

CSM Practicing Certification Renewal Assessment

Name: Kelley Louie email: kelley@danube.com

date: 1/28/2005

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. Describe:

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

My company offers software project services to external customers. We build customized software to meet our customers' business needs. Over two years ago, a management consulting firm (Client) gave us a Scope of Work to build a student information system (SIS) to support a standards based school district (School District).

The Product Owner is the Client, but we are building the product based on the School District's business needs. The complexity lies in the fact that the Client and the School District have different goals. The School District wants to automate their unique educational system, whereas the Client wants to scale the project for resale to other school districts. The School District wants to build features that would alleviate some of the workload for its employees. The Client wants to generalize features such that it can be marketed to potential customers. At times, the School District requests features that would be restricted to their own school district and would not be applicable at other school districts. In order to develop a product, we continually balance the School District's features with the Client's generalized functionality and scalability. The Product Owner should have been the School District, but the Client insisted on being the Product Owner and act as an intermediary to the School District.

Ultimately, the project's goal is to automate the complex tracking of student progress. The School District was buried under piles of papers to support the standards based system. Since the system does not resemble the K-12 educational system, it is foreign to users not familiar with the district's "Quality Schools Model". By automating the school information system, it would alleviate some of the user's workload and demonstrate to other school district concepts and ideas that were implemented by hand but could not easily be explained. One such concept is standards. In a K-12 educational system, students work on homework assignments and then grades are assigned to the homework. The School District tracks grades based on how well students understand the concepts the homework assignments are meant to teach. Standards are simple to explain now, but this concept was difficult for the team to grasp because the School District was entrenched in these ideas. It was difficult for the School District to explicitly communicate their business rules. The only way to build the software was to develop our domain knowledge over time, allowing the School District to refine the software until it met their business needs.

- Length - what was the duration of the project?

The original project began in February 2003. We delivered the first version of the product in August 2003. Teachers and administrators were extremely satisfied with the initial set of features provided and anticipated more features to be implemented. After the success of the first version, the School District and Client signed another contract for my company to build the second version in December 2003. We are still working on the second version which is scheduled to be completed in March 2005.

- Cost - what were the budgeted and actual costs?

The first phase of the project was in the low six figures and the second phase of the project is about \$750,000. The total for the project is about \$1,000,000.

With any project, there are three scales: scope, cost, and schedule. For the first version of the SIS, the cost and schedule were fixed. My company signed a fixed bid

contract to win the contract and the School District needed the SIS for their 2003-2004 school year. Therefore, we could only scale down on the scope of the project. Initially, the Scope of Work only vaguely defined requirements to implement; the details were lacking. After the initial phase of analyzing the standards-based curriculum, we determined that all of the features within the Scope of Work could not be completed. We proposed a set of features and milestone deadlines. With the Client's approval, we began developing the software according to the first set of requirements that was determined by the School District and team as manageable. We proceeded with the highest prioritized Backlog Items until the first phase was tagged and deployed for the teachers and administrators to use within the school district. We had to constantly negotiate with the School District and Client to determine the features and robustness that would be implemented in the first version of the SIS.

For the second version, the Client has a set of features in mind he expects to be implemented. Some other stakeholders were expecting certain features that were not implemented in the first version of the SIS. This information was unknown to my company until the beginning of the second version. We established the initial product backlog for the second version of the product based on the Product Owner and stakeholder expectations.

For the second version, the cost and schedule are fixed. The School District allotted an amount that can be used for the SIS. The schedule has been determined by the date in which the funds will run out. As a result, scope is the only maneuverable variable. We are negotiating features and robustness with the School District and Client as we develop.

