TATA CONSULTANCY SERVICES

Experience certainty. IT Services
Business Solutions
Outsourcing

Adopting Agile Methodology in a Large Scale Distributed Development Environment

Much has been written about the exhilarating benefits that Agile methodologies bring to an organization. The latest trends and industry studies too show a steady rise in the adoption of Agile methodologies. Organizations across the globe, irrespective of their size and business, have started dabbling with this new methodology; some embracing it openly while the others adopting a little more cautious approach.

However, even after showing a considerable promise, Agile is still considered the second class citizen in the world of IT processes, which is dominated by the likes of CMMI®, ITIL®, and RUP. What is it that is limiting a wholesome Agile adoption? Why do most organizations still hold on to the procedural methodologies as their core processes? Industry studies have cited various reasons; the most prominent being the non scalability of Agile for large and distributed projects.

So does this mean that Agile is incompatible with large and global distributed development – or is it just our perspective of Agile that creates these potential roadblocks? This white paper talks about some of the author's practical experiences as a senior Agile Consultant in TCS, including the key challenges of adopting Agile in organizations running large and distributed projects, and the approach recommended to overcome these challenges to achieve a successful Agile adoption and deployment

About the Author

Dipanjan Munshi

Senior Agile Consultant

Dipanjan Munshi is a senior process and quality consultant at Tata Consultancy Services. He has extensive experience in the fields of Agile and Iterative processes and has led several consulting engagements dealing with Agile readiness assessment, Agile maturity assessment, adoption of Agile processes in large distributed environments and adoption of Agile processes in Service Oriented Enterprises. He has also done extensive research and development work in the area of integrating best-of-breed tools to provide an effective Application Lifecycle Management solution for an Agile environment.

He has executed several projects in the role of an Agile project manager and Scrum Master across various domains like insurance, financial services, embedded systems, transportation and hospitality, using the approach suggested in this paper. He has a master's degree in Computer Science from Utkal University, Orissa, India.

Table of Contents

Introduction

The Rise and Rise of Agile

It's official. Agile is now a global phenomenon and is rapidly becoming organizations' most preferred IT process around the world. As a part of the process consulting group in Tata Consultancy Services (TCS), Asia's leading IT Company working with a diverse customer base from start-up companies to fortune 500 ones, we are seeing the demand changing. Large enterprises which had been a staunch follower of plan driven methodologies have started to add an Agile playbook (or cookbook or handbook) into their organizational process library. The bolder ones have already announced their plans of going agile while the other more cautious ones have started deploying them internally in pockets.

The adoption has been further fuelled by setting in of the recent recession where suddenly, no one is sure of how the business landscape will change in the next few months. Delivering software a year down the line is no more an option; funding that are committed to an important project today, is being cancelled six months down the line. There is a sudden need for projects to add as much value as possible in the shortest time span, and who else but Agile can be the knight in shining armor during the troubled times.

Adoption Challenges

Although adoption of Agile is in full swing, unfortunately the same cannot be said about its 'organizational success' - and before I feel the pangs of the acerbic criticism from the ever growing Agile community, let me quickly define as to what I mean by 'success' of a process from an organizational perspective. Within the consulting dimension, we know a process is successful when it starts getting adopted at an organizational level and gets followed all the way up to the level of the CEO or MD. So yes, CMMI® is successful since it becomes the de-facto standard for an organization wherever it is deployed; ITIL® is successful since it is the de-facto process for service management wherever it is deployed. Small projects done with Agile will definitely be successful due to the very nature of the process but we are still way behind in getting Agile adopted at an organizational level for large projects. But of course there are exceptions and this paper will talk about some of those exceptions and how they managed to deploy Agile at an organizational level. But right now, let us think of the challenges that are stopping Agile to attain a first citizen status for many organizations.

Size Does Matter

It is no secret. The major challenge of Agile methodology is its scalability. Remember, we are talking about large organizations here with perhaps IT strength of more than a thousand people. There are smaller projects that run within this huge IT team with 8 to 10 people, and these are the pockets where these organizations have tried and succeeded with Agile. However, most of the projects that are built within these IT shops are monstrous. Teams of 200 to 400 people executing projects with highly complex technologies like mainframes at the backend and .Net at the client side coupled together through something like the MQ Series. The skill levels of the team are different, each unit is highly dependent on the other and it takes at least 3 months to build even the first increment of the working software.

Distance Does Not Make the Heart Grow Fonder

If size complexity wasn't enough, added to that is the mother of all complexity, the ubiquitous "global distributed development". With parts of project being run out of Los Angeles office, parts of it from Sao Paolo and the rest from Beijing, the vision of using an Agile methodology is the last thing that a CIO would have in her mind. And this is not only about offshoring to an external IT vendor. Multi national companies have development centers at all parts of the globes. Even local firms may have their IT development happening at multiple sites. E.g. a US based firm may have an office in Los Angeles, another in Austin and a third in New York. By far, arguably, the market perception of the biggest roadblock to Agile adoption is distributed development, especially those that have multiple vendors involved. Well, this doesn't come as a surprise since the very requirements of a distributed development seems to be an antithesis of the Agile manifesto. Consider this:

Agile manifesto says that "Individuals and interactions over processes and tools". Think of a distributed development. With the individuals sitting virtually on the opposite sites of globes, interaction is the key challenge. Face-to-face interaction on a daily basis is out of question. Mails and forums do work but there is usually a pretty big time lag. With multiple vendors the situation becomes ever more critical since they have their own processes, service level agreements and tools. In such cases if you do not have processes that are well-defined and a state-of-the-art tools landscape, it will be impossible to meet the deadlines with acceptable quality. So the distributed development manifesto would read "Processes and Tools over individuals and interactions", which is quite the opposite of the Agile spirit.

