

# CSM Practicing Certification Renewal Assessment

Name: Kiran Thakkar email: kirankumar.thakkar@siemens.com

Date: January 18, 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 12 months that provided common functionality to a number of products built upon a similar architecture.

My project was planned to deliver re-usable internal components for use by my company. That would in turn support the GA (General Availability) of broad range of solutions. Components would be built, bought, or acquired, based on individual component criteria. These components will:

- **Satisfy application specific requirements** – by providing capabilities required by multiple applications (e.g. User authentication)
- **Satisfy solution requirements** – By improving cross-application workflow for the end user (e.g., Through shared sign on capabilities)
- **Improve margin and customer ROI** – By lowering Total Cost of Operations for the customer and our datacenter (e.g., through single points of maintenance).
- **Reduce development costs** - (e.g., by avoiding the cost of redundant development)
- **Produce direct revenue** - Components pass through a life cycle in a similar fashion as products, as components become fully adopted and deployed in the field, they have the potential to increase revenue and sales by encouraging value-added services and optional sales

Our common output management system accepts, creates, and conditionally routes our key product's output to a customer's desired destinations. Destinations include printers, e-mail addresses, FAX, files, image archive, and other small devices. This system provides utilities to easily setup, manage, and audit routing criteria, destinations, and document types. It provides services and tools to allow applications to control and access output-based data so applications can seamlessly integrate output management functions into their specific workflows. This system offers many value-added benefits, including:

- Multiple destinations types
- Document adaptation using Microsoft Word
- Server-based document generation/rendering
- Temporary artifact storage
- Automatic routing criteria
- Browser-based UI for configuration & management

The project was budgeted at US\$1.1 million and was completed with an actual cost of US\$720,000/-.

**Value:**

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.

**Team Structure:**

The project team consisted of 10 employees - 7 developers, one technical writer, one product analyst, and an education/training writer. For the duration of the project, the team functioned as one scrum team, where the writers were shared between the teams (often, however, the writers alternated between the teams). 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.

**Training:**

Jim Schiel provided in house scrum training in Nov-Dec-2004 timeframe. The customers and stakeholders of the components had been consulted to create a prioritized product backlog prior to the inception of the project. Later in early 2005 I received Scrum Master Certification training from Ken Schwaber.

**Agile Project Management:**

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 burn downs 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 burn downs provided up-to-the-minute visibility of each team and overall progress through the release backlog. Customers were given direct access to the burn downs, 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.

**Findings:**

Scrum immediately provided the root cause and impact of changing priorities of our customers. In other words, before Scrum, the team worked (invested time and money) on unnecessary functionality and did not realize until the end of the project (usually 6 to 9 months). When we started using Scrum, 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 fact, I encouraged and engaged our customer more than just sprint planning and review meeting. 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.

I was the Scrum Master for this project; so the Scrum Master for this project is also a project manager. The project team, for the most part, was direct reports of the resource manager. There was some initial concern that weak matrix structure might cause a problem, however the team, resource manager, and scrum master agreed on simple ground rules to mitigate problems. The Product Owner is the Product Manager. However, the Product Manager was not experienced enough on setting constant priorities from sprint to sprint. 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 two 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.

### **Testing:**

For this project, team completed integration testing for Sprint 1 during Sprint 2. The team completed unit testing within the sprint. This was very useful to the team and we avoided tester's ideal time in initial period where tester could not be involved as a developer (breaking silos). The disadvantage was that defects found during the Sprint 2 integration testing took away development time from developers.

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.

### **Success:**

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 3,500 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, re-planned, 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.

### **Material:**

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?**

At the start of the project, team provided with very inaccurate estimates. Apparently, the developers had never been trained on estimating tasks. They usually gave wildly optimistic time frames and understated the effort required by a factor of 2 or more. This made estimating what could be developed in a Sprint pretty hard. I noticed that they usually did not account for all the things that could go wrong and that they also did not really understand what the requirement actually meant from a coding perspective. I also noticed that team/project did not use any estimation technique.

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:

1. Use completed features as a gauge to estimate similar efforts in a similar manner. Learn from the actuals to improve the estimates.
2. 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 did not, 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 someone picks up every backlog item on the team and ensuring that impediments to the team's progress are cleared efficiently and rapidly. After two sprints; I created a weekly checkpoint with the team and product owner on how we are doing on accomplishing sprint goal and review open issues and risk items. 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. In addition to Sprint burndown, we created the release baseline in Primavera and high-level sprint schedule and resources estimates. Primavera tool provided release level view on overall timeline, resource allocation, and monthly-earned value report. 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 Burn down, 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.

This 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. In addition to product backlog; we are struggling to manage release backlog that can align with large number of sprint backlogs (currently 30 sprints at the same in clinical) in a global distributed development environment.