Impediments to Enterprise Agile

by Kane Mar

Once introduced into an organization Agile software development practices can spread quicker than can be controlled. Without a strategy Agile projects can emerge with different or even conflicting approaches. This is not necessarily a bad thing, but a traditionally top-down management structure can be uncomfortable adjusting to the more egalitarian Agile approach. There will be concerns over introducing Agile practices in a consistent manner without causing undue risk.

**Impediments: People, Process and Technology**

In the paper *Enterprise Strategy for Introducing Agile*, I discussed an overall road-map for introducing Agile methodologies into an enterprise followed by a plan. That paper only considered an idealized scenario, one in which there was a logical progression and there was little (if any) objection. This is seldom the case.

There is often very strong resistance to the introduction of Agile methodologies from many quarters. This resistance may be direct, or it may be indirect. Ultimately however it is related to the organization’s or individual’s perceived threat to their position, authority or compensation (i.e., power and money).

This article discusses some of the impediments that may be faced by a transition to Agile methods.

**What Enterprises will successfully adopt Agile?**

This question is asked a lot, but in a different format. The question that is asked is usually “What types of projects are most suitable for applying Agile practices?” Although these appear to be quite different questions they not because ultimately it’s not the type of project that determines Agile success, but rather the organizational and cultural constraints of the enterprise.

The following quotation from Tom Poppendieck nicely sums this up. I’ve added the emphasis in order to drive the point home.

“*The constraints on when agile approaches can be used are primarily organizational and cultural, not project types.* Some organizations and some contexts are incompatible with agile/lean thinking. When these organizations eventually come up against a strong competitive threat, they will fail to meet it unless they change their values and mindsets. Lean/Agile is at its foundation, the fourth industrial paradigm, the first being Craft Production, Factory Production with machine tooling, Automation and Taylorism. These
come along every hundred years or so and take a few decades to work through. Each paradigm includes the preceding one and makes it dramatically more productive.

There is no need to sell agile except to organizations that want to survive long term. If they don’t see the threat/opportunity they cannot succeed with agile or lean nor can they sustain economic viability in the long run.”

-Tom Poppendieck

The main traits that I look for when I work with an organization are a willingness to listen, and to change accordingly. Many organizations will insist that they listen to their employees and that they have an agile culture. Although this may be true to a certain extent, the real test always comes when the Agile values conflict with the organizational values.

If the organization does a lot of up-front analysis and design, what is the response when you ask them to write User Stories? Do they insist that Use Cases are better than User Stories, or do they ask you to show them how? If the organization traditionally provides individual offices for their developers, what is the response when you suggest common team rooms? Does the organization try to justify their current status quo, or do they experiment with alternatives?

Provided that an organization is willing to listen and change then all these issues will be eventually resolved; it will only be a question of time and effort.

People, Process, Technology, Teams, Management and Culture

When initially introducing Agile practices to a team, the difficulties experienced by the team are all centered around the immediate adoption of the practices, and the consequences of that adoption. After some experimentation with Agile methods the focus becomes centered on larger problems that confront an entire team. Once experienced with Agile practices the problems are large still and will confront may teams. The scope and nature of the problems faced by an Agile team grow over time as knowledge of Agile practices spread throughout the organization in ever increasing circles of influence.

In an effort to discuss the issues faced by Agile teams, I’ve broken down these issues into the following categories: People, Process, Technology, Teams, Management and Culture.

People

Organizations will have people issues with or without introducing an Agile methodology. But I think it’s fair to say that the introduction of Agile methods increases the rate of change, and expose any problems that might have been previously hidden beneath the surface.

I feel that any solutions to people problems should be based around fostering good clear communication between team members, and should emphasize team work over individual heroism. The following are typical situations that are frequently seen in Agile projects:
• **Overly specialized skill-sets** requiring work to be handed-off several times before it can be described as complete. For example, an Analyst will describe the problem before handing-off to a DBA, who will work with the database before handing-off to a developer, who write the functionality before handing-off to a Tester. The act of handing-off partially completed work is very expensive, and it is desirable to keep this to a minimum.

An approach to overcoming this is to ensure that all hand-offs [ideal there will be no hand-offs but reality often dictates otherwise] are informal and to encourage pairing at all stages. Encouraging the team to work in small chunks of work also helps in this regards as it forces the team to meet and communicate more frequently.

• **Lack of ownership by the team.** This may be a problem at the start of an Agile project. Team members who are use to being handed instructions often have difficulty engaging with the project. Their learnt reaction is to wait until being told.

