

Challenges in Managing Software projects following Agile Scrum Methodology

Sanjib Ghosh (Member PMI)

Mercedes-Benz Research
and Development India, Bangalore



Leveraging project management for excellence, growth and transformation



Contents

1.1	Abstract	3
1.2	Key Words:	3
1.2.1	Agile:	3
1.2.2	Scrum:	3
1.3	Introduction	3
1.3.1	Teams from different geographical and cultural backgrounds:	5
1.3.2	“Help at hand”	5
1.3.3	Progressive elicitation of Requirements:	5
1.3.4	“Bug Slippage”:	6
1.3.5	Development Team and Testing Team working on different versions of the UseCase:	6
1.3.6	Rising Bug Curve:	7
1.3.7	Feature “Spill over”:	7
1.3.8	Movement of team member s across scrum teams:	8
1.3.9	“Performance” issue always comes later:	9
1.4	Conclusion	9
1.5	References	10
1.6	Author Profile	10



1.1 Abstract

Agile Scrum Methodology for Software Development is being used by many organizations. There are many challenges that are faced during managing such projects. This paper presents the challenges faced and the best practices that evolved during the life cycle of a live project which is being executed following the Agile Scrum methodology.

The project team consists of people from more than 7 different companies, and divided over two geographical locations with different cultural backgrounds and with a time zone difference of 3.5 hrs. These, coupled with the management trying to balance the expectations of end-user with the evolving Requirement Spec, makes this project a good case study for this topic.

With the specification being available just few weeks before the coding starts, getting the requirement freeze before start of coding was never achieved. This resulted in testing team and development team working on different versions of the Use Cases. When more and more users started using the first version on Productive environment, the performance issue cropped up and took priority over feature implementation. This resulted in few features being moved out of the plan which began to show badly of the company responsible for development.

This paper discusses the above and other challenges faced during the course of the project and how the management team handled them.

1.2 Key Words:

1.2.1 Agile:

Agile emphasizes building working software that people can get hands on with quickly, versus spending a lot of time writing specifications up front. Agile focuses on small, cross-functional teams empowered to make decisions, versus big hierarchies and compartmentalization by function, and Agile focuses on rapid iteration, with as much customer input along the way as possible. [1]

1.2.2 Scrum:

One of the fastest-growing Agile methods is Scrum. Scrum is an iterative, incremental framework. Scrum structures product development in cycles of work called Sprints. [2]

1.3 Introduction

This paper is a case study of a project that decided to follow agile scrum methodology and then found many practical challenges in doing so. The team found solutions for them from mutual discussion and over a period of approximately one and half years, the team has settled down and the results are showing.

The project is about development of a tool to integrate the entire life cycle of product testing (both car and components) and thus doing away with the various home-grown heterogeneous solutions from each department. The life cycle includes creation, maintenance and approval/release workflow of Test Cases (Test Library Module), planning the tests (Test Planning module), documenting the results of the tests (Test

Execution module) and evaluating the results (Test Evaluation module). The main responsible for development of such a tool is the IT Department of the parent Automotive Company (Daimler AG) situated in Germany. It took the services of many local suppliers for requirements management, GUI prototyping, architecture recommendations, testing etc. The main development task was given to it's subsidiary Mercedes-Benz Research and Development, Bangalore, India (MBRDI). This is a Rich Client application with EJBs holding the business logic.

The source of requirements (and the end users) of this application are the various engineering departments manufacturing different components (for component level testing) and the quality assurance department. As mentioned before, they are using different tools and techniques ranging from MS Excel spreadsheets with macros to small home grown tools. It was a mammoth task to bring all these parties together and arrive at a consensus for the requirements for the tool. This was done by the IT department of the parent company.

Needless to mention that it was not possible to finalize the detailed requirement for the entire application before starting the development of the tool. So it was decided to follow an iterative development life cycle which will be flexible enough to incorporate changes to the requirements. There was a need to work with the parent company as an "Extended team " and not as "Supplier" as discussions and clarifications were needed on a daily basis both with the parent company and the other suppliers involved in the project. Further, there was a need to have a overview of task remaining and effort available at any given point of time so that resources can be moved across different sub-teams, wherever possible, to achieve optimal utilization. And the methodology chosen was Agile Scrum methodology.

