

Product Savvy Consulting, LLC



Product Savvy Consulting, LLC

One Research Court, Suite 450

Rockville, MD. 20850

Phone: 240.403.4025

Fax: 301.519.8001

info@productsavvy.com

www.productsavvy.com



1. Case study: Distributed Scrum Project – US / Israel - 2011

1.1. Executive summary

Scrum is an iterative process for developing and managing software products. Product Savvy Consulting harnesses this agile process to create software product in situations of evolving requirements and where control over chaos is essential. In this case study, we describe how we successfully executed a Scrum project, which included developers in Israel (with our partner Log-On), lead by a Product Savvy Consulting Scrum Master and Product Owner in the US. We conclude with some of the key management techniques that made this globally distributed project a success.

1.2. What is Scrum?

Scrum is an iterative, incremental process for developing and managing software products. It produces a potentially shippable set of functionality at the end of every iteration. Scrum is an agile process to manage and control development work. It is a team-based approach to iteratively, incrementally develop systems and products when requirements are rapidly changing and is most famous as a process that controls the chaos of conflicting interests and needs.

Running, tested features are an Agile team's primary measure of progress. Working features serve as the basis for enabling and improving team collaboration, customer feedback, and overall project visibility. They provide the evidence that both the system and the project are on track.

In early iterations of a new project, the team may not deliver many features. Within a few iterations, the team usually hits its stride. As the system emerges, the application design, architecture, and business priorities are all continuously evaluated. At every step along the way, the team continuously works to converge on the best business solution, using the latest input from customers, users, and other stakeholders. Iteration by iteration, everyone involved can see whether or not they will get what they want, and management can see whether they will get their money's worth.

Consistently measuring success with actual software gives a project a very different feeling than traditional projects. Programmers, customers, managers, and other stakeholders are focused, engaged, and confident.

Yet, the implementation of Scrum and adapting it to the changing realities of different projects is a large factor in the success or failure of those projects. In this case study, we describe how we successfully executed a 5 man-years Scrum project, which included developers in Israel (with our partner Log-On) and a (Product Savvy Consulting) Product Owner and Scrum Master residing in the US.

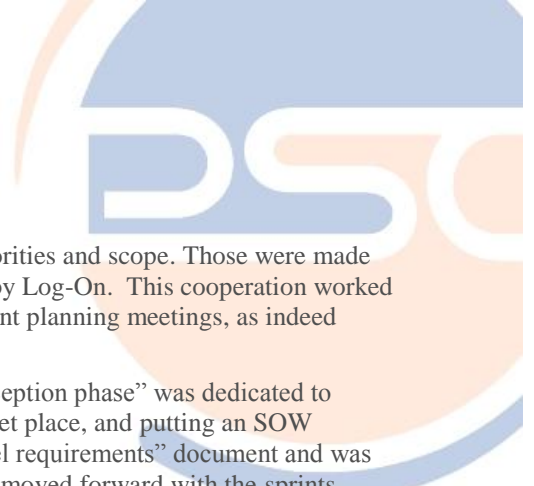
1.3. Background

The system we set-out to build was a Health IT related system, that allows immediate, on demand access to one's medical records in a private (username / password protected), secure (256 bits symmetrical encryption), way, anytime anywhere, as needed . The system was developed for a US company, with offices in Bethesda, MD.

The system has a server side component (IBM technology based) and a mobile component, which includes an Android and iOS version, both available on the perspective markets.

The customer engaged our company to build the said system from scratch. We introduced an Agile approach using Scrum, focusing on close cooperation with the customer, open communication and working in small increments of 2 weeks.

We started the project with what we call the "inception phase" to gather requirements, set up the development environment and everything that is needed before the first sprint starts. This 8 week phase was done by PSC's Product Owner.



The Product Owner, was responsible for the high level decisions about priorities and scope. Those were made by cooperation between PSC's Product Owner and the Team Leader provided by Log-On. This cooperation worked well, and we strongly recommend that both people always be present at the sprint planning meetings, as indeed happened in this project.

As there were no customer requirements at all, a significant part of the "inception phase" was dedicated to gathering those requirements, learning about the existing competition, the market place, and putting an SOW (Statement Of Work) document together. This SOW served us as the "high level requirements" document and was the base for any future requirements that were refined further and further as we moved forward with the sprints. Even though an SOW existed, the actual development and "detailing" for each sprint was driven by user stories that we put together and that helped us better understand and analyze the usage of the system.

As we moved through the sprints, each had an estimate assigned to it at the sprint Kick-Off, and those estimates were tracked using a "burn-down" chart that also helped us estimate the velocity of the team. It is clear to us that estimates are ever changing, but, some estimates (that get better as we move forward) are better than no estimates at all.

1.4. Setting up and Executing

The project started off with a 5-person team working in two-week iterations. To align expectations and create personable relationships between the team in the US and the team in Israel, the first sprint started with the Product Owner (PSC) visit to Israel during which system requirements were reviewed with the team, basic user stories discovered and discussed, system architecture was put together and the initial development started under the supervision of the Product Owner.