In order to work like a team and to take ownership of the problem, teams need to be able to communicate in an open fashion. This can be encouraged by letting the team work directly with the customer, by ensuring that technical conversations are not shut down and by soliciting input from different members of the team.

• Architects, Developers, Testers, etc **refuse to interact with the team.** It is not uncommon for some team members to refuse to participate in the Agile process. They may have many justifications for this … they have to many prior commitments, their work cannot be broken down in a fashion that’s amenable to Agile SW development, their time is too important, they work better as an individual etc. These are nothing but excuses. Regardless of the reason, the end result is that they are deliberately putting themselves outside the team.

The team has to make it’s best effort to resolve these issue and to work with others who are attempting to separate themselves. However, every member of the team needs to contribute and if one or two members of the team are putting themselves above the team then this needs to be quickly address by management, usually by the quick remove of the individual(s).

• **Senior Management giving mixed signals** regarding support for Agile. Nothing is more efficient at crippling an Agile project than a few carelessly chosen words from Senior Management. Senior management need to decide prior to introducing Agile methods if this is something that they are committed to. Regardless of their own personal perspective, once an organization has decided to adopt an Agile methodology senior management have an obligation to publicly support the effort.

Agile teams and customer must also realise that they play an important part because unless they communicate their experiences to the wider organization, their successes will
remain unacknowledged. It’s important for Agile teams to communicate frequently with senior management, to clearly articulate their successes, and any obstacles to their success.

**Process**

One of the common attributes for all Agile methods is that they are very process-light. Scrum specifically is often described as more of a framework than a process. What Agile methods explicitly acknowledge is that we don’t need complex processes to build software. In fact, the opposite is true.

But for something that is conceptually simple, a lot of projects have difficulty trying to keep it simple. Here are some commons process related problems:

- **No single customer (Product Owner) can be identified.** This is common a problem with there is a large separation between the business units and the software development departments, or in situations where several different groups have an equal interest in the success of a project.

  In my experience, most business customers really want their projects to be successful but they lack the understanding of how they can constructively interact with software projects. Methodologies such as Waterfall or RUP have reacted to negative customer involvement by trying to distance the customer from the development teams. Agile on the other hand seeks to integrate the customer, but with a better understanding of the interaction.

  In the situation where a single customer cannot be identified, it’s really necessary to identify the project sponsor … the single person who ultimately approves the funding of the project. By working with the sponsor and clearly articulating the need for a single business representative on the project team this situation can usually be quickly resolved.

  In the situation where several different groups have an equal interest in the success of a project, the team still needs a single representative who is willing to work with each of the different groups and prioritize accordingly. Again, working with the project sponsor will provide the path to a solution.

- **Management wants to combine elements from both RUP and Agile.** The end result of doing this will have the disadvantages of both methodologies without the advantages of either. There are several different approaches to doing this; RUP management process combined with XP engineering practices being the most common.

  The motivation for doing this is often a desire to continue to provide familiar reports without having to re-educate senior management or business partners. I personally feel that this approach under estimates senior management, and does the organization a
disservice. Better results can be had by educating management and business partners while adopting a single approach whether it be RUP or Agile.

- **Scrum Master refusing to protect the team.** The primary function of the Scrum Master (or Iteration Manager) is simply to protect the team and to remove impediments. It takes a lot of courage to report problems to senior management and to make changes that others in the organization may view as a direct threat to their position. At the very best most SM will [at some point] have their credibility questioned.

Without an effective SM, however, who will protect the team? Who will standup to those who would distract the team with unrelated activities, or to the Product Owner to say that their expectations are unreasonable?

The SM role is essential to maintaining health communications between the team members and to protect against distracting external influences. If the team isn’t getting the help they need, then perhaps the firstly problem that needs to be resolved is a more effective SM.

**Technology**

Process practices are common to all Agile methods, but engineering practices are not. Extreme Programming (XP) is different in that it explicitly prescribes engineering practices in addition to process practices. But why would specific engineering practices be of interest and how is this different from other (Waterfall, RUP) methodologies?

The heart of the answer lies with reducing the cost of producing a usable product on a frequent and repeatable basis. Iterations are common to all Agile methods, and the outcome of those iterations is typically a usable product. It takes some considerable overhead to build, test and package a product. In order to be able to do this repeatedly (every iteration) in a cost effective manner requires a significant amount of automation, hence Agile engineering practices.

