Article

Skills for Scrum Agile Teams

Posted by **Prasad Prabhakaran** on Jul 26, 2010
Community Agile Topics Adopting Agile , Team Collaboration , Unit Testing Tags Automation , Refactoring , Continuous Integration , Productivity
Share |

This paper brings out the uniqueness involved in Agile Scrum projects, and identifies the traits required for a successful team. The paper concludes with recommendation of interventions required for effective scrum team.

WHITE PAPER

VERSIONONE

THE AGILE BUSINESS ANALYST

EXECUTIVE SUMMARY

Engineering uniqueness of agile projects

There are several things which cause an agile project to be different from projects based upon more traditional approaches, including:

1. **Setting up the development environment**

   In a traditional project, the team can spend sufficient time in setting up the environment; in case of agile teams, they need to be productive form the first hour. From our experience, we have realized that lack of documentation on setting up the development environment is a key reason why the setup time is large. The second key reason is the number of manual steps involved in the setup process. In sprint 0, we must document every little thing that a developer needs to do in order to start writing code and integrating with the rest of the team's work.

2. **Automated builds**

   Let us fail early! We have learned that manual builds are liable to be both fragile and specific to a single machine, and time lost to making those builds work is time lost to development and testing. On anything but the smallest projects, having an automated build process is essential. We realized that, even if you have to take time out to create an automated build environment, it's time you'll get back later. It also makes it simpler to ensure that we have a standardized build that everyone on a project can share. Key tools we've used include Ant, Maven and Nant.

3. **Continuous integration**

   Form our past experience we have learned that waiting for weeks on end before we integrate code from different team members is a recipe for disaster. If you've got an automated build in place, the next thing is to go for continuous integration. Of course an automated build and continuous integration environment pre-supposes version control (or Software Configuration Management to give it a more formal and impressive name). A key lesson learned is, the sooner that you identify integration errors the sooner you can fix them. Key tools we've used include CruiseControl, CruiseControl.Net and Bamboo.

4. **Unit testing**

   In a highly fluid environment with multiple developers, shifting requirements and changing priorities it's essential to ensure that what worked yesterday works today. We also had challenges with integration errors. A practice (which we learned the hard way) is to use unit tests so that code changes do not break existing functionality. We also started writing unit test cases before coding. Key tools we've used include JUnit (and other xUnit tools such as NUnit, HTTPUnit, etc) and MockObjects.

5. **Refactoring**

   In a traditional environment, normally an individual protects their codebase until integration, but in agile we practice code ownership - in this view all code belongs to all developers, who are free to improve the code when they feel it's necessary. Over a period of time, our code base started behaving strangely - the solution to this was refactoring (thanks to Martin Fowler who popularized the term refactoring in his book of the same name). Refactoring essentially boils down to code changes which improve the structure and clarity of the code without necessarily changing the functionality. A key lesson learned was to have unit tests as a safety net before refactoring the code, and some key tools used include Eclipse, NetBeans, IntelliJ IDEA and Visual Studio.NET.

As it is clearly evident there are certain unique things in agile projects engineering practices, so the teams need to be prepared for and oriented towards these practices.

Behavioral traits required to work in agile teams

As agile teams work differently from normal teams and depends a lot on effective and efficient communication and fast execution, there is an increased need to use soft skills. If we are aware of this and actively encourage the use of some of these traits and skills, we can make agile teams more meaningful and productive.

Self-organization usually relies on basic ingredients like Positive feedback, Negative feedback, Balance of exploitation and exploration, and multiple interactions. From our experience, a team may fail to give the right kind of feedback and shy away from interaction because of many cultural and social issues.

From my experience, this remains a 'myth'. Always we tend to have 'predictability syndrome' - the more planning we do the more predictable we will be.

The team needs to have good discipline, the ability to take responsibility, be committed, and take accountability and ownership.

One of the key skills that the team needs to have is the ability to ask for help and to seek out review. In certain cases we had seen the 'ego' factoring out as a major impediment.

Taking responsibility, being committed, and a spirit of collaboration are sometimes taken for granted; however from our experience we sometimes need external interventions to make these happen.

Certain key skills which we normally tend ignore are the ability to take initiative, enjoying working in an intense environment and adapting to new situations and frameworks easily.

Most of our projects are distributed, meaning there will be a co-development scrum between client and the service provider. In these contexts, skills like managing diversity, time management, diplomacy and leadership are very essential.

Success 'Mantra' for agile teams

For any agile projects to be successful and hyper-productive, the team needs to show **more enthusiasm** and the right attitude towards ,**learning from peers** in spite of seniority and expertise. A **safety net** for fearless expression needs to be ensured so that **real camaraderie** can be exhibited, which in turn will increase focus on the goals of the team rather than 'what is in it for me'?

Conclusion

From my experience and observations, the skills required to be hyper-productive in an agile project are different from those required by a traditional one. We have identified behavioral and technical skills required for a team to have that edge. Anyone who acquires these 'delta' traits will be equipped with the right set of behavioral and technical skills, which enable them to work effectively in an agile project. The summary of the skills are represented in the table below.