Similarly, while the Agile manifesto says "Working software over comprehensive documentation", it is assuming that a single team is responsible for the entire lifecycle from analysis to deployment. In distributed development, often the entire ownership is distributed amongst various vendors and therefore comprehensive documentation becomes the way of life.

The scalability challenge in Agile is not new. The Agile gurus and its most staunch proponents have voiced their concern about going Agile in a distributed model multiple times. They argue that the primary premise of Agile method, which is face to face communication and collaboration, suffers heavy restriction in a multi-site model pulling down the productivity to such an extent that it overshadows the cost benefits received from the distributed environment.

Large sizes and over-complexity of projects are a deterrent even with Agile techniques like Scrum of Scrums. It's easy to understand 50 people running 5 simultaneous Scrum teams and having a scrum of scrums with 5 scrum masters, but how do you explain 200 people running 20 scrum teams and a scrum of scrum with 20 scrum masters – anarchy?

The Silver Lining

So, does this mean that the usage of Agile is not suitable for large and complex projects and will distributed organizations never give Agile the first citizen status for their organization IT process? The good news is that "it is" and "they will". This paper will discuss our opinion and experience towards deploying Agile methodology for complex projects in a distributed development and institutionalizing Agile as an organizational process. We will explain our high level methodology, some of our best practices and talk about a few real life case studies where Agile has been adopted at an organization levels.

Agile Adoption in TCS

Our Experience

TCS has since long recognized the value of lean and agile development. We have been increasingly adopting agile practices in our deliveries for the past two decades. At the time when agile concepts were at their nascent stages in the IT industry, TCS had been applying rapid application development, lean approaches, model driven development and many other agile principles regularly, to ensure quality deliverable at a much quicker pace. With more well defined agile methodologies like Scrum and XP becoming popular, TCS has been leveraging these successfully, delivering projects using one or more of these agile methodologies, either in their out-of-the-box form, or using a modified version.

TCS has successfully delivered over 100 projects using Agile methodologies in a globally distributed environment, thus meeting our customers' business needs of time to market, quality and responsiveness to changing requirements. With over 4000 person years of agile execution experience, these agile projects have been executed in a variety of areas such as application development, maintenance, conversion, data-warehousing and embedded systems as well as for a wide range of business domains such as Insurance, Banking, Manufacturing, Retail and Telecom. The efforts range from small projects to programs with over 300 person months, team sizes ranging from 10 to 100 associates, and iteration lengths varying from 7 to 90 days. The typical offshore leverage in these projects is up to 70%. TCS has also successfully partnered with multiple clients to deploy Agile methodologies, either in their out-of-the-box form or as a hybrid model.

TCS has standardized its process for executing Agile in a globally distributed environment using its patented Global Network Delivery Model (GNDM™). All TCS projects are facilitated by the Process Excellence Group which continuously harvests the experiences and best practices from across the organization and uses it to continuously refine the standard process. TCS Agile projects collaborate using our internal knowledge management systems, communities of practice, and have access to toolsets and enablers hosted in the centralized organizational process repository.

TCS Global Consulting Practice offers consulting leadership to organizations providing thought leadership and helping them to realize the business benefits of organizational Agile adoption by assessing their Agile readiness and using a best practice framework for delivering optimal agility in distributed development.

Our Position on Distributed Agile Development and Agile Institutionalization

As a result of this extensive experience, we have come to realize the fact that a successful Agile deployment has a much wider implication than just taking an out-of-the-box Agile methodology and start following it. When an organization decides to go Agile at the enterprise level, there are many other aspects that it is looking at, in addition to using Agile methodology. Most of the times, it is the "enterprise agility" that is important and Agile methods form one part of this whole big movement, albeit a very important part. So to promote Agile to a status of an organizational process, we need to look at it in terms of enterprise agility, that is how quickly is the enterprise able to respond to a change in market or even better how it can take the position of a market leader through triggering the market changes.

Based on this thought, we see the following five key points, as mandatory to ensure institutionalizing Agile methods at the enterprise level. We call it the A-B-C-D-E of institutionalizing Agile methods.

