

# CSM Practicing Certification Renewal Assessment

Name: Alex Pukinskis      email: alexpukinskis@gmail.com    date: 22-Feb-2006  
Certification: Jean Tabaka, Denver, CO November, 2005

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

I've been using Scrum as a consultant and ScrumMaster for several years, and currently work as a coach for Scrum teams and ScrumMasters at multiple clients. I'll focus below on one project I worked on last year.

This project involved building a web-based system for processing small business loan applications. The intended duration of the project was 5 months, although the team has continued on additional Scrum projects since. The budgeted cost was around \$350k and the actual cost was about \$450k. In an organization used to 200% overruns, this was a major victory. The benefits were significant; the organization would be able to receive unsolicited loan applications online, so that when a small business owner need a loan, s/he could simply visit the website and apply, drastically reducing costs to close the loan. Before we did this project, there was support for a two-part process that required a follow-up call to gather more information. Also, this project would enable us to take advantage of realtime credit decisioning, which increased the likelihood of completion of the loans.

Our team size was a little large at 12 people, but some of these people were specialists who left to work on other projects after a few iterations. The team included developers, designers (lead developers), data services developers and architects, the product owner, myself as a ScrumMaster, and one overworked tester. Because we only had one tester, developers got pretty good at getting features reviewed by the product owner to reduce the work the tester had to do. We tried to be cross-functional, but the organization was highly specialized and developers looked to the lead developers for a lot of guidance. We found that on some areas, especially UI development, developers were able to work directly with the product owner. Because the key lead was assigned to 5 projects, he was rarely around in the afternoons, and so the other developers gradually learned to take more initiative.

This was an unusual project for me because it had been started as a Waterfall project and had been completed through feasibility, analysis, and detailed design. However, changing business circumstances had invalidated the design and requirements – we had to implement the system on a different platform.

I introduced Scrum and Agile to the team over 2 days of workshops, and then we worked together to build a backlog. Since the Product Owner and technical lead had already been through the requirements once, it was relatively easy to build a backlog. We estimated the size of the features, and then planned out our first sprint. We had an unusually good environment – a 30x30 room with whiteboards on all of the walls, movable desks, and so on, which really helped us; we were able to track our sprint and product backlogs on the walls where everyone could see them. (I have some experience with XP, so I drew from that and we used index cards to track features and tasks). During standups, I wrote obstacles that were raised up on the wall; in the beginning, we had quite a few, but everyone could see where things stood.

Because the organization was matrixed, people had other responsibilities that drew them away from the project sometimes. To deal with this, we set core hours from 9-12AM every weekday. Our daily Scrum happened at 9AM, and everyone stayed in the room at least until noon to work on the project. Many people arrived at 8AM, and about half the team was still in the room in the afternoon. At first, I was really disappointed that we couldn't all be there full-time, but at one point the Product Owner told me he was really able to sleep well now that he knew everyone on the project would be working on his project every morning.

Sometimes stakeholders would attend the daily Scrum meetings to see how progress was going. (I think I really offended one stakeholder when I told the pigs and chickens story – timing is everything!) We also updated both Sprint and Release burn-down charts (we were working in 2-week sprints). I sent weekly status emails to stakeholders as well, and they attended our Sprint Demo and Review meetings.

There were a lot of challenges on this project. The matrixed team, partial allocations, and architectural constraints all got in our way. Also, several of the developers were “offshore” in Canada. We were able to mitigate this issue by bringing them down for the first 6 weeks of the project, and leaving a speakerphone open between the two locations during the mornings. The Scrum process made it painfully clear that the offshoring wasn’t saving any money, and this was politically difficult in the organization. In the end, we continued with the offshored arrangement, and we still got a lot done, but things were much slower than when we were collocated. We had significant delays that are typical in large organizations – access rights, database changes, hardware, and that sort of thing. Our tester’s computer and desk were 5 miles away in another building, but I was able to borrow a laptop from the methodology group for her.

I had been a ScrumMaster and Agile Coach for several years, so that role wasn’t a change for me. We discovered that the person assigned as Project Manager was really a Product Owner in disguise, and he took well to his new role. (He had been trying to transition from being a Project Manager to an Analyst anyway). Because this was a public-facing internet site, we couldn’t have used an actual customer, so he was really the best person to be our Product Owner. He took to sitting with his laptop facing the door of our room so that if anyone tried to come in and steal one of the team members for another project, he could intercept them first; between the two of us, we did a pretty good job of shielding the team from outside meddling, at least in the mornings.