Skill table

Role	**Technical skills (in respective platforms)**	**Behavioral skills**
Developer	CRUD operations, interfacing with different layers of the development frame work. Unit testing (tools – NUnit, JUnit) Code coverage concepts and tools Code review concepts and tools Continuous integration tools Refactoring concepts Code-smell concepts Scrum process	Communication Collaboration Time Management/ Planning Thinking Conflict Management Dealing with Change/ Flexibility Decision making Teamwork/ Teambuilding Handling stress Problem Solving Leadership Diplomacy
QA	Definition of done -> acceptance criteria Test management Automation / scripting Environment setup Database concepts	Same as developer
Scrum Master	Scrum process Templates and usage Project Management tools Continuous integration tools Development environment setup	Developer skills + facilitation

About the Author

Prasad, has 10 years of experience in the IT services industry, His first exposure towards agile was from a Microsoft project in 2005; from then onwards he did solutions development, coaching, consulting, and teaching on agile and its flavors for many companies like GE, Cisco, Coke etc. Currently he is working as a Manager in Agile Labs at Symphony Services. 40% of projects at Symphony are in some form of Agile and its flavors, and started providing business critical value for the customer through Agile since 2004. He can be reached at pprabhak@symphonsysv.com

5 comments **Watch Thread** **Reply**

XP for dummies? by Rhys K Posted Jul 26, 2010 11:42 AM
Re: XP for dummies? by Luca Minudel Posted Jul 26, 2010 12:30 PM
Re: XP for dummies? by Prasad Prabhakaran Posted Jul 28, 2010 1:58 AM
Scrum - no commitment, just a buzz-word by no scrum Posted Jul 27, 2010 4:16 PM
not in the real world by phloid domino Posted Jul 27, 2010 5:11 PM

Sort by date descending

XP for dummies?

Jul 26, 2010 11:42 AM by **Rhys K**

You can't seriously suggest there are readers of infoq who learnt anything by reading this article.

Reply

Re: XP for dummies?

Jul 26, 2010 12:30 PM by **Luca Minudel**

@Rhys K

for sure this article could suggest to you a way to improve: how to provide feedback in a constructive way.
under the column "Behavioral skills" indeed you find Collaboration and Conflict Management. The same for Developers, QA and SM.
if you need more help, I can suggest an ever green, the perfection game

Reply

Scrum - no commitment, just a buzz-word

Jul 27, 2010 4:16 PM by **no scrum**

You wrote: "The team needs to ... be committed ...". What you mean by commitment here?

1. Commitment - is to do one's best in a sprint?
This may not be sufficient to complete the sprint successfully, because of some wrong assumptions during the planning, some unexpected problems like side effects of code changes from other teams.

2. Or commitment - is to really complete all the planned user stories, no matter what it costs? Even if it means overtime?

If you mean the 1st one, then it just means to be honest to the colleagues, to the employer, to the customers who will use your product. From the company management perspective, this is what every employee is expected to do and is paid for. No matter if you work by scrum or not. In this case only the one's best efforts are guarantied. There is no guarantee, that the plan will be completed till the sprint end.
If there's no guarantee, then what's the difference from the non-scrum projects?
If there's no guarantee, then such a "commitment" is worth nothing, "commitment" becomes just a buzz-word. Nothing more.

If you mean the 2nd one, then "commitment" means guarantied result. It is a real difference from many classical projects. It's kind of a small "fixed price" contract. But for the team it means some risks, mainly the risk to work overtime. To mitigate the risks:
- the team can explicitly reserve some time and commit to do less than one would expect
- the team can prefer quick and bad solutions for some user stories, if other stories took more time than planned. Reworking/refactoring this means additional efforts in future, which is not planned in the project budget.

So, with fixed week hours there is no real "commitment".
Either there is no guarantee that the work will be 100% completed till the end of the sprint.
Or the work will be 100% completed, but it will cost some reserved (and may be not used) time and/or worse quality, and thus make the work more expensive.
Neither is good.

So, real "commitment" is not compatible with the fixed working week, which most employees have. Thus:
- "commitment" in scrum is just a buzz-word
- scrum is not compatible with the fixed working week

Reply

not in the real world

Jul 27, 2010 5:11 PM by **phloid domino**

first, the 5 engineering aspects (unit testing, automated builds, etc) are absolutely the right thing to do, and still a hard sell in many organizations

promoters of 'agile' methods should focus on these practices and de-emphasize the 'behavioral' aspects

in the real world, that is, 90% of business environments where software is developed, the impediment to the behavioral recommendations is the same source of all software problems: management

the average manager's understanding of scrum or xp or any 'agile' method is limited to one thing: short deadlines

that's all managers care about; all they know how to do is look at a calendar

everything else will be subgrogated to the calendar

most agile methods have some element of adjusting based on feedback, but in the business environment it's still every man for himself, and nobody gets rewarded for telling the truth

so any process or method where developers are encourage to be honest will fail

i wish it were otherwise but that's the reality

Reply

Re: XP for dummies?

Jul 28, 2010 1:58 AM by **Prasad Prabhakaran**

May be not for you! I come from a world of IT services, where teams claims following agile may not follow 20% of what is said above.

Reply