1. (A)ssess: Make sure that you know of the organization's inherent maturity to adopt Agile. Identify and plan for mitigating any risk.
2. (B)alance: Devise the right Agile methodology. Agile development is not a digital condition. It is analogous. Organizations can achieve maximum enterprise agility even when they are lesser agile in their software development process.
3. (C)ommunicate: Communicate, communicate and communicate. Ensure that everybody, right from the president of the company to the developer knows and understands all about the new methodology and have their expectations set. This is irrespective of whether someone is involved in an Agile project or not. It is important to set the right expectations and build the right skill for succeeding with Agile at an enterprise level. The change management has to be well thought of and rigorous to promote the sustained use of Agile development methodologies.
4. (D)igitize: Even while you chant "Individuals and Interactions over processes and tools", strive for the maximum automation possible without hindering the productivity of the developers. In a large and distributed project, automation, and the right way of doing it is the key to success for Agile projects. And automation is not only restricted to development and testing. We should look at all possible areas as long as we remember that it is suppose to make people productive.
5. (E)volve: Agile institutionalization is not a goal, it's a journey. Its perfectly ok if the organization feels that they cannot follow a typical Agile method and instead will go for an iterative approach. Its more important that the Agile values like self organization, early feedback, etc. are built into the team irrespective of whether they are following a spiral, an iterative or even a plan driven approach. As and when the organization gains more maturity, their way of working will become more and more Agile. For example, if the experience of iteration duration of two weeks is unnerving, it can be changed to four or six weeks; however, once you have chosen the duration, the rhythm needs to be maintained.

"A" for Assess Agile Readiness

Problem

Our experience shows that clients, who fail to plan appropriately, mitigate barriers/risk and implement Agile thoughtfully, may dramatically increase project risks, and fail to achieve the intended benefits. It is important to note that Agile methodologies, in their out-of-the-box form, are not meant for every organization or every project.

E.g. if your business model demands a strict hierarchical organizational structure or if a certain project has totally static requirements, Agile adoption may either fail or not show any value over a traditional model. Applying Agile to a project or organization where it is not applicable is a recipe for disaster and may actually result in losses, conflicts or at least a bad-taste-in-mouth.

Solution

It is extremely important to understand an organization's highs and lows related to Agile deployment before jumping into the bandwagon. Deploying Agile methodology is not a digital signal, i.e. it is not correct to assume that an organization is either using Agile or not. Agile adoption is actually like an analog signal where the degree of agile adoption may vary from organization to organization depending upon its internal machinery.

It is also not correct to assume that the more the degree of agility in an organization, the well-off it is. In fact, in true sense, an Agile Enterprise is the capability of an organization to sense or trigger change and provide a quality response in the quickest possible time. The word "quickest" may vary depending upon an organization or project. For a space center project, "quickest" can mean a few years whereas for a banking project, "quickest" may denote a couple of months, and again for telecoms, "quickest" would mean a couple of weeks. The connotation of the word is different and therefore, the extent to which an organization would want to follow an Agile software development procedure is also situational.

Agile methodology runs on certain assumptions that can be realized in terms of various parameters like requirement dynamicity, external dependency, business involvement, development platform, legacy code, etc. An organization needs to benchmark each of these parameters for suitability of Agile adoption and then measure its current state to decide the ones that are more likely to be detrimental to the adoption.

Some typical adoption risks may be an organization having extensive legacy code, a project having static requirements, or a team made of inexperienced associates, or a large number of multi-location development centers. Of course, it is not that Agile cannot be deployed in such cases; however, we need to keep in mind the inherent risks due to the internal machinery and decide on their mitigation early.

In TCS, while helping a distributed organization to adopt Agile, the very first task that an Agile consultant will perform is to do a due diligence and understand the organization's inherent machinery and identify the primary risks of its Agile adoption. We do this through some focused work on studying the organization's existing processes, review sample deliverables, conduct interviews, etc. and finally feed the data into our Agile readiness assessment tool. This tool would analyze the data around 29 parameters like technology, executive support, skill level, requirement dynamism, etc. and pull out the aspects that can pose serious risks to the organizational level agile adoption. Even at a project level, we run a tailored version of the tool to understand the project's readiness of Agile.

The solution is applicable to organizations that are starting on their Agile adoption journey as well as those who are already into Agile but are not receiving any substantial value-add with the methodology.

"B" for Balancing Agility and Discipline

Problem

One question that customers often ask us is which Agile methodology is best suited for their adoption. Most organizations go through a large amount of literature and web sites, and spend considerable time on research to understand the various Agile methodologies like Scrum, XP, DSDM or FDD. If an organization is loyal to a particular method or a brand, they may find this choice easier and just go with the methodology professed by their brand. E.g. organizations with brand loyalty towards IBM are more likely to go for the Agile Unified Process. Similarly organizations that are thick into UML may opt for Agile Modeling or FDD.

However, while this kind of selection may allow the organization a premium access to support and consulting from their preferred brand, it is not necessary that the methodology will be 100% suitable for them. In fact, arriving at an objective justification on why we need a certain methodology is extremely difficult and there are no known universal checklists available for this in the industry.

Of course, in some cases, the choices become apparent. For example, we know that Scrum has been proven to be one of the most successful methodologies for new adopters, since it gives them the flexibility of finding their own way of doing work and focuses mostly on keeping the team free of impediments. Therefore, any new adopters may want to go with Scrum. However, this still is not an objective approach and even with Scrum, teams may have to turn to other sources for prioritization techniques, estimation techniques, engineering practices, etc.

In our experience, we have seen that almost always, no out-of-the-box methodology is complete enough to address an organization's need to be Agile. As mentioned before, all existing Agile methodologies works on certain assumptions that may not hold true at every organization.