The MBRDI team has grown organically and now is 26 members strong. It includes 2 Project Managers, 5 Architects, 3 Manual Testers, 2 Automation Testers and rest client and server side developers. The "coding" Sprints are typically 4 weeks long with overlapping weeks between Sprints as shown in Figure 1. (SM stands for Spec Meeting with end users)

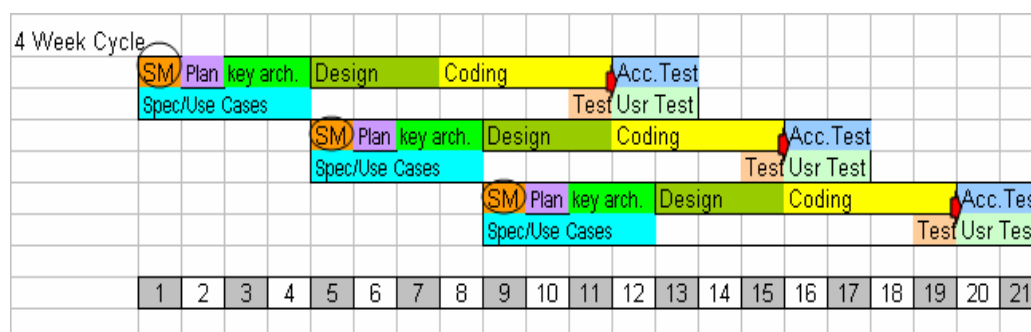


Figure 1 - Overlapping Sprint Cycles

Scrum Master(s) are located at the offshore development location in Bangalore. Product Owner(s) are located in Germany.

The following sections list down the different constraints present and / or challenges faced and the solution arrived at by the management:



1.3.1 Teams from different geographical and cultural backgrounds:

As there were resources from more than 7 companies involved from different cultural backgrounds and language proficiency, it was very much essential to form the basis of communication, e.g., the communication model and the communication language. It was more so important as MBRDI is working as an extended team and needed to communicate will all on a daily basis.

Solution: To declare the common language of communication and it was decided to be English. It meant that all project communication and documentation were to be done in English. Though it looks very trivial, but it was a very important stand taken by the parent company upfront. Else, there was every possibility of one landing up in a telecon / videocon where he / she cannot understand what is being discussed and be a mute spectator.

1.3.2 “Help at hand”

When the entire development team is sitting geographically very far, the client will be relieved if he is assured of help at hand in case of any need or clarification.

Solution: From the very beginning, one Project Manager and one Architect is always stationed at the parent company location in Germany to co-ordinate the project management, development and deployment activities. This is of great help as whenever there is a clarification sought by the parent company or any issue in the test or productive environment, they have a member of the development team at hand to resolve the same or get it resolved with the help of their offshore colleagues.

It is interesting to mention here that during one yearly feedback session, “onsite presence of MBRDI colleagues” got the highest “vote” when all the involved parties were asked to write down 3 things they think are most helpful and want to continue the following year.


1.3.3 Progressive elicitation of Requirements:

The end users from various engineering departments had agreed to some high level requirements. But these were not detailed enough either for the architects to design the application or for the testers to write test cases or for the developers to code. Hence, it was decided that a Specification Team will be formed which will take care of translating the end users’ wishes to Use Cases.

As most of the end users are comfortable in German and the development team comfortable in English, working knowledge of both German and English was a precondition for the Spec Team members. The Spec team provided the UseCases for the features to be implemented in the coming Sprint before the start of the Sprint.

But gradually the team found that this itself was not enough. Once the Use Cases were delivered and the team started working on them, there were so many clarification questions. By the time all the clarifications are answered, the team was well into 2nd (and sometimes 3rd) week of coding Sprint.

Solution: Spec Team was asked to deliver the UseCases for the upcoming Sprint at least 4 weeks before the start of the coding week. At the start of the design phase, the Spec Team in Germany hold a telecon or videocon with screen sharing and explain the concept behind the UseCases / Features. Usage of State Diagrams and Sequence diagrams was found to be very helpful and has now become part of the UseCase



deliverables from Spec Team. The architects now go through the UseCases in detail and ask clarification questions. (They are added in the same UseCase document as “Questions”). Accordingly, the UseCase is modified. “Track change” mode is kept ON to have the history of all such changes.

Our general observation is that the Spec Team is very good in writing the “Normal flows” for any scenario, but they miss out the different “Alternate flows” that are possible for this scenario. Interestingly, the Testing Team is the one who are very good in pointing out all the missing “Alternate flows” in the UseCase document.