Processes which either increase the cost [of producing a usable product] per iteration, or do not actively reduce the cost per iteration should be immediately eliminated. Examples of wasteful practices are listed below:

- **Not having a reliable build system and processes.** The source control, build and testing of software is an intimate part of any software development team. It simply makes sense to have this process as efficient as possible. If it takes 2 hours for a developer to check in code and do a build then automating the process is going to allow more constructive use of that time.

  Clearly, if the team is building and releasing code many times per iteration then any effort expended to create a reliable build and testing framework is going to be well worthwhile.
Not addressing QA issues. With more traditional development methodologies testing is usually addressed at the end of the cycle. So, it’s not surprising to see teams take up Agile and then not pay appropriate attention to defects and code quality. Code quality will quickly become a problem with repeated iterations; especially if the team is trying to build new functionality on top of defect laden code. In addition, delaying testing until the later stages of a project will result in late discovery of unanticipated side-effects.

Producing high quality code from the very onset should be the ultimate goal of every project. It is only by addressing quality issues early and proactively that the fully nature of the software will be known.

Ineffective tools mandates by a committee or external parties. Large organizations often feel the need to control the number and types of tools that are used by project teams. This can be for a number of very good reasons such as licensing, IP and IP rights management, and security.

But when the authorization process or the tools themselves act as an impediment to the team making progress, something needs to be done. If a tool doesn’t meet developer needs, then why should they use it? If the authorization process for introducing CruiseControl takes more than a few weeks, then what does this say about the organizations attitude towards software quality?

At this point the SM and project team need to make their point of view known to senior management. They need to explain why tools that are relevant to the work that they’re doing should be supported, and that having the right tools can make the difference between a product and a high quality product.

Why all the discussion on Agile impediments?
Dealing with issues, problems and difficulties is part of everyday work life and there is nothing new in Agile methodologies which tells us how to deal with these problems. So why all the discussion on Agile impediments?

For me than answer is that Agile methods make visible those problems that were previously ignore, hidden or otherwise invisible to the organization. It is only by actively acknowledging the dysfunction that teams and organizations have that we are able to address them. And it’s only by discussing these problems that we’re able to share information, to help us understand that many of the problems we face today are shared by individuals in software organizations all over the world.

Impediments: Teams, Management and Culture
When initially introducing Agile practices to a team, the difficulties experienced by the team are all centered on the immediate adoption of the practices, and the consequences of that adoption. After some experimentation with Agile methods the focus becomes centered on larger problems that confront an entire team. Once experienced with Agile practices the problems are larger still.
and will confront many teams. The scope and nature of the problems faced by an Agile team grow over time as knowledge of Agile practices spread throughout the organization in ever increasing circles of influence.

In an effort to discuss the issues faced by Agile teams, I’ve broken down these issues into the following categories: People, Process, Technology, Teams, Management and Culture. The final segment of this paper discusses team, management and cultural impediments faced by teams transitioning to an Agile model.

**Teams**

On the whole, teams seldom present impediments to Agile adoption. Most Agile frameworks strongly advocate close, cohesive team work and the result is typically advantageous to team environments. The learning process that the team needs to undergo may be very slow [at least initially], but I personally won’t consider that to be an impediment. I simply view this as being the nature of change.

So, if there is going to be any friction, it is likely to be between different teams than within a particular team. There is the possibility of friction in any situation when one team is dependant upon another or where two different teams are using different methodologies (e.g., Agile vs. RUP). In both these situations the standard approach is to prioritize dependant functionality early and code to an agreed upon interface ... but this is overly simplistic. I feel that it’s important to recognize that there is a dynamic relationship between dependant teams which needs to be actively, and continuously worked. Failing to acknowledge the risks involved in dependant projects can quickly lead to confusion and impact the team performance.

**Management**

>“In the culture of management, the worst thing you can do is admit to anyone that you have a problem you can’t handle by yourself. If you really do need help, you have to sneak it in somehow without admitting in public that there is any problem at all.”

- Gerald Weinberg

Educating a new client on how to build an Agile organization can be very slow and difficult work. In order to be successful you need to be able to speak persuasively at many different levels of the organization. Speaking to the team or to individual members of the team is the usual approach because this is the level into which most consultants are brought. But there is also a wider audience to consider which includes functional managers, the PMO and HR. Failure to address this wider audience can hobble the transition to Agile (see below).