Let us take an example. We know that Agile relies on skilled individual and tacit knowledge. This is a fair assumption when we are working with a team of 8-10 people. However, for large scale projects spanning to 50 or 100 team members, it is next to impossible to get so many senior skilled resources. Most company banks on using a certain percentage of junior staff with required training and mentoring. In such scenarios, a regular out-of-the-box Agile methodology may need to be tweaked in order to nullify the above assumption.

The above example takes us back to the earlier readiness assessment. It would be then safe to assume that if our readiness assessment chart shows no risk, we may go ahead implementing an out-of-the-box Agile method. However, in practical world, this is an ideal state and is hardly ever fulfilled. In most cases, organizations will have a number of risks to mitigate, which means, almost always we would need to "tweak" an Agile methodology or create a hybrid for adoption. Now, here comes the million dollar question. What do I need to "tweak" so that I have the optimal Agile methodology for my organization? Remember, we are talking at the enterprise level here, not a couple of isolated processes where we can find out by trial and error.

Solution

In "Balancing Agility and Discipline: A Guide for the Perplexed", Barry Boehm and Richard Turner explains that adopting Agile is not a digital condition. Agility is a continuous spectrum based on a set of parameters. The following diagram from the book shows the five parameters that are considered vital to Agile adoption.

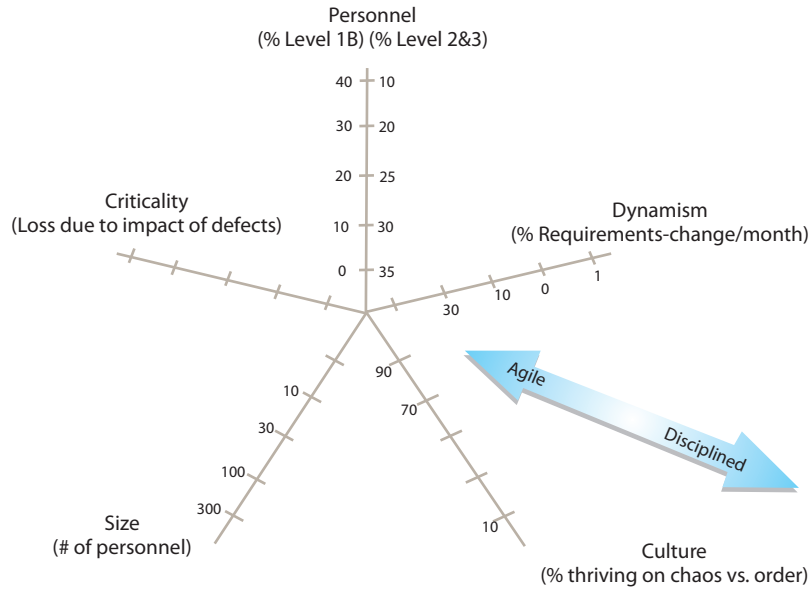

Figure 6: Building Armor to Agility (Source: Barry Boehm, Richard Turner - Balancing Agility and Discipline, Addison-Wesley)

The more an organization is to the center of this graph, the more ideal it is for pure Agile adoption. However, in practical cases, for an organization, some of the parameters may be closer to the center while some would be far away from it. E.g. an organization may have a high level of changing requirement (towards the center of the Dynamism axis) but the project may be of gigantic size (far away from the center in the Size axis). In such cases, the process should lie somewhere along the blue arrow. The Agile method adopted by an organization will have to consider an appropriate mix of practices from various Agile methodologies and even from the plan-driven world, which would be optimally suitable for the organization. The authors refer to this as "building the right armor to agility". Of course getting the right armor is a pretty humongous challenge.

In our consulting experience as well, we have seen that a key to a successful Enterprise Agile deployment is to create an Agile methodology that is best suited to the organization. However, it has to be remembered that even if we are creating a hybrid, we should still be aligned to the Agile manifesto. The process then gets continuously refined as and when the organization gains more maturity on the Agile paradigm.

For example, we can start with the simple agile practices like creating architectural spikes, paired programming, rapid application development or using Kano model for prioritization. This method will get more and more refined till, let us say, we reach at the point of executing enterprise level Scrum.

With yet another view, we may decide to go with Scrum but retain internal and external quality assurance, formal sign-offs and other plan driven techniques as required by the organization. Down the line, these get overridden by more agile practices till the entire enterprise is following Scrum.

The crux of devising an optimal Agile method, therefore lies in the ability to identify the right practices. In many organizations, this is done based totally on experience, and the methodology goes through refinement for several iterations till it reaches the optimal benefit level and gets standardized. However, we suggest a little more precise approach (just a little more precise keeping in mind the law of diminishing returns).

Here, we would like to introduce the concept of "Quality Attributes". A "quality attribute" is a metric against which the success of the process or the developed product will be judged. E.g. time to market, productivity, effort, schedule are process related quality attributes, whereas performance, product quality and defect density are product level attributes. An optimal Agile process will be the one that will meet all the goals for its quality attributes. To understand, how an optimal process can be deduced using quality attributes, we need to understand a second concept of "practice based process model" a concept that has been made popular by Dr. Ivar Jacobson when he used it for his Essential Unified Process model. In TCS, we have been following the practice based process model approach for a long time through our ETVX cards.