1.3.4 “Bug Slippage”:

The number of bugs that were not found by internal testing team, but were found during acceptance testing, was a matter of concern. A careful study of these “slipped” bugs and discussion with the testing team revealed the root cause. As the UseCase clarifications went late into the Sprint, the planned internal releases to the testing team used to get delayed. Thus the testing team got just enough time to execute all the test cases, but had no time to do “monkey testing”. Most of the “slipped” bugs could have been found by internal testing team, had the releases to them been on time.

Solution: The solution here was pretty straight forward. Scope team needed to give the UseCases on time and the architects and developers need to get all the doubts clarified with first week. This ensured that there was no delay in internal releases and the testing team got enough time to do a thorough testing.

1.3.5 Development Team and Testing Team working on different versions of the UseCase:

Once the UseCase clarification is done with the Architects, the UseCases are discussed with the development and testing team. This is typically done either during the first one or two days of the coding sprint or the last week of the previous sprint. Interestingly, there are some scenarios and questions asked by the development and testing team which did not come out from the Architects. So once again there is an update to the UseCases by the Spec team.

This gives rise to a very difficult scenario. Suppose on day 2 of the coding week the testing team has started writing test cases for UseCase UC-1 and finishes the same on day 4. The tester then goes ahead with writing test cases for the next UseCase. On day 5, the UseCase UC-1 gets modified due to a clarification sought by a developer. So the testing team and the development team are now working on different versions of the same UseCase UC-1. Needless to mention, when the testing release is given to the testing team, they raise a bug for a correctly implemented functionality as the test case is now stale.

Solution: Our experience says that valid clarifications sought by development team and testing team cannot be avoided. Pushing these changes to the next Sprint release, in most cases, will be an overkill. Hence the solution arrived at was to freeze the UseCases at the beginning of second coding week. A tag is created in the configuration management tool and both development and testing team work on the version checked out from this tag.

However, if still some changes to the UseCase(s) cannot be avoided (and possible to implement in the current Sprint as decided by architects), a new version of the UseCase with the changes is checked in, the tag moved and both the development team and the testing team are informed of the same.



1.3.6 Rising Bug Curve:

The development team primarily focuses on the features as planned in the Product backlog. The only time they can work on bugs is during the last week of the Sprint when the internal testing team raises bugs during IVV releases. Then also, they focus on the critical / showstopper bugs from the current implemented features. The focus on the existing bugs gets lost and this leads to a rising overall bug curve.

Solution: Two separate Scrum teams were formed – Feature Scrum Team and Bug Fixing Scrum team. The task of the Bug Fixing Scrum team is to look at the Bug Tracking tool and pick up the bugs that are important to the “end-users” (and it varies from the criticality given by testers), pick up some “low hanging fruits” (bugs that can be solved with very less effort) during the Sprint planning meeting and work on them during the Sprint.

A more effective way followed to tackle this (and also increase the general stability of the product) is to have a “Consolidation” Sprint after every 3rd Sprint. A consolidation Sprint does not have any Feature assigned to it. Only bug fixes are done during this Sprint. This is also a convenient place to “catch” any spill over feature from previous three Sprints and implement them. Further, any “urgent” and unplanned Change Request coming in between can also be accommodated in this Sprint.

1.3.7 Feature “Spill over”:

In Steering Committee meetings involving department heads, IT Department of the parent company needs to mention the features that they are going to deliver to the Engineering Departments during the course of the year. To arrive at this list of features, estimates for developing and testing them are needed. But as the UseCases are written and delivered progressively, the offshore development team does not have the details available to estimate for the features to be implemented during the course of the year. Still, they need to provide this input for the Steering Committee meeting. So these estimates are provided based on a very short description of the Feature / UseCase (sometimes in single sentence) and based on that the Product Backlog for the coming year is made.

Our experience for 2010 was that for almost 60% of the UseCases, the final estimates (after going through the detailed UseCase) were more than 160% of initial estimates. This resulted in “spillover” of the UseCases from one Sprint to another and finally out of scope for 2010 (even after “catching” few in the consolidation phase).

This boils down to the IT Department not being able to deliver the promised features to the Engineering Departments, which, in turn, gives the message that the development team could not implement in time. (This conclusion is not fully correct as the basis for this is the estimates given with one line description of UseCases.) When escalated, this has a very high negative impact on the team morale.

Solution: For 2011, we are providing the estimates now. But we are better prepared. The Specification Team has provided us with “Raw concepts” for the “planned” 2011 features. These include state diagrams, proposed UIs etc. besides a very short description of the UseCase. With these “pictorial” documents, the Spec Team and the Architects have a 2 hour discussion. Following day, the architects estimate using “planning poker” methodology. 20% buffer is added for all the new scenarios that will come up when the final UseCases are delivered.