- Value - what were the projected benefits and actual (if measured) actual benefits?
The School District anticipates that teachers will be able to use the SIS to reduce their manual paper workload. With paperwork strewn across schools that are hundreds of miles apart, each teacher has his/her own *ad hoc* process to track progress. With a single automated system, all users will conform to the business rules built into the software. This aids teachers that may not be as familiar to the School District's unique curriculum and ensures that teachers are correctly using the standards based curriculum.
As of today's date, no other school district has purchased the SIS. The School District hopes to expand the concept of a standards based curriculum with the SIS as supporting software. Potential customers can physically use a software implementation of the concepts used by the School District. It will no longer be charts and diagrams detailing how to implement this curriculum at another school district. The Client projects to expand this SIS to six more school districts within the state of Alaska.
- Size - how many people were on the project team(s), how were they organized into teams?
For the first phase of the project, the team consisted of four people. The team consisted of one Scrum Master, one senior architect, one developer, and one on-call UI designer. The Client requested that the team have at least one architect. The Client wanted at least one person with many years of software development experience.
For the second phase of the project, we have shrunk the size. There is one Scrum Master, one senior architect, and one developer.
- Teams - were the teams cross-functional and self-organizing? Were the teams collocated in an open space? Were the teams physically separated within one location, or located in more than one physical location?
The team was originally split into two locations. The Scrum Master and architect were in the Seattle office and the developer was in Hungary. With a split team, the developer was not involved in the lengthy business requirements gathering meetings. Therefore the architect aggregated business requirements and distilled the information to the developer to implement into working code. Everyone agreed that this was not the ideal working situation. The developer should be involved in all requirements meetings to

fully understand the business requirements and to bring his expertise of the software to the conversation. With one developer abroad, communication was impeded. The architect and I had to email our questions or wait until the developer abroad was available. Ideally, the team should have been co-located from the start.

My company eventually filed for a visa to bring the developer to the US and as of November of 2004, the off-site developer joined our US team in Seattle. Our office has a bull-pen. This allows team members to quickly and freely begin conversations. Once the developer and architect were co-located, they were able to pair program and communication flowed freely. This increased code quality and transfer knowledge. We also preserve time delays due to the different time zones and work schedules. The team's productivity and efficiency increased greatly.

I cringe when I look back at when we were working with 1/3 of the team offsite. It has been much simpler having the programmer co-located. All team members participate in the requirements gathering meetings. Every team member can voice their opinions on changes to the application based on their experience. No longer does the Client and School District need to wait a few days before determining the complexity of changing a part of the system that the developer is the most knowledgeable. In terms of converting features into working code, the architect and developer meet to discuss and determine how to implement features. When I find bugs during QA, I can demonstrate the bug to the developer on my development environment. By having each team member co-located, the software development has become more collaborative which is essential for Scrum.

With such a small team, the team has to be cross-functional. No one person can only work on tasks defined by his/her job title. Each person has to do whatever needs to be done for the project to progress. As Scrum Master, I realized that I had spare time to devote to this project, so I became the test engineer. The architect designs the system architecture, performs code reviews and writes the code involving difficult business rules or architectural complexity. The architect has taken on the role of lead programmer. As a fellow team member that partakes in the requirements meeting, we discuss how to fulfill the School District's and Client's business needs with software. The developer codes the features that the architect has prepared for him. This breakdown of roles evolved after each person began to tackle the work on hand and is unfortunate fallout of not having the developer in the bull-pen.

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

It was a slow process to initiate everyone to the concept of Scrum. For each concept, the purpose was analyzed. Each concept was dissected into the following questions: "What do we do, and why do we do it?" People are more receptive to change if they know the underlying goals. Otherwise, people assume that the work is not productive and detracts from development which matures as an aversion to the change. As the ScrumMaster, I ensured that Scrum concepts were applied in the project. If a concept was not implemented or was on the wrong track, I would collaborate with the team member to determine why the implementation and concept differed. Each project's implementation of Scrum varies, so I wanted differing ideas to develop the appropriate Scrum processes for this project.

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

The team holds semi-weekly requirements meetings to clarify questions that pop up during development with the Client and School District. Before the semi-weekly meetings were scheduled, the team was often blocked by details that they could not foresee during the initial meetings to understand the features.

As time passed, the purpose of these meetings expanded. The original goal of the meeting was to clarify requirements, but it evolved into a feedback mechanism as well. We have found that discussing features without visuals often created more

confusion than solutions. During the meetings, we would show mockups or demonstrate partially completed features to aid the communication. The Client and School District loved seeing the work in progress because their ideas were actually implemented into working software code. The team would demonstrate fully functional features as they were added to the system.

However, this opened the door to the Client and School District making requests that would derail the progress of the current sprint. We quickly handled this problem by classifying the request as a minor or major request. If the request was a minor change such as text, we would inform the Client and School District and make the change. If the feature did not fit within the sprint, I added it to the product backlog. The team would remind all meeting participants that we should discuss the details of the new work when we are about to implement the feature because the feature could evolve by implementation time or the feature could be prioritized out of the current release. The major feature requests would be discussed at the sprint planning meeting.