With this concept, you no longer have a process but a set of practices. Each practice does something specific, given a defined set of input and produces a defined set of outputs. Using a practice based process model, you are no longer bound to a single process. You can assemble the practices from different models and create your own customized process. For example, combining paired programming with V-model quality assurance or combining rapid prototyping with test driven development. The practices are like plug-and-play components making the entire process model adaptive and "agile".

Now back to the quality attributes. Since what we now have is a set of practices (from the various agile as well as plan driven methods), we can grade each practice against the process or product quality attribute defined by the management. E.g. consider a sample below:

Business Needs	Use Cases	Joint Application Development	Service Modeling	Peer Reviews	User Stories	Architectural Spikes	Daily Scrum
Predictability	◕	◕	◕	◕	◔	●	●
Volatile Requirements	◔	◑	◑	◔	◕	◑	●
Quality	◕	◕	●	◕	◑	◕	◑
Productivity	◑	◑	◕	◔	◕	◑	◑
Maintainability	●	◑	●	◑	◔	◔	◑
Time to Market	◔	◑	◕	◔	●	◑	●

● Excellent
◕ Good
◑ Satisfactory
◔ Poor
○ Very poor

The above table is just an illustration to explain the strategy and may not reflect actual gradation.

Based on the business goals or risks of the project, we can now select the right set of practices that will enable the team to reach the goal in the best possible way. So if maintainability is a risk to the project, we may not want to adopt user stories and go for usecases instead. However, if time to market is a business goal, then we would prefer to have user stories rather than usecases.

In real life, this decision is complex because for most projects there will be multiple goals or risks that are conflicting to each other. E.g. what do you select for a project which has high maintainability risk but also requires a faster time to market? I can think of arriving at set of guidelines which can help the team to decide on which requirements can be expressed as stories and which as usecases? The complexity increases as a project has more needs and risks like high data integrity, high quality with low effort, etc.

Consider a third angle. The practice of "architectural spike" is common to both Extreme Programming (XP) and Rational Unified Process (RUP). However, the way these two methodologies implement this practice is very different. While both define this practice as a part of the exploration phase, XP uses a more metaphorical approach with the focus of arriving at a more confident estimate and plan. RUP, on the other hand stresses on more thoroughness of this whole activity, identifying every possible and foreseeable technical risk in the project and creating a spike or a proof of concept to eliminate the risk. The focus in RUP is therefore more of risk elimination than providing a more confident estimate. Of course, with all those work typically done in elaboration phase, the estimates at the end of the elaboration phase will be much more accurate. However, by then we have already consumed 40% of our project schedule in running inception and elaboration and still not have working software on our hands. XP will talk about identifying a few metaphors and quickly doing some minimal trials and spikes to understand the complexity and come up with more confident estimate in a matter of probably a couple of weeks. Therefore, we see that the focus of the same practice differs by methodologies. Each has their own strengths and limitations. Deciding on which practice to use was complex enough and not we have to deal with how we want to interpret the practice. For a project, where a number of new technologies are being used, the RUP spiking may be more favorable, whereas for an architecture that is not too risky, XP spiking may be preferable.

As mentioned earlier, the model has quiet a bit of complexity; but then so does life. However, this model gives the process owners a good handle on what are the potential risks that the adopted process may have for the project. It can provide an initial all-inclusive draft of the process model that is suitable for the project and can be fine-tuned. Most of all, since this mapping is maintained at the organization level, any organization level best practices can also be added to the matrix and subsequently to the adopted process.

Going down the line, we see this model to be more quantitative than qualitative as it is of now. With a quantitative model, the project manager will be able to make quantified statements of the risks and gains. E.g. if we use user stories in this process, it will be 30% less maintainable but will have 60% reduction in time to market.

"C" for Communicating the Change to All Stakeholders

Problem

Agile adoption is a complete change in paradigm and requires a structured and people oriented approach to be successful. Organizations have often found it difficult to reorient the mindset of the stakeholders to an Agile landscape, especially at the levels of management and business. The concepts of "servant leadership" may not be understood or agreed upon by many managers who is used to a "command-and-control" structure. Similarly, the fact of receiving semi-finished products and collaborating with IT on a daily basis may not be well received by business.

The adoption is no less tough for the development team itself as they have to now be in a self organizing and self commanding mode. If the team is used to receiving assigned tasks from the lead and executing them in a time bound manner, the absence of a command and control leadership may either make them apprehensive or unproductive.

Moreover, if the management or the team is looking for the promised Agile benefits right from the first iteration, they are bound to be disappointed. The first few iterations from teams who are new to Agile may actually show a decrease in productivity. In many cases, when the management and team do not see the value in the first few iterations, they revert back to the traditional model, killing the adoption. Besides, they may still want to stick to their plan driven metrics like schedule slippage or effort slippage which may not make much sense with Agile.