For the team morale, one very important and effective step taken was to mention clearly in all forums that the Spill over is a short coming of the entire project team (i.e. IT Department of parent company, Spec Team, Development team etc.) and not of the development team alone.

1.3.8 Movement of team member s across scrum teams:

As mentioned before, there are more than one scrum team (Feature Scrum and Bug Fixing Scrum). And it is sometimes necessary to re allocate resources from one Scrum to another (sometimes partially) during the Sprint. Also, the availability of team members (both planned and unplanned leaves) needed to be captured and reflected in the burn down chart.

Solution: An “Overall Scrum sheet” was created. The first step during Sprint planning is to update the availability of each resource in this sheet and then allocate him/her to one or more scrum teams, as shown in Figure 2. The availability column of the resources in individual scrum sheet has reference to this overall sheet. So re allocating a resource (fully or partially) from one scrum team to another can be done just by changing the allocation in the overall sheet for this resource.

		S25	38	39	40	41	42	S26	S25	S25	S25	S26	S25
Person	Role		Sept.			Oct.							
		Velocity						Availability Total [h]	BF-CR Scrum	Feature Scrum	Arch. Scrum	Planned Total [h]	Utilisation
abc	Performance A	100	5	5	5	4	4	184	120		64	184	100.0%
def	Architect	100	4	5	5	4	4	176	126	25	25	176	100.0%
ghi	Architect	70	3.5	3.5	2.8	0	0	78.4	20	38.4	20	78.4	100.0%
jkl	Architect	70	3.5	3.5	3.5	3.5	3.5	140			140	140	100.0%
mno	Architect	100	5	5	5	4	4	184		140		184	100.0%
pqr	Developer S	100	4	5	5	4	4	176	40	136		176	100.0%
stu	Developer C	100	5	5	5	4	4	184		184		184	100.0%
vwx	Developer C	100	5	5	5	4	4	184		184		184	100.0%
yz	Developer C	100	5	5	5	4	4	184		184		184	100.0%
bcd	Performance C	100	5	5	5	4	4	184	184			184	100.0%
efg	Developer C	100	5	5	5	4	4	184		184		184	100.0%
hij	Developer C	100	5	5	5	0	1	128	40	88		128	100.0%
klm	Developer S	100	5	5	5	4	4	184	50	134	0	184	100.0%
nop	Build Mgmt	70	3.5	3.5	3.5	3.5	3.5	140	120		20	140	100.0%
qrs	Developer S	100	5	5	5	4	4	184		184		184	100.0%
tuv	Performance S	100	5	5	5	4	4	184	184			184	100.0%
wxy	Developer S	50	2.5	2.5	2.5	2	2	92		92		92	100.0%
zab	Developer C	100	5	5	5	4	4	184	184			184	100.0%
cde	Build Mgmt	50	2.5	2.5	2.5	2	2	92	92			92	100.0%
fgh	Test	100	5	5	5	4	4	184		184		184	100.0%
ijk	Test	100	5	5	5	4	4	184	52	132		184	100.0%
lmn	Test	100	5	5	5	4	4	184	52	132		184	100.0%
opq	Auto Test	100	5	5	5	4	4	184	184			184	100.0%
rst	Auto Test	100	5	5	5	4	4	184	184			184	100.0%
uvw	Quality	30	1.5	1.5	1.2	0	0	33.6		33.6		33.6	100.0%
Total			38	39	40	41	42	4000	1632	2055	313	4000	100.0%
	Build Mgmt		6	6	6	5.5	5.5						
	Architect		16	17	16.3	11.5	11.5						
	Quality		1.5	1.5	1.2	0	0						
	Test		15	15	15	12	12						
	Developer C		30	30	30	20	21						
	Auto Test		10	10	10	8	8						
	Developer S		16.5	17.5	17.5	14	14						
	Performance A		5	5	5	4	4						
	Performance C		5	5	5	4	4						
	Performance S		5	5	5	4	4						
			Coding Sprint 25										
	Build Mgmt		29										
	Architect		72.3										
	Quality		4.2										
	Test		69										
	Developer C		131										
	Auto Test		46										
	Developer S		79.5										
	Performance A		23										
	Performance C		23										
	Performance S		23										
	Summe		500										

Figure 2 - Overall sheet (with availability and scrum allocation)

This means the baseline effort for each scrum sheet can also change during the course of the Sprint. To keep track of that, we have added another line called “original baseline” (as shown in Figure 3) which is nothing but the baseline availability at the start of the Sprint and it remains constant.