The semi-weekly meetings have proven to be integral to our process since we will not be gathering details about features that are not within the current sprint. It has focused all stakeholders' minds to the current work and everyone has the same goal of meeting the sprint goals.

In addition to the semi-weekly meetings, I send the Client and School District a weekly report that states the goals of the sprint, describes the status of each task, features for the current release and possible features for the next release. Task status is either "not started", "started", "blocked", or "finished". The next release's features are based on the priorities as determined by the Client and School District. I started writing this report at the beginning of the project when the Client was nervous about progress and before the semi-weekly meetings were scheduled. I have maintained writing this report because it takes only ten minutes and has become an expectation of the Client and School District. In general, we rely on the semi-weekly to demonstrate our progress and the Client and School District love this process since they can see the most up to date code which is truly the best metric.

Progress is reported to management during a weekly meeting in which different divisions of the company gather for a meta-scrum and discuss the work completed since the last meeting and the impediments that management could help alleviate.

- Change - what difficulties were surfaced by Scrum that had to be resolved? How were these resolved?
- Success - to what degree was the project successful? To what degree was the Scrum process instrumental in the success of the project?

The first project was extremely successful. The Client and School District were so satisfied with the product that they signed a second contract in December 2003. Without Scrum, the team would not have been able to build an application that would please both the Client and School District. Features would have been half implemented. The team would have been side-tracked by requests from either the Client or School District. The high visibility of Scrum has focused all of the stakeholders to the highest prioritized Product Backlog and set the expectations for both the Client and School District.

Working as a sub-contracted company forced the team to deal with feature requests from the Client and the School District. By defining the timeframe of each sprint, the team could only implement on an agreed upon set of features. If the Client requested features outside of the sprint, we showed our task breakdown and demonstrated that we had planned our tasks for the current work cycle. Any feature requests would adversely affect the current sprint and that either features accepted by both the Client and School District would be pushed out of the current sprint or the new feature request would need to be revisited at the sprint review and sprint planning meetings. This solved the underlying problem of working as sub-contractors to build software that would satisfy the end client and the middle-man.

The School District wanted frequent production deployments, but each

production deployment required overhead work to determine that the software is relatively bug-free and would support the existing production data. The initial version of the software allowed users to enter corrupt data into the system. Unfortunately, we did not have the foresight to see this as a problem until the second version of the software. With hindsight, we now know that some of the data was erroneously entered into the system or the requirements were not clearly defined. As a result, we retrieve production data on a regular basis to test on our development environments to hopefully find any possible issues earlier in the development rather than the day of deployment. Another issue is that the first version of software had no automated tests. The application needs to be tested manually. We informed the client that deploying to the production server would slow down the rate of new features completed because of the QA testing overhead and assuring a solid piece of software. The School District has accepted frequent deployments to their demonstration server and periodic deployments to the production server.

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

I was a project manager and tester prior to assuming the role of ScrumMaster. I believe that I have been fairly successful as ScrumMaster because I never believed that I knew all of the details and complexity of software design and implementation. How could a project manager determine the schedule and propose a date of completion when that person is not working directly with the code?

I have maintained my role as tester, but this does not conflict with our self-organized team. I do not try to control the project. I have built my reputability as a fellow developer by helping the team test the application and not dictating the progress or schedule. These issues are dealt with during the sprint review and sprint planning meetings.

Accepting the role of Product Owner was simple for the Client, but truly following the rules proved to be more difficult. The Client maintains a business relationship with the School District and is consulted before changes to the norm are asked of the School District. We asked the Client to prioritize the product backlog with the School District. After a face to face meeting between the Client and School District's assistant superintendent did not result in a prioritized backlog, the team pushed for priorities and was finally allowed to discuss the priorities directly with the School District with the Client's input. The team began to tackle features based upon the priorities. Some features were pushed out of the current sprint or release because it was lower in priority. Even though this pushback occurred, the School District was satisfied because their highest priority features were added. After the first release, the Client saw the importance of a prioritized Product Backlog. No longer were features just added whenever the School District thought that it would be nice to have. Now the School District weighs the different options whenever they request features. The School District and Client are making more informed decisions.

