various ubiquitous features. It is, as yet, too early to determine the actual derived value while wait for the participating applications to fully adopt the componentized functionality.

The project team consisted of fourteen employees – twelve developers, a technical writer, and an education/training writer. For the duration of the project, the team was split into two teams varying in size between 7 and 8 members, where the writers were shared between the teams (often, however, the writers alternated between the teams). During the latter sprints, some of the team was moved to another project and the team reorganized itself to a single Scrum team of ten members. While the teams were always cross functional, the physical arrangement of the teams changed from cubicles that were reasonably close together (on the same floor, but in separate hallways) to newly-built team rooms that brought the teams into a shared location without separating walls.

The training of the project team and initiation of the project was not a factor in this project, as the team had been previously trained in the use of the Scrum (in July '04) and the customers and stakeholders of the components had been consulted to create a prioritized product backlog prior to the inception of the project.

Progress reporting was managed through reports generated in our project management tool to show the higher level sprint-to-sprint planning and through the product and sprint burndowns maintained by the Scrum teams. The project management reports provided longer-range reporting that satisfied more conventional management views of command-and-control while the burndowns provided up-to-the-minute visibility of each team and overall progress through the release backlog. Customers were given direct access to the burndowns, which were posted to a web site each day. By providing the URL to our customers and managers, anyone could look at any time to see the condition of the project.

Scrum immediately caused the impact of changing priorities of our customers to be surfaced. By this, I mean that we, before Scrum, had often completed functionality for our customers just to find that their priorities or their timeframes had changed and the needed functionality was no longer needed at that time or, in rare instances, no longer needed at all. When this project began, the team made a commitment to begin no work that did not have an engaged and committed customer ready to try out the new software at the end of the sprint – if they weren't interested enough to at least invest 5 hours each month (4 for the planning meeting and 1 or less for the review), they weren't involved enough to ensure the team built the software properly. In these cases, the features were left on the Product Backlog, but reprioritized to ensure that we'd look at them again at a later date.

The Scrum Master for this project is also a resource manager. The project team, for the most part, was direct reports of the manager/Scrum Master. There was some initial concern that this might cause a problem, but the team and manager agreed on simple ground rules to mitigate problems. The Product Owner for the project was the component teams' Product Manager. Being responsible for the content and success of the components, we felt that the Product Manager would make the best Product Owner. The team also took advantage of opportunities to work direct with developers in other development groups that would be using their software. The only difficult we encountered with the Product Owner was their lack of experience setting priorities on a constant basis – previous projects were run based on whatever the Product Manager thought was important at the time. The Product Backlog, however, ensured that the Product Owner was consistent in their approach to prioritization and that they were constantly checking with customers to ensure that the prioritization was correct. This effort often required the assistance of analysts and other managers to ensure that the Product Backlog was properly maintained in time for each sprint-planning meeting.

From a software development perspective, the difficult problem to overcome (it took nearly three sprints to get it right) was how to get developers that were primarily software validation employees to work day-to-day with the rest of the team to develop software and tests at the same time. Previously, testing was something that was done when most of the software features were completed. However, in order to produce “done” software each sprint, testing had to be done earlier, faster, and in smaller increments. This was a deep-cultural change for the group.

At the end of the project, we planned a single sprint for stabilization. The structure of the stabilization sprint was fairly basic – we started with several runs of our automated unit tests on the different platform variations on which the software is supposed to be able to run. Then, we added application level tests, running more complex drivers that used the components like a “real” application would, but at much greater volumes (e.g., 5,000 users instead of 500). As it turned out, the majority of the testing was completed within two weeks. So, the team opted to improve the test suite even further, creating more and more boundary condition checks. Still, the sprint was able to end almost a week early and software was delivered to the internal users by the original completion date of the sprint.

In the end, the project was very successful. The team finished the release backlog on time. All features were testable with automated unit tests. We delivered early versions and final versions of features to our internal customers and we were able to easily support their take-on due to the high quality of the code. We were also able to save about 4,000 hours of work from this project (and some from the next project which would have included certain prioritized features) when the architect, in the first few days of a sprint, determined that we were approaching our solution in the wrong manner. We cancelled the sprint, replanned, and were successful. We cite the Scrum process as being the main driver of our success. While the team was already very cohesive, Scrum provided a simple mechanism that allowed everyone to be on the same page with everyone else all the time. The team began controlling more and more of their own destiny and the software quality and the team's performance improved with every sprint. In the end, even the team's Scrum Master was hard pressed to provide much more value than updating the backlog and facilitating the Scrum meetings.

As for the Scrum process itself, our rule was to follow the Schwaber/Beedle book to the letter, deviating only when there was an extreme need. Our assumption was that the process documented in the book was the result of several years and many, many projects. We felt we were hardly in a position to question that, even in our second Scrum project. In the end, we used the process without any significant deviation.

2. How do you cause the accuracy of Product Backlog estimates to improve? To what degree does their accuracy matter?

Accurate Product Backlog estimates are extremely important as they help mitigate the variability

inherent in software projects. In other words, a lot of hour-adding surprises occur in projects because they were not accounted for in the original work effort estimate (that is to say, you usually don't get in trouble for telling someone that a 4,000 hour feature will take 4,000 hours. It's when you tell them it'll take 4,000 hours after you told them 2,000 hours that the problems begin). When estimates on the Product Backlog improve, important market decisions can be made that can improve your profit margin immensely. Product Backlog estimates can be improved through a few different ways:

- Use completed features as a gauge to estimate similar efforts in a similar manner. Learn from the actuals to improve the estimates.
- Use user stories and high level use cases to improve your ability to "see into the future" without spending an inordinate amount of time on your effort estimation.

3. How do you cause the accuracy of what a team commits to for a Sprint to what the team actually delivers?

Scrum Masters have the greatest impact on what a Scrum team commits to in the Sprint planning meeting. It is here that the Scrum Master helps devise the Sprint Goal and helps the team determine what fits and what doesn't. By keeping focused, the Scrum Master can help the team understand what worked in the past and what didn't, and can keep the team on track during the planning meeting. During the course of the Sprint itself, the Scrum Master needs to maintain the backlog, ensuring that every backlog item is picked up by someone on the team and ensuring that impediments to the team's progress are cleared efficiently and rapidly. The Scrum Master should also schedule retrospectives for each Sprint, and should ensure that the learnings are brought forward into all of the successive Sprints.

4. What metrics do you use to track the development process? Which metrics have been changed, removed, or newly implemented as a result of using Scrum?

We primarily track the development process by using a linear trend line superimposed on the Sprint burndown graph. Also, when the team make-up isn't changed too much, we use velocity (the number of hours of effort removed from the Product Backlog each month) as an indicator of performance. We have also created two new projections on the Sprint Burndown – one is an ideal burndown (what if the entire team were dedicated 100% to the project – instead of taking time out to account for emails, meetings, vacation/sick time, etc.). The second is the anticipated burndown that shows how the sprint would burndown if everything went as planned and everyone was able to put in exactly the number of hours to the sprint that they committed at the sprint-planning meeting. These new "projected" burndowns give us an idea of how the team is performing relative to the hours they could commit. These metrics are never used to punish a team, however. These metrics are similar to the meter on a car's dashboard that shows RPMs – these metrics give us an idea of how the team is performing, and gives us a way to visualize potential problems and help the team solve them before they become serious.

5. What type of training, resources, or tools would best help you successfully employ Scrum in the future?

Our training situation is good, but we're having a very difficult time finding a tool that can manage an enterprise level product backlog (40,000+ items) with various inter-dependencies that also supports the creation of sprint backlogs and the appropriate burndown graphs.