

CSM Practicing Certification Renewal Assessment

Name: Christian J. Hessler

Email: iakom@yahoo.com

Date: November 12, 2004

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

Purpose - what business goal was the project intended to deliver?

The project, herein termed DSS, was a project designed to consolidate data sources within the software engineering and services organizations. Disparate source databases were organized in terms of "subject areas" in keeping with the Bill Inmon Corporate Information Factory model, and fronted with a data abstraction layer/API for the purposes of data access, provisioning and collaboration. The purpose was to provide a single, coherent interface to the wide spectrum of services data for users, applications and downstream web services. The ultimate business goal was more secure, expedient access to higher quality data across the org. The project was spearheaded by a sub-initiative, called ABP#10, a reporting application that would be the first end-user facing client of this DSS service. This segment was the focus of our Sprints.

Length - what was the duration of the project?

The project consisted of 5 Sprints, 1 for design, 3 for construction and 1 for release/stabilization. Originally, the initiative was targeted for 9 months, but by using Scrum, we effectively reduced that to 5.

Cost - what were the budgeted and actual costs?

As the Services division was funded in an indirect manner from the higher-level projects (and this was an internal component), I am not aware of our budget. However, we were allotted 10-12 head count as well as corollary resources from other partner teams (QA, release eng, production support, etc).

Value - what were the projected benefits and actual (if measured) actual benefits?

There were two levels of target project benefits and 3 measured actual benefits. The two targeted were:

- a) Increased access and use of the availability data via DSS API and ABP#10 reporting client (and thereby reduced independent access through unsupported or distributed means)
- b) Increased quality and comprehensiveness of the availability data (via DSS) across the reported customer install base

The three measurements used to validate these were the following:

- a) Availability and uptime of the DSS service
- b) Concurrent internal and end-customer users of the system/data
- c) Number of bugs and complaints against the accuracy, comprehensiveness or

timeliness of the data and reports

As this was the first project of its kind within the org, the measurements were compared against similar metrics from the source systems and their existing manual operations and access.

Size - how many people were on the project team(s), how were they organized into teams?

There were three Scrum teams consisting of 4-8 members per team. We organized along architecture interface lines: one team for the back-end data systems and API, one for the reporting engines and data marts, and one for the UI/interface. There were three Scrum Masters, one architect (myself, who acted as one of the Scrum Master's as well as the Scrum of Scrum's Master – we affectionately termed “Scrum Lord”), and some matrix team members (production support, QA, PM) who participated throughout the Sprints.

Teams – were the teams cross-functional and self-organizing?

The teams were certainly cross-functional (matrix members, crossing department lines) but the self-organizing was an iterative phenomenon. In the beginning, the Scrum Master's had to construct the teams and their initiatives based on architectural and interface lines – as opposed to the current org structure. Once that happened, the teams began to self-organize to a greater degree.

Were the teams collocated in an open space? Were the teams physically separated within one location, or located in more than one physical location?

One unique and effective aspect of our Scrum approach was the use of “war rooms” for the entire duration of the project. Three rooms were selected, collocated on the same hallway, consisting of several terminals (for demos), a large conference table, laptops (for development), wireless hubs, and a small library of reference books. Only one team member was remote – and that proved difficult to mix with the team dynamic. So difficult in fact, that this particular member soon left the team to be replaced by a local person.

Initiation - how was the project initiated? How was the team trained to use the Scrum process?

The project was initiated by a top-down affirmation of Scrum and the business goals. In advance of this, several members were certified with Ken in the Denver training courses. Others, while uncertified, had used Scrum and other Agile approaches in other projects. We assembled the team and conducted 2-day training on campus. We quickly proceeded with compiling the Master Backlog based on functional specs, held Sprint planning and began to Sprint! Unfortunately, we did not have an official “kickoff” (a lesson learned and rectified for later projects). It was a very agile way to roll out agile – as we iteratively improved the process through estimate evaluations, Sprint reviews and feedback from our stakeholders and process partners. By Sprint 3 – we were quite settled and effective in our approach.

Reporting - how did you report progress to management and the customers?

In the beginning, we used a self-made spreadsheet to track our Backlog, Sprint Backlogs, tasks and burn down. We exported this sheet daily (after Sprint meeting) as HTML and

posted on the web. We also provided weekly roll-ups of progress and burn down at the task and feature level to the stakeholders at the regular P-Team (project) meeting. Outside of this, management attended the daily Scrums as chickens, as well as the Scrum of Scrums. Of course, all attended the periodic Sprint Demos and Reviews. Our Manager was the Backlog owner who represented the stakeholders, and was also involved deeply in the Sprint planning process.