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

By demonstrating working functionality to the client once it has been implemented has changed some of our engineering practices and environment. We no longer write up lengthy design documents trying to determine the overall design of the application. By working on the next highest priorities, the School District can change the features before each release and sprint. We do not know the overall feature set, so we cannot plan an overall design. The only plan in place is that we should not lock the design into something that is not flexible and maintainable. That is the overall design of the application.

We only create UML diagrams, mockups, and use cases for requirements clarifications. Producing and maintaining technical documentation for the software is not meeting the School District's business needs. Working software, not diagrams or documents will meet their business needs. Technical documentation that no one will

read becomes outdated so quickly that constant updating of the documentation would be a Backlog Item in itself. This is not important enough to the School District so it has been prioritized out of the sprint and release.

We strive to provide fully implemented features at the end of each sprint, so we thoroughly test the application. Since the client will see the product during the sprint review, we do not want to embarrass ourselves when demonstrating buggy and fragile software. From experience, we have determined that the best time to QA test is immediately after the feature is completed. The code is still fresh in the programmer's mind, so it should be faster for the programmer to fix the bug. We find most of the bugs during this time frame and fix the bugs.

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

The release date is planned far in advance. It is typically at the end of the third sprint. This is enough time to bring several different Product Backlog Items into fruition and it keeps the end-customer (teachers) interested in the product. Each sprint results in at least one completed feature. If the feature is complex, an automated test is written. Each feature is manually tested. The manual test consists of tedious data confirmation and affirmation of client expectations on supported browsers.

During the last week of the third sprint, the team focuses on finding and fixing bugs. The latest version of the application is uploaded to a demonstration server for acceptance testing by the School District. Luckily by constantly QA testing, we have not found too many bugs. If a major bug crops up that would delay the release, we immediately contact the Client and School District and inform them of the problem. On several occasions, the School District has accepted a temporary work-around with the knowledge that the full bug resolution would be prioritized within the Product Backlog. The team fixes bugs and implements any minor robustness requests during this week of feature freeze. With the School District's acceptance, we deploy the release to the production server.

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

The team implemented Scrum by following the guidelines of the different types of meetings. This was a starting point because we focused on the goals of each of the different meetings. If the meeting seemed to be heading in the wrong direction, then as Scrum Master, I would point out the potential problems and discuss it with the team to determine a possible solution. Our implementation of Scrum has always been collaborative. The process will only work if every team member buys into the concepts.

Originally, a consultant recommended a two week sprint to keep the Client and School District interested. The sprint planning, definition, review, and retrospective meetings pulled the team away from work that produced code for a full day. The remaining time for the sprint was not enough time to fully understand and implement functioning code. We pushed out the two week iteration to three weeks and have remained with this cycle. The team is now acquainted with this time frame. Usually the first and second weeks focuses on understanding the features. The last week is spent on implementing the remaining features. This work cycle has proven to be productive for our team and allows the School District and Client to constantly test on the demonstration server after each sprint.

The team thinks about some of the principles of Scrum for each of its proposals or decisions:

1. High visibility
2. Self-organized teams
3. Functional software

In terms of high visibility, we inform the Client and School District of all possible delays and impediments. The School District and Client should be informed if features take a longer period of time and should make the decision to continue with the current work or to re-allocate the team to another feature. The Client and School District can not make informed decisions if we do not disclose all of the relevant information. The team demonstrates features more frequently than only at the sprint review. We have found this invaluable for our requirements gathering and it keeps the Client and School District well informed. Roles were naturally taken on by each team member. The architect has the most experience in software engineering and became a lead among the team. The architect drives the requirements gathering and design of the application. I have been the project manager and QA for all of the projects with my company, so I took on these two roles in this project. The developer has been coding many of the different parts of the software. The architect and developer pair program when new designs are introduced or blocks need to be resolved. The underlying goal for all stakeholders, Client, School District, and our team is working code that will help the teachers. By keeping the Scrum principles in mind, we have implemented a version of Scrum that works for our team and project.

