When your 'Agile' Team Moves at Snail Pace: 5 Key Roadblocks and How to Overcome Them

This item in chinese

Having worked as a Trainer and Manager for Agile projects over the last decade, I find myself agreeing more and more to a quote I read at the start of my career:

> *We've gone from one dogma [waterfall] to another dogma [scrum and XP] -- I thought the whole point was that we were supposed to be agile. What happened to thinking? – notes from Brendan's Braindump.*

While most software engineering teams – especially lean development units – swear by Agile process frameworks like Scrum and Kanban, there is often a big gap between theory and practice. This is especially true when Agile is a relatively new concept for an organization, or for new team members.

Agile approaches can definitely help address persistent IT concerns: lack of team effectiveness, spiraling time & cost of development, and difficulty of accommodating changes. However, it is definitely not a silver bullet: I've seen many agile teams under-deliver massively, and the following five are the most common reasons I have encountered.

Resistance to agile

One sure-shot recipe for disaster is trying to shove agile processes down the throat of an unconvinced team. Many large organizations simply hire an agile coach, train the team for Scrum or other process and hope that all will be well. But habit is a hard drug to give up. Talking about resistance to Agile, Mountain Goat hits the nail on the head:

> *Some may resist because they are comfortable with their current work and colleagues... Still others may resist due to a genuine dislike or distrust of the Scrum approach. They may be convinced that building complex products iteratively without significant up-front design will lead to disaster.*

***Real life Story**: A product team I was working with used poker to assign priority to a backlog. But any word from the managers would still be taken as a line written in stone, defeating the effort to take a team decision. While the managers, developers and testers understood the theory behind the process, the habit of deference was too difficult to shake. The leaders did not want to lose their influence and chose not to adhere to many key parts of the process.*

This meant that while we were following 'agile' processes on paper, our actions were anchored in old ways of thinking. And this way of working was comfortable for the leaders as well as many of the team members. Needless to say, the team did not reap any real benefits from the process.

How to overcome this: In order to succeed with Agile, it is important to note who is losing and who is gaining clout. And you need to ensure that the whole team can benefit from the process. This requires convincing all the stakeholders, demonstrating and listing clear benefits from the process, listening to any feedbacks, and, in extreme cases, segregating obstinate naysayers and reducing their ability to negatively influence the team dynamics.

Inability to juggle changing priorities

Most Scrum Masters and Project Managers are under pressure to deliver big with small teams. As there is no possibility of deviating from the sprint, any unexpected requirement can cause severe problems. In case you have limited resources and are not able to get more people on the project, any urgent requirements can force you off the track.

For too many teams, staying on the track while accepting newer requirements can be an almost impossible task. Consequently, the team fails many times to deliver what was promised during the iteration planning.

Sure, you can say no to the ad hoc requirements that assume priority mid-sprint. But that is not always possible. In such a situation, you may have to remove some requirements from existing sprint to make room for the new ones. In other cases, you will be forced to cancel the sprint and focus on urgent requirements.

Real life Story: *For a project I worked on a few years back, the client wanted a minimalistic login form, which would allow the customers to login simply through their email. The team created a design, which got approved. Thereafter, in the next sprint when the backend was being developed, the client decided to add more parameters to the form to get more data on the users.*

This put a spanner in the works. The scrum master quickly reshuffled the resources and changed the priority for the sprint to focus on delivering the design first, getting it approved and then using the rest of the time for backend development. As the changes were relatively simpler, the changing priorities didn't affect the process too badly. But bigger changes that affect subsequent sprints can be tougher to handle.

How to overcome this: The best way to avoid this situation is to have a trained and experienced Scrum Master at helm, who is able to visualize and predict future requirements, prioritize effectively, and create an optimized plan for product engineering.

At the same time, knowledge of various programming practices, and which ones can be applied to resolve a specific problem will help tackle the unexpected issues. This is where training from a seasoned agile coach can be of immense help. The more experienced a scrum master is, the easier it is to find a way to resolve the problems thrown by changing priorities.

Unmanageable defect backlogs

Agile's focus on continuous delivery & deployment forces teams to ship fast, and delivery often takes precedence over quality. It is not uncommon to find a list of bugs and defects pile up in form a huge, intimidating backlog. Over time, an inordinately large portion of the team's effort may go into addressing the backlog, leaving too little time and energy to focus on new design or development.