There is an assumption that management are able to see the changes for themselves and that they will automatically understand the value of Agile methods. I’ve personally found management to no different from everyone else and that they also require education, coaxing and convincing that there is a better way to develop software. The only difference is that the topics of conversation
are different for management than they are for developers: instead of discussing TDD, discuss Agile metrics (burn-down graphs over actual developer hours); instead of discussing Pair Programming, discuss the need for collocation.

When working with management, I’ve found that I’ve needed to pay special attention to both Agile metrics (or the lack, there-of), and adaptive planning over predictive planning. The Agile approach to both of these is counter intuitive for most classically trained managers and requires constant reinforcing.

Company Culture
As I’ve mentioned in previous articles, cultural changes are the most difficult to make within an organization. There will constantly be the argument “That’s not how we do things here”, or “We can’t implement that here”. These are “Yes but ...” arguments as Ken Schwaber points out in his course. There are no easy answers, especially for something that’s as intangible as company culture.

“... logic and culture have nothing to do with one another.” - Gerald Weinberger

Cultural specific impediments are often related to how an organization rewards it’s employees, specifically the compensation, promotion and career planning models. In an environment where performance is determined by specialization of knowledge, the promotions and compensation models reward the compartmentalization of knowledge. Over a period of time (several years) this results in the organization having two (and maybe more) attributes: Certain activities are performed by specific individuals; and, the organization is managed as a matrix.

The first is immediately obvious. If Alice is rewarded for her understanding of the security system, for example, then she is likely to continue doing this until the reward mechanism is changed.

The second point is not nearly as obvious but is a reaction to increased specialization. In order to try and help circulate knowledge and information, groups are formed where the individuals share a common function. Typically this results in an organization where there are groups for Analysts, Architects, Developers, and Testers etc. Project teams are then composed by selection individuals from each of the different groups. Ironically, this functional grouping of people serves to in further segregate of information. Agile methodologies breakdown these arbitrary boundaries on a project-by-project basis by encouraging cross functional teams. Long term solutions are dependant on rewarding teamwork and breadth of understanding.

This is what you should not do ...
When I was a child one of my favorite books was “The Bike Lesson” where the common refrain was always “This is what you should not do.” In the same spirit I offer the following example of what not to do. As an example of failing to address all relevant parties in an Agile transition, I’d like to use an actual project of mine from several years ago.
A small team of experienced XP developers and I were asked to help a client introduce XP into the organization. We had some initial success and quickly brought the team up to speed. After a couple of months we had a collocated team, working in pairs to deliver tested software every two weeks. At the end of eight months however, we had left the client and had had only minor success in influencing other Agile projects.

So what went wrong? Looking back on it I made several mistakes of omissions:

- I failed to adequately address dependencies between two project. The result was a delay in completing functionality, and this directly impeded the team’s progress.
- I failed to recognize and address issues around developer compensation. Developers were reward (by promotion and compensation) for both specialization and building large frameworks. Introducing Agile and encouraging generalization over specialization of skills was a threatening move because it challenged both their position and their year end bonuses. By failing to address the issues around compensation I failed to gain influence with the architects, and eventually was unable to make necessary recommendations.
- I failed to fully educated senior managers on adaptive planning. The result was that they continued to try and fix an end-date which became increasingly unrealistic with each passing iteration and casting doubt on the other successes that the project team were achieving.

My reason for providing this example is to make you aware that successfully introducing Agile methods into an organization is not a simple task. It takes many different skills both technical and political. The physical and political environment that you’re work within requires as much attention as the technical problems that you’re trying to solve.

**Summary**

Over the last few months, I’ve discussed a roadmap for the introduction of Agile methods, one possible model for that introduction, and some of the impediments that can occur along the way. If you find yourself going down a similar path, I wish you every success and I would love to hear about your journeys.
About the Author:

Kane Mar: I’m a Agile coach specializing in Scrum and Extreme Programming currently working with great people at Danube Technologies ( http://www.Danube.com ). I’ve been a ScrumMaster and Agile Coach for over five years. I have recently become a Scrum Trainer (CSMT) and I’m looking forward to sharing my experience with others in the Agile community. My personal blog can be found at http://kanemar.wordpress.com.

Prior to my work with Agile software development, I had a 15 years of experience as a developer and project manager for waterfall and RUP projects working with Java, Smalltalk, C, SQL and PL/SQL. My recent technical interests have turned towards Ruby primarily because of it’s similarity to Smalltalk.

References:

1 Weinberg, Gerald, *Secrets of Consulting: A Guide to Giving and Getting Advice Successfully*
2 Ibid.
4 Berenstain, Stan and Jan, “The Bike Lesson”