In addition, I created a Sprint review report at the end of each Sprint. I dealt with three main sections: feature and task coverage/completion and burn down (various metrics on team utilization, estimates and functional spec coverage), unit test, QA and quality/performance data on the components produced, and a prosaic section on roadblocks, lessons learned and various team and stakeholder feedback. These reports, produced for each Sprint, created a great repository and history of our Sprint experiences. The goal for these reports was to feed the future Sprints and other teams with the empirical information.

Change - what difficulties were surfaced by Scrum that had to be resolved? How were these resolved?

The primary difficulties we encountered fell into two categories:

- a) Lack of good and stable requirements, or solid governance thereof
- b) Dependencies on external teams and resources that we could not control

As with most Scrum projects (and why we chose this approach) the initial requirements were not solid. Therefore, we declared an initial interpretation of them in the Sprint planning and worked with what we had. The validation of them at each demo/review was invaluable. Nevertheless, we found it difficult to “control the chaos” without solid Backlog ownership. There was still some direct stakeholder to engineer contact. This situation was remedied in later projects by declaring more official backlog ownership and accountability, additional management Scrum training and the use of a web-based tool.

The second area of difficulty involved external dependencies. The existence of hardware, support agreements and other items external to Scrum were out of our control. In addition, our code utilized other public interfaces (authentication modules and LDAP systems, for example) that we could not control with respect to versioning or availability. While more a question of architecture, this last item presented particular difficulty. Our resolution was to stub in the assumed interface and demonstrate functionality at the review, knowing we would go back and repair the interface with the actual service in a later Sprint.

We managed these issues through our Scrum of Scrums, roadblock escalation at the daily Sprint meeting and the repurposing of tasks to isolate these risks while proceeding with valuable development.

Management - what was the previous role of the Scrum Master? Who took on the role of Product Owner? To what degree were they successful in fulfilling their roles?

The Scrum Master was (and remained) the project architect. The manager took the role of Product Owner/Backlog Owner who represented a committee of Stakeholders (the PAC). While not ideal, this process at least provided us a traceable flow from inception to

delivery. The Scrum Master was very successful, but the Backlog Owner/Product Owner was marginally successful – due to the wide variety of Stakeholders and competing priorities. We planned to mitigate this in the future with more backlog governance training and forcing a stricter requirements gathering process on the front-end.

Engineering - what software engineering practices or environment had to be changed?

The primary engineering practices and environments we modified (and matured) for Scrum was the source control, continuous integration and release processes. We consolidated code under CVS with a standard management and branching policy. We agreed on a standard code review and RTI (request to integrate) process, as well as a naming, ANT-based test automation and packaging process.

From an infrastructure perspective, we simply obtained dedicated lab hardware at the beginning of the Sprints to ensure cohesion and stability.

One addition engineering process we adjusted was the iterative approach to QA. Previously, QA was a capstone entity, creating test cases at the beginning and then testing at the end. For Scrum, we asked QA to do white-box testing at the end of each Sprint (for the Sprint goals) as well as continuous system-level test in parallel with the continuous integration. Ironically, this worked out very well and the QA team was eager to get into the agile rhythm of Scrum.

Stabilization - for how long did the software have to be stabilized before it could be released? How did you structure this stabilization process?

We stabilized for 1 Sprint. The primary catalysts for this cycle were our “BugTraq” tool, an exception and issue tracking system fed by QA, engineering and customer issues. We dealt with only P1/S1 (priority/severity) issues that did not involve new functionality. Once stable, we pursued a formal UAT (user acceptance test) process with the Stakeholders who validated the end-to-end product. While the same Stakeholders participated in each Sprint demo as a “mini UAT”, this capstone exercise was valuable and fed our corporate product lifecycle requirements and artifacts.

We structured the stabilization Sprint as a mixture of fixed-length deployment tasks as well as time-boxed stabilization “placeholders” for the bug and issue remediation. We tracked this by documenting hours burned on a bug category at the daily Scrum meeting and updating the task metadata with the actual bug numbers and descriptions.

Success - to what degree was the project successful? To what degree was the Scrum process instrumental in the success of the project?