Too many teams, especially the ones new to agile, end up in an untenable position where a large portion of the entire development effort is focused on addressing the backlog. In the worst cases, the process of building software can feel like driving a car with the handbrake on.

Real life Story*: One of our teams had been working on a rather straightforward project with clear requirements. However, when the client rolled out parts of the product to a larger user base, entirely unexpected conceptual defects with initial features were detected, which played havoc with ongoing sprints.*

The product manager quickly recalibrated the percentage of time allotted to defect clearance and change management, taking into account the frequency of client's thought variation, the team velocity and our quality benchmarks. By allotting 20% time to defect backlog and adding an extra resource for a few sprints, and by utilizing pair programming to speed up the process, we were able to weather the storm and stop things from spiraling out of control.

How to overcome it: There is no easy answer – the team simply has to generate fewer defects and get better at resolving them. For practical purposes, putting 10–15% (or the percentage that suits your project) of development efforts for resolving the bugs found during testing is a good way of keeping the backlog from getting too large.

A skilled Scrum master with the ability to rapidly prioritize the defects and utilize the right Extreme Programming and Test Driven Development methods will play a key role in the success of the project.

Delays in building MVP

While working towards an MVP, many agile teams end up carrying out UI Design and Backend activities in parallel – without building a bridge between the two. Let's say that the product development time estimate is 9 months. In such a situation, there may be as much as 4 months of parallel development of UI and backend (without building a connection between the two). In such a scenario, changes resulting from first viewing of the unified product have far-reaching impact, which often leads to widespread modifications.

Real life Story: *Working on a mobile and web application, one team I worked with started with designing screen first. Once the PSDs or JPEGs had been approved, the middle layers, DB and business logic elements were tackled. However, as the interactions on the application were visible only after the design had been approved, the client usually came back to change the designs, throwing the entire process out of whack.*

The scrum master was able to simplify and streamline the entire process by focusing on building entire interactions together. So, the design, development and interactions for every process were built in tandem to deliver a working module or feature to the client. This enabled the client to get a clearer picture right from start, thereby reducing the number of last-minute changes.

How to overcome it: Only when all the parts of development – design, JPGs, HTML, code, middle layers and DB – are joined together, can the product owner get a clear view of an actual useable product. Teams can deliver a functional, interactive product each sprint. The product is built layer by layer, giving a clearer picture to the product owner with every iteration. This helps deliver the MVP on time, reducing major changes in later stages of development.

Lack of agile education and communication gap

When the Product Owner isn't able to communicate with large teams on a regular basis, or when he or she doesn't understand agile processes, problems tend to crop up.

Real life Story: *For instance, one team I worked with ran into several delays simply because the product owner was not able to provide timely inputs on sprints every week.*

On another project, there was no single product owner. The role was divided between a technical head and a functional head – who had different priorities and outlooks. The two heads were removed from the day-to-day work of the teams and often sent conflicting requirements. This created a big problem with prioritization and made life difficult for the developers.

How to overcome it: While taking decisions together with inputs from the entire team and stakeholders is important, it is equally important to have a place where the buck stops. An individual who has a clear and comprehensive vision of the product, and who can take the final decision, is needed in most cases to avoid fragmented approach to development.

At the same time, it is essential to educate the product owners and executives into how agile processes work, and what role they must play for it to succeed. For this purpose, hiring an experienced agile coach can be a profitable decision.

Wrapping up

Of course, there are many other hurdles you will need to overcome if you want to reap the benefit of any agile process. But the above are some of the most common and avoidable errors that still plague many teams, simply because organizations are not able to change their habits and thinking processes.

What are your pet peeves about Agile? What is hampering the agility of your software product development process? Please share your insights and experiences with us in the comments.

About the Author

Bhoomi Mehta is an IT Professional with proven cross functional expertise at Cygnet Infotech - Consulting, Operations, Marketing & Sales, and Application Portfolio Management. She focuses her energies on motivating agile and lean teams to perform their best in their respective functions. With focused approach towards delivering success across entire Product engineering Services Lifecycle. Bhoomi's motivation is based on one clear idea: “Teams are not good or bad. The successful outcome of a project / assignment is directly proportional to the mentoring, tools and trust you are able to provide.”

Popular in Culture & Methods

- **Domain Driven Design Quickly**
- **The C4 Model for Software Architecture**
- **Google Software Engineering Culture**
- **Evolutionary Architecture**
- **The 4 Questions of a Retrospective and Why They Work**

3