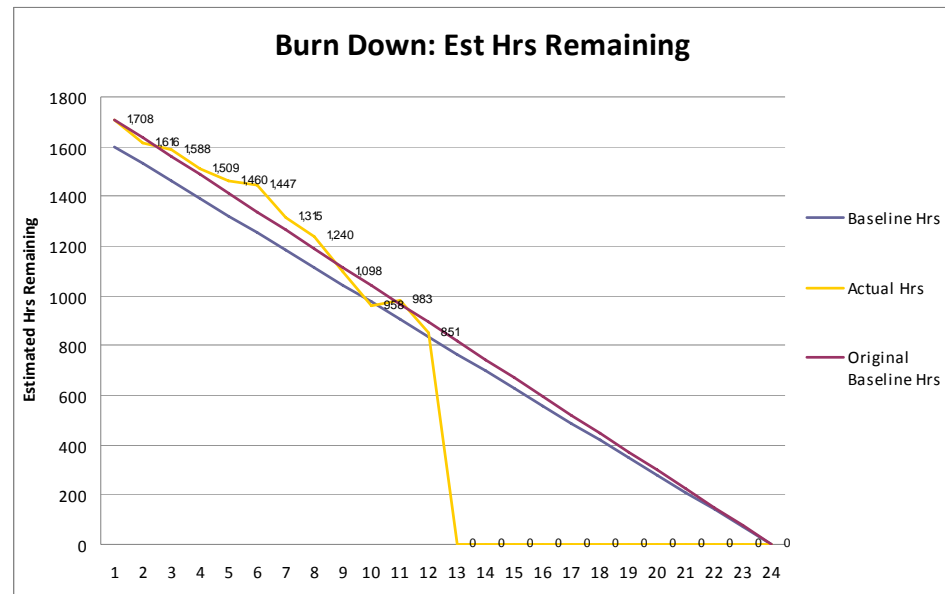


Figure 3 – Burn Down with Original and current baseline hours

1.3.9 “Performance” issue always comes later:


In the initial few Sprints, when the number of users is not very high and also the data to be handled is not huge, everything seems to work fine. As more and more users start using the tool and more real time data gets loaded, the end users start complaining about performance.

As all the requirements do not come in the beginning, it is sometimes not so easy for the architects to decide on some finer issues effecting performance. Further, the focus is always on delivering features followed by reducing bug count. Till there is no complaint from any quarter, no one bothers about the performance issue.

Solution: A dedicated team of 3 (One Architect, One serve side developer and One client side developer), work exclusively on the Performance Issues. Just like target for features and bugs, the target for performance for different operations are also set and tracked after each Sprint release.

1.4 Conclusion

Agile Scrum methodology, one first sight, looks the perfect solution for managing software projects where the requirements are evolving and quicker end user feedback is necessary. But in practical scenario with lots of constraints, deviation from the copy-book version of Agile Scrum is sometimes necessary. We may lose out on some of the benefits of “pure” agile scrum, but the project as a whole gets benefited. In this case study some such practical scenarios have been cited which are not the ideal cases (e.g. UseCase change during Sprint etc.). We need to adapt our methodology to these situations rather than saying “No”.



This methodology seeks more involvement of the different stakeholders, especially end-users and parent IT company (which out sources the development). Hence, the fruits of this endeavor also have to be shared by all parties – be it sour or sweet. Without this ground rule, this methodology has very less chances of a happy ending.

1.5 References

1. Pete Deemer, Gabrielle Benefield, “The Scrum Primer – An Introduction to Agile Project Management with Scrum”
2. Pete Deemer, Gabrielle Benefield, “The Scrum Primer – An Introduction to Agile Project Management with Scrum”

1.6 Author Profile



Sanjib Ghosh has completed his B. Tech. from Indian School of Mines, Dhanbad and later Post Graduation in IT from Indian Institute of Information Technology, Bangalore. He is working with Mercedes-Benz Research and Development India (MBRDI) since 2002. His areas of interest include, among others P2P technologies, Prognostics and Software Engineering. Currently he is working as a Program Manager. Before joining MBRDI, Sanjib has worked with Telco Construction Equipment Company Limited (Telcon) in Marketing Division.

E-mail: sanjibdcx@gmail.com
sanjib.ghosh@daimler.com