The technical environment made it difficult to be Agile. The underlying system had been built using Waterfall over many years, so it was quite brittle, and the design had not been updated. We did our best to deal with this, but in that organization, the technical debt was a political problem, so we worked around it. We were very careful about making sure we didn’t break any existing functionality as we moved forward. Toward the end of the project (after I moved on to work with other clients) the team size reduced; the last sprint was spent implementing only 1-2 features and most of the effort went into testing. Also, at that point the team produced the release documentation required by the waterfall-oriented release groups.

We were pretty strict about the Scrum process. We did shorten our sprints to 2 weeks, and due to the organizational constraints, the team wasn’t allowed to employ “any means possible” to deliver on our sprints, but in an organization still waterfall-oriented, we did the best we could. It’s really risky to remove any part of Scrum because each piece is critical. With a 900-page waterfall methodology, you can skip a few steps. Scrum only has 10-15 moving parts, and if any one is missing, the project is at risk.

In the end, the project delivered on time and the customer was happy; the team was able to deliver features well beyond the scope of the original plan.

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

A lot of this happens naturally. Many developers start out by padding their estimates, or by underestimating; after a few sprints they start to converge. I’ve had a lot of success by doing product backlog estimates in terms of relative size using points, because teams are pretty good at predicting relative size. The burn-down chart tells us a lot about where we stand, even when estimates are inaccurate, but it takes a couple of sprints for people to see that.

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

Experience as a team helps; they get better over time. It's really handy to use XP's velocity measure – look at how much work we got done last sprint, and if the team is the same this Sprint, chances are we'll get the same amount done.

Last summer I was working with a Scrum team for about 10 weeks on a port of a system that runs hospitals. At the end of the Sprint Planning meeting, we had about 240 hours worth of work in our sprint backlog, but in the previous sprint we got only 160 hours done. We had run into a lot of obstacles last time, and a majority of the team seemed to think we could get more done this time. I used the "Fist of Five" consensus-measuring tool, and found a few people were strongly opposed to this commitment. We went back and forth, people were tired, the Product Owner really wanted all the work done, I thought it was feasible, and so I made the mistake of saying "Ok, we'll do it" on behalf of the team. Of course, we didn't make our commitments. Since then, I feel pretty strongly about using the fist of five and waiting for consensus to form, or urging the team towards the lower commitment to get to consensus faster. It's much easier to add scope to a Sprint than to take it away.

#### **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?**

Time remaining per task and the burn-down charts within sprints; observed velocity to do long-range planning over the backlog. My pre-agile experience was on the no-process, code-and-fix side of things, so these have been new metrics for me over the last few years. I've also got a lot of mileage out of test coverage metrics and successful build percentage.

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

I'm starting to see more interesting things going on in the area of backlog management. I used to be pretty staunchly low-tech - wall-based backlogs work wonderfully if the team is collocated in one room. It seems like most companies don't have this luxury yet. A central repository for the backlog, rather than an Excel spreadsheet, seems to be critical to moving towards real self-organization. It seems like an Excel spreadsheet forces the ScrumMaster to constantly ask everyone how much time is left; it's better for the dynamic if people can update this information themselves.

I'm particularly interested in seeing how Enterprise Architecture changes in response to the adoption of Scrum. As larger organizations get more and more scrum teams going, enterprise architects are starting to try to figure out how to maintain a sensible architecture with lots of autonomous teams. I'm hoping that within the next year good ideas in this area start to coalesce.

#### **6. (Optional) Scrum and Extreme Programming are sometimes used together. What must be considered when this is done?**

I just wrote a blog entry on this, which will be showing up in the new AgileDevelopment magazine. If Scrum and XP are being mixed, it's critical to do Scrum by the book, and then pick and choose the XP practices. XP is a highly optimized approach that requires specific things – a collocated team, an onsite customer, control over the code base, etc. If you leave out any parts of it, it tends to fall apart. The same, of course, is true of Scrum, but Scrum seems to be an easier sell for most organizations. (Maybe it's the sports metaphor). I generally see a cycle – adopt Scrum, get it right, start using user stories for your features, start doing continuous build and integration, and automated acceptance testing. Some teams pick up pairing, refactoring, TDD, and simple design over time. If you keep applying Scrum, and keep inspecting and adapting, you'll eventually get as close to XP as is feasible in your organization and culture.