I feel Scrum was the key to our success with this project. We controlled chaos and forced the business and the delivery organization to commit (through behaviors reinforced through Scrum) to actually delivering incremental value and iterative deliverables. It helped the team to focus on simply delivering the solution and shielded them (partially) from the chaos of a shifting business proposition.

In addition, the punctuated equilibrium and continuous adaptation helped us evolve and take new directions that would have otherwise derailed the project for longer than 30 days. Finally, the business really appreciated the Sprint demos and validation opportunities throughout the lifecycle. It increased our credibility with them, and their satisfaction with us as a development team.

Scrum Process – to what degree was the Scrum process implemented "out of the box??

I feel we implemented the Scrum process about "80%" out of the box. We modified the backlog process since we did not have strong governance – it became more of a declaration and validation process. We also used a few Sprints for preparing infrastructure and researching architecture. These did not yield the traditional "potentially shippable code", but rather demonstrated the architecture proposals or working lab infrastructure.

To what degree did you have to modify the Scrum process for this project? For each modification, how did you formulate the modification so that the basic inspect/adapt mechanisms continued to function? What parts of Scrum couldn't be implemented, or failed, and why?

We modified the nature of a few Sprints (ramp-up Sprint, research Sprint), but continued to use the Scrum process to track burn down and relevance of the tasks to the stated goals. We also maintained the daily meetings and "Sprintly" demo aspects to provide project transparency.

The basic inspect and adapt mechanisms fit nicely with this approach – since were able to continuously track our progress, assess our estimates with actuals, load-balance work among the team members – despite the nature of the work being more knowledge-based than code-based.

The part of Scrum that faltered was the cross-team dependencies. We needed to employ the services of a project manager to help us track cross-Sprint sequencing and orchestration. Our Scrum of Scrums helped manage this on a technical level, but many project forces were beyond our control and needed to be tracked externally to the Sprints

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

We took what I believe to be the traditional path for new Scrum teams with respect to estimates. We began by overestimating the task load. After analyzing the first Sprint review metrics, we over-corrected and under-estimated the next Sprint planning. Feeling we would under promise and over deliver, the second Sprint actually left cycles on the table. By the third Sprint – we had hit a better rhythm with estimates and task decomposition. One challenge was to accurately estimate all aspects of each task – from unit test design and code, to primary coding, to review and integration. This three-layer task approach improved our ability to account for all task items – not just the task itself. Our burn down chart straightened out by Sprint three – yielding a nice 45-degree pitch.

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

I think the self-organizing aspects of Scrum helped us delivery what we committed – even if it was stubbed or simulated and refined later. The Sprint commitment was a team covenant – and we had to live up to that agreement. By continuously inspecting the daily progress, we could more confidently either stick to or revise our Sprint goals – this ironed out in the later Sprints. Secondly, looking at estimates, actuals and roadblocks/dependencies helped us "protect" deliverables and cohesively ensure they were "doable" versus risky.

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 track a few development metrics inside engineering (unit test, code/test coverage, formal QA, code reviews, performance testing) but most are milestones tracked through an external, master-schedule process. This is problematic for an Agile approach such as Scrum, but we simply added those artifacts or measurements as tasks to be completed within a certain Sprint.

We removed a great deal of the heavyweight metrics enforced by our corporate lifecycle – and merely fed that larger wheel with our smaller iterative wheels. The new measurements added were obviously burn down, estimates and task per requirement/feature/use-case coverage. These were quite effectively translated into higher-level metrics and dashboards.

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

In the future, we hope to employ more formal Scrum training and certification for leaders, managers, engineers and executives. Scrum brings about a culture change – and we need to do a better job on the philosophical and structural levels, in addition to the tactical ones.

In addition, our current tool, XPlanner, does not facilitate backlog governance as much as we'd like. We are searching for new web- and data-based tools to help automate and relate all of the data and metrics involved in Scrum planning and execution.

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

I've found (and researched) that Scrum provides a good management structure around the agile development practices of XP. Highly complimentary, Scrum forces a rhythm to the dynamic and rippling aspects of XP. Scrum is the planning, management, validation and delivery container within which XP focuses on development, coding, testing and refactoring.

There is \$100/year fee to register with the Scrum Alliance as a Practicing CSM. Visit the Scrum Store at www.Scrumalliance.org to register or go directly to <http://tinyurl.com/58ugy>.