Consider the scenario, which is a real life experiences our consultants faced on ground. This was a large organization, which was about to launch a very important product, and thought of adopting Scrum to achieve a quicker time to market. Six months down the line, the adoption had failed and the organization had to revert back to the traditional model. While studying this failure, our studies showed a glaring gap between the expectations and understanding that the various stakeholders had around Agile.

- The business understood that Agile meant having the flexibility of demanding any requirement at the middle of an iteration and receiving them at the end of that iteration
- The executive management understood that Agile is something to be executed at the lower development level and it will not mean any change to the overall organization hierarchy or policies
- The project team understood that Agile meant that they can get away with no documentation. Any bug resulting from an iteration is taken in the next iteration as a change and therefore their release has no defects
- The process engineering group, who was given a checklist for Agile projects, understood that whichever project was not following the checklist to the tee was not Agile
- Finally, the vendors, support and maintenance teams were least concerned with the internal methodology and followed their own processes that did not gel into the Scrum method

Looking at the above scenario, we realize how important it is to have the entire set of stakeholders on the same page as to what should be the expectations in Agile. Each level of stakeholders has their own role to play in the adoption and yet these roles are all dependent on each other. In our experience, almost 90% of the Agile adoption failures that we have seen are due to lack of people change management.

Solution

The solution to this problem would require an organization to understand that an Agile adoption program is not just adopting a new process but a systematic and structured change management directed not only towards IT but all stakeholders who would be affected by this adoption like business, management, end customers, etc.

We see the change management roadmap work in three phases – Envision & Explore, Establish & Enable and finally Execute and Evolve.

Envision and Explore: Identify the change agents and risks of agile deployment early in the journey and limit the usage of methodology to the few senior associates who are Agile enthusiasts and ready to make a sincere effort to change their mindset. These will be experienced people who can handle conflict and identify failures as improvement opportunities. Once we have shown the value that we gain out of these projects to the management and business, we have a case.

Establish and Enable: The next phase is to scale up the methodology and take it forward to other teams. At the same time, the early adopters are now eligible for taking up the roles of Agile project or program managers. Continuous sharing in the form of coaching, mentoring and constructive discussions is needed to ensure that there is a steady flow of knowledge and everybody understands the spirit of Agile. The central idea of this phase is to make the organization agile-enabled. Different norms are set at executive and business level so that their expectations are aligned to Agile values. Enablers and models are put in place. However, the use is still limited to a few key projects (of various sizes) under the strict supervision of the Agile managers and coach.

Execute and Evolve: Once, we have most of the organization at a comfort level with Agile projects, we work on evolving the methodology through continuous expert mentoring, research, center of excellences, forming Agile communities and exposing the organization at an industry level. This will result in multiple changes to the methodology, which will need to be communicated and exercised with proper coaching. At this point of time, we expect that enough value has been shown by the Agile projects, to make most of the organization Agile enthusiasts and ready to take up Agile methodology.

At the end, we would like to stress that even when the organization is doing small projects in the "envision and explore" phase or limited projects in the "establish and enable" phase, the results, the advantages, the limitations and the challenges should be made completely transparent to everybody in the organization. Communication is the key to success.

"D" for Digitize to the Maximum Extent Possible

Problem

Individuals and interactions over processes and tools" – or so says the Agile Manifesto, but like all other statements within the manifesto, this is often been misread as "no processes, no tools". I remember of a project that was being executed with Scrum. When the organization wanted to go distributed, they wanted our help in understanding the consequences of going distributed with an Agile model and what recommendations we may be able to provide. As a part of our recommendations, we suggested them the use of a collaborative development tool. This tool will record user stories, record estimates, maintain burn downs, generate metrics, store deliverables and artifacts. Apart from that it also had collaboration functions like discussion forums, blogs, wikis, mailing lists and so on.

While the organization accepted the justification on needing a global collaboration forum, they rejected the idea of using a tool to enter user stories or estimate them, or for maintaining sprint backlog. They saw it against the Agile manifesto in automating their everyday project execution. Consequently, they went distributed using a Web 2.0 portal. The results were disastrous. The two different sites were not able to keep their stories synchronized; the backlogs from both sides never matched and the team had already divided itself into "us" and "them". The project productivity took a hit and 6 months down the line, they were asked to fall back to an iterative method instead of Scrum to complete the project. The project was completed eventually but without the promised benefits of Agile.

In most projects that are executed with Agile methodology, the scope of automation is often limited to testing and system builds. Most Agile practitioners actually believe that using tools in other areas is counter to the Agile spirit since it limits the individual interaction and hampers productivity. They would prefer user stories to be tracked as post-it notes on a white board or burn-down charts plotted on a piece of chart paper and fixed to one of the walls in the development area. To a certain extent, they are right. However, it is easy to understand that a group of 8 to 10 people sitting in the same room facing each other will not require a separate tool for requirement management or design modeling or logging defect data but when we are talking of teams sitting half-the-globe away, it is a different game. Post-its on white board, charts on the wall and dedicated team rooms will simply not work here. We need to fall back on automation and use them as much as possible – "without hampering the team productivity".