One of the first steps towards becoming a team is to agree on how to work together. To facilitate that, we held discussions (while visiting Israel for the kick-off) with all team members including the Product Owner and Scrum Master. In the meetings we decided on practices like how to do programming, use of tools, quality targets, core hours etc. Getting the team's buy-in on the process, as well as involving them in the kick-off and setup of the development environment proved as a very good practice that allowed us to make sure that we all are on the same page and confident in each other's abilities.

In the first sprint, the team proved to be able to build, test and demonstrate basic capabilities and some user stories. To us, using sprints to demonstrate progress is a great way to keep customer happy, this way, customer sees progress very quickly and has control over the progress of the project.

It is interesting to observe and see how a Product Owner and a Scrum Master can work with a development team, across the ocean, and yet feel like they share the same office. So, first of all, we use Skype intensively. We have webcams, headsets and microphones. This way, we can do one-on-one meetings as well as all-hands meetings together. We also use WebEx, with this "screen sharing" application, we can see each other screens, share designs and "white board" ideas almost as if we are looking at the same board, or computer screen, in the same room.

SubVersion was used for code repository, document repository and test cases repository (version control). Good network connection is crucial for this. All of this works with off-the-shelf, cost effective components and existing software.

Finally, we used VersionOne (Team Edition) as a tool to keep track of who is working on what and how the sprint is progressing. Because of our distributed teams this worked much better than a whiteboard. VersionOne also worked well when discussing the product backlog with the product owner, developers and client.

It is important to notice that VersionOne has been selected because it was successfully deployed by over 10,000 teams (70,000+ users) around the world and more than 40 Fortune 100 companies using agile software development and scrum development practices.

Yet, we did have to address some challenges. For instance, the definition of "in progress" vs. "done" tasks by the development team. At the early sprints, some developers would mark requirements as "done" even if they were less than 100% complete, mostly due to a different set of "parameters" that would render a task as "done". As we addressed that in the early sprints the problem was solved, and reporting became more and more accurate as we moved forward.

Another drawback was management of bugs...trying to figure out how to mark those that are addressed, in progress and completed. We eventually decided that only the QA person can mark bugs as “Done”, but, everyone, can open and report bugs, thus, even the developers could report bugs that they saw in the system.

1.5. Requirements

Our Product Owner (Pragmatic Marketing certified), created the High Level Requirements – an extensive document based on customer meetings and requirements gathered during the inception phase. For our process just user stories on a backlog and the explanations of the Product Owner during the planning meeting were all that was required to move forward with the sprints. Detailed requirements that were drilled down during the sprints were recorded into the VersionOne system and used for planning the sprints.

Our user stories, plus product owner explanations, were sufficient for building and testing the software in our Scrum team. The high level requirements document was valuable for providing a good system overview to the developers and serve as a “go-to manual” for the team members.

1.6. Testing

We applied testing during the project to allow us to deliver tested software at the end of each sprint, without significant regression bugs. We did have an issue with regression bugs, but this issue was addressed by reaffirming the build process and tightening up control over it. We also required the testing team to repeatedly perform “sanity tests” at the release stage of every build. After a build passed sanity (meaning, no regression bugs) the QA person went forward and tested the new features for each build.

We tests had two levels: unit tests and QA tests. Unit testing was done by developers, prior to delivering the different modules to the build, while QA was done only once an official build (usually once a week) was released. An advantage of this approach is that the testers can be effective from the start of the sprint, creating test cases before the user story is implemented and using those test cases to test the system once the build is ready. All Test Cases were kept in the same repository as the code, SubVersion.

1.7. Deployment

The system is deployed on the IBM cloud, a great way to allow small companies to utilize high quality, robust servers at a fraction of the cost.

The system is using WebSphere and DB2 as the underlying IBM technologies and is currently deployed in two environments, “production” and “staging”.

While the production environment always holds the QA’d, released version of the product, the staging environment is used for intermediate releases, before they are approved for production. This way, we always have a working version, while the “next version” is in the “cooking” stage.

1.8. Outcome

Hindsight shows that, as with most projects, the required functionality shifted as the project progressed, yet, it was maintained within budget and original time estimates. Progress was regularly discussed with the client while the work was going on, and they expressed satisfaction with the result. Roll-out time was dictated by the client, as roll-out depended on market forces as well as system readiness.



2. Lessons Learned

- Scrum is well-suited to execution with distributed teams. Quarterly trips combined with daily Scrum calls created an effective communication channel that allowed us to address many issues in a professional, amicable manner.
- Having a product owner with both detailed knowledge of the requirements as well as the mandate to set priorities is crucial. This way, client may avoid having more than one person sharing responsibilities. Having a one person responsible, made everyone's life easier.
- When planning the project time and budget, it's important to make sure that a good portion of the product backlog is available and estimated (70 – 80% if possible). Any estimate is better than no estimate, in combination with the team's velocity and burn-down, release planning can be done.
- Requirements are necessary, but, cannot be fully estimated at the initial stages of the project. Further, requirements cannot replace the use of user stories and, with the initial set of requirements each requirement will be drilled down as the project evolves.
- Having a face-to-face meeting in Israel (during which requirements are reviewed and working procedures are established) is crucial for the project's success – in no way should such a meeting be avoided.
- Testing is vital to deliver software incrementally, unhindered by regression bugs. Before the project is over, the return on investment will outweigh the cost.