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

Retrospective meetings are a perfect opportunity for the team to inspect and adapt Product Backlog estimates. Team members are accustomed to reflecting on the sprint and can easily inspect another aspect of the project. Our team focuses on the following aspects when determining the Product Backlog estimate:

- Source
- Stability
- Complexity

The source is important because it determines the clarity of the feature request. If the school district's assistant superintendent requests a feature, he has thought about the business needs that he wants to meet. If the Client requests a feature, he is usually relaying requests from other parties and it will take more time to determine how to build the feature and the feature is more likely to evolve. Stability is the likelihood that the feature will differ. Complexity of the feature is based on several things: is the feature dependent on a third party? Does the feature replace an existing concept? The lower the stability, the more likely it will take more time. The higher the complexity, the more likely it will take more time. The longer the team works on a project, the team will become more knowledgeable about the technology, domain, and personalities and be able to provide more accurate Product Backlog estimates.

First, I want to define an accurate estimate. An accurate estimate is not the accuracy of time, but it is the accuracy of complexity. We can not predict all of the bumps on the road that will slow down development of a feature, but we can estimate to our best knowledge, the complexity of a feature. From historical sprints, we can determine that we can complete X amount of Backlog Items. We need accurate estimates to determine how much work we can promise for a sprint, release, and contract. We need to be able to fulfill our promises. The Client and School District does not want empty promises. The team can not renege on the features that we promise to be completed. If the team could not keep the promises, then the Client and School District would lose faith in our estimates and could lose faith in the team as a whole.

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

For this project, we focus on releases more than sprints. We aim for finishing features within a single sprint, but the Client and Product Owner want a bundle of features built within a release to be deployed to production. The Client and Product Owner understand that each production release has a bit of overhead and detracts from

the amount of feature work the team can commit to.

The team tackles the highest risks in the first sprint of the three sprint release. In the most recent release, we focused on a complex report that aggregates information from several sources. The team asked many clarification questions during the semi-weekly meetings. As we delved deeper into the feature, the Client realized that the feature did not provide as much business value as he had hoped. The report was being generated by a third party and the software did not need to generate the report for the School District. The amount of work needed to implement the report greatly outweighed the business value. Therefore, the Client and School District decided to drop the feature and replace it with another feature. Fortunately, the team was prepared for such an occurrence because it considered the complex feature to be high risk and attacked it in the first sprint. We changed the direction of the sprint and still met the proposed deployment date to the production server.

We keep the mentality of attacking high risk first within sprints, too. The direction of the sprint is clearly shown in the semi-weekly meetings. These meetings have become another time for inspect and adapt. The team can gauge the amount of work of each feature within the first few meetings of the sprint. If the team needs to adjust the features promised to be delivered, the team will bring it up during the meetings.

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

It would be very tough to use a set of resources to train people to use Scrum. Scrum changes with each project and each group of people. It is an empirical process that can not be assembly line produced for each scenario.

It is invaluable for experienced Scrum users to mentor new Scrum users. When our team first used Scrum, we read the books and thought that we understood Scrum. As time passed, we realized that we truly had not grasped the concepts and principles of Scrum. Things started to smell bad and we came to the conclusion that our implementation was not in line after discussing it with more experienced Scrum users.

Although there is no substitute for experience, I would love to have some tools and aids that would help the team use Scrum. It would be nice to have a tool that could easily generate chart burn-downs, generate reports, and manage backlog and tasks. The tool should enable high visibility by being accessible online and user friendly. For these reasons, my company has been building the ScrumWorks product. While using ScrumWorks, the team has to remain flexible and not let the tool shape our implementation of Scrum. We need to constantly look for the things that smell bad and fall back on the Scrum principles.

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

The team must choose to implement Extreme Programming. If it is a request from management then the team will be reluctant to implement such a change. If the team is successful, then why should the team change its software engineering practices? If the team is not successful, why would Extreme Programming solve the problem? This would conflict with a self-organizing team. The Product Owner informs the team about the feature to be implemented and the team should choose the manner in which to implement the feature. The change needs to be come from the team itself. It could be suggested at a sprint retrospective.

Another issue that team members must be comfortable with XP. Changing engineering practices may be difficult for some individuals. For example, with pair programming, the more experienced programmer must be willing to mentor and teach the less experienced programmer. If either the experienced or less experienced programmer is unwilling to work in such a situation, then pair programming will fail. I have found that teams will find solutions that will meet their needs and Extreme Programming offers some ways for the team to become more productive, but the team must make the choice.