Watch those words – "without hampering the team productivity". Let me discuss one more of our experience. A large project working on a distributed Scrum model did very well to automate most of their work. They had automated requirement management, code development, testing, build and of course version control. They used industry standard tools which were well-known in their areas. However, two months down the line, the productivity was going downhill. As an Agile coach, when we surveyed their process, projects and tools, we found that the requirement management tool and testing tool were from different vendor and did not talk to each other. All requirements that were entered in the requirement management tool had to be reentered in the testing tool to build the test cases. Well, not that difficult, just that someone had to sit for hours cutting and pasting requirement between tools. However, a bigger issue was yet to come. We found that when small issues were found in the requirement during testing, the code was updated and the requirements in the testing tool were updated to generate the changed test cases. However, these changes were not synchronized back to the requirement management tool consistently. As a result, the requirement in the requirement management tool and that in the testing tool were hopelessly out of sync. And eventually when someone tried to synchronize them, they overwrote the version in the testing tool which had the latest changes.

Solution

Let us first define our view on automation. We believe that the core principle of any Agile organization is to respond to change in the fastest possible way. Automation is an important means to enhancing productivity. Therefore, it will definitely help the organization to be more Agile. If I can generate 80% code with a model driven code generator in 2 seconds, then why would I want to write the code from scratch? Especially, when talking about distributed development, automation is essential as we saw in the first case in the "Problem" section.

However, the other side of the coin can be that organizations may have deployed way too many tools than they really need or the tools deployed may be unnecessarily complex than what is actually needed. E.g. if a small project team executing small and fairly simple project needs to perform requirement management, the need is as simple as "have a searchable repository of the user stories". It is easy to see that it will be much more productive if the team uses an excel sheet kept in a shared location instead of going for a big branded requirement management tool in the market.

Another reason why tools may affect productivity is that organizations often deploy multiple tools that do not have any defined handshake with each other. Such tools introduce a lot of data redundancy and effort duplication, just like the second case example we talked about in the "problem" section. Sometimes, organizations use tools or tool-suites that do not align to the organizational process. For example, if for maintaining user stories, an organization uses a modeler that stores usecases, it means that each user will have to redesign the user stories to align to usecases to be able to store it in the tool. This is not a healthy sign. Tools are supposed to render processes. If they cannot render the process effectively then they are not the right tools. You do not change your process just because it does not fit your tools.

We believe that a process is not just a couple of handbooks. Tools are a part f the process and therefore when we are setting up an organization wide process, be it CMMI® or ITIL® or Agile or any other process, its important to evaluate and create what we call as an end-to-end tools "ecosystem" that can render the process from start to finish with as minimum user intervention as possible. We look at the following three steps to build a tool-ecosystem:

1. Assess and evaluate various tools to find out if they will render the process effectively and how much changes need to be done towards this. Keep the tools as light as possible as long as they serve the purpose of the process. Fancy tools often come across with lot of complexity and not to mention the costs. Many open source tools are as good as their branded counterparts. If I need to just generate test data, I should buy a tool that would do just that. Tools with additional features will just increase the complexity and my total cost of ownership.
2. Once an organization has selected all the necessary tools, it needs to create a blueprint of the ecosystem. Its important to study how the tools will be connected to each other, how much work will have to done be done manually, what scripts can be created for enabling the handshake and so on. With this, the expectation is set from each of the tools as well as from the project teams that will be using these. A sample blueprint context diagram is provided below:

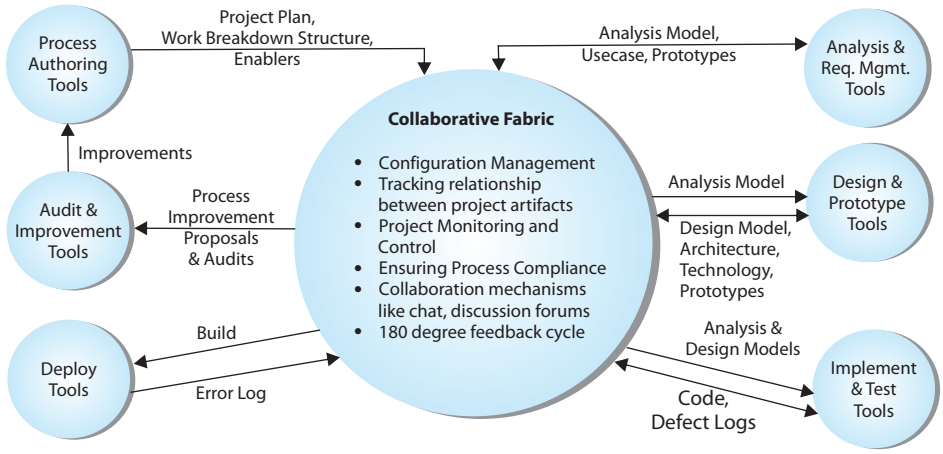

Figure 7: An illustrative example of a tools ecosystem blueprint

3. Finally do the actual integration. It is always advisable that if the tools are from different vendor, it makes sense to do the weaving using a centralized collaboration fabric. The collaborative fabric will typically make the tools invisible to users and will have the onus of running the scripts. Run end to end prototypes to ensure that the ecosystem does not break the process and renders it end to end.

The importance of a tools-ecosystem can be realized when we consider the recent disruptions being caused by the Rapid Application Development (RAD) paradigm. RAD unlike others methods has a high technology context. It is changing the way software is being developed and projects are being managed. For instance Ruby on Rails today has inbuilt auto test documentation, auto-design documentation, test factory, template-driven development (Model Driven Programming), parallel runtime and design time (less Service Transition). The RoR paradigm was relatively obscure until the success of Facebook, Twitter and Salesforce who support collaborative development in highly scalable environment. Although this may seem to reduce the significance of technology agnostic Agile (or for that matter any) process model, the truth is that RAD compliments the Agile process models beautifully by making digitization a way of life and ensuring that the team spends less time worrying about the creating industry standard technology models or proving them. The productivity of such an environment is high and most ideal for Agile adoption. The change can be felt from latest vendor moves, where Microsoft is forced to include RoR in MSF and its Development environments, and Sun in JE6. Also MS Team System and Eclipse are including Rails.

In some extensive research that we did in-house to compare the productivity of a non-digitized Agile environment to a digitized one, we noted an approximately 24 times increase in the productivity with a digitized environment. We found that the average time taken for an associate with intermediate skills in J2EE, to create a simple end to end application with one screen sending and retrieving data from a backend database was almost 4 hours. Using one of our proprietary code generator tools, he could create the same functionality (and with better quality and coding standards) in less 10 minutes (which would be around 24 screens in 4 hours).

Therefore, digitization does enhance the agility of a process, especially when working in a distributed environment.

"E" for Evolving the Process, Continuously Removing Waste

Problem

Organizations are dynamic. They change based on their environment. What may be a differentiator today may be a routine feature tomorrow. All organizations need to constantly change to ensure that they stay ahead in the competitive landscape. Since application development and maintenance process is an integral process for a company, it is obvious that the process also need to change continuously to accommodate the changing face of the organization.

E.g. If a quarterly release of a product was a driving feature for an organization in the market, down the line, they may have to be faster, say provide bi monthly release, in order to keep up with competition. In such a case, the existing process that caters to a quarterly release may not be able to scale up to a bi-monthly release and need to be changed. It is not just a case of changing the release cycle. It may mean that the organization will have to review the entire process, removing practices that bring down time to market, improving on practices or selecting new practices for higher productivity, having new or enhanced set of metrics for managing the projects and so on.

Enterprises which do not take the initiative of reviewing their organizational process are seriously limited in their abilities to scale.

Solution

We believe that process improvement is a continuous journey and not only triggered by external entities. It has to be a proactive initiative. Process improvement can happen on various levels:

1. Periodic assessment of the process and comparing it against the existing business goals
2. Improvement using Lean Six Sigma
3. Focused innovation based on understanding business trends and opportunities

Periodic Assessment

This is the simplest and quickest way for process improvement. In this approach, experience Agile coaches or consultants sample a number of different type of projects to understand the various opportunities of improvement. They study the process and project deliverables; sit with the team during their planning and retrospectives; conducts group or individual interviews to understand the team self organization and motivation level and so on. This method helps in quickly identifying the gaps that the current process has with organizational goals as well as the Agile spirit. The process can then be revised accordingly.

Using Lean Six Sigma

This is a more structured way of process improvement using the DMAIC and DMADV methodologies. It will use mathematical concepts and tools to measure and improve the process with much more precision. This approach is quiet involved and requires dedicated time and resource and is therefore advisable only for processes which is constantly failing on one or more business goals even after repeated assessment and revisions.

Innovation

Innovation will allow the organization to study the trends and the market deviations thoroughly and suggest improvement recommendations with a futuristic view. Innovation may be disruptive requiring retiring the entire existing process and going for a new one. Innovation may happen at a project level where some developer comes up with a new and different way of testing, or at the process level where the organization adopts a different business or process model

Summary

Can the IT shop of a Fortune 100 company build an application with the same speed and neatness that a start-up in Silicon Valley would do? A less thoughtful answer would be no. Agile has been perceived as the quality of the small and ingenuous, and we seem to believe that as the IT shops grow in size, more controls take over in operations. Hence, agile qualities in large IT shops have always sounded philosophical. This perception is changing. Recent facts and studies support the notion that large IT shops today can quickly deliver large & efficient software using Agile methodology. For instance, in banking, many of the foremost banking suits have been spin-offs from large banking companies.

Therefore, the better question is how a large unit can act like small in agility, but remains big in risk management. When we tend to emphasize on the latter, we move towards the classical Waterfall approach, which has hidden cost in terms of requirements gaps and probable program failure. We then try to be agile by retaining most of the heavy practices but making the software development lifecycle iterative or spiral. In other words, we simply loosen the controls and start calling it agile development, and expect it to behave like one as well. Many a times, we are not conscious of the best practices that Agile methodologies use to reduce the risk of eliminating controls. For instance, Scrum has the principle of "documenting less but broadcasting more" - a mechanism that makes the project inherently risk-aware despite having less artifacts.

Today, Agile is at crossroads not in terms of "how loose" or "tight" should the software development lifecycle behavior be, but in terms of how it brings in the right collection of best practices, which would make the project both requirement-aware and risk-aware, ensuring high flexibility, reduced risk and increased visibility. This correction in our perception of Agile has happened rather late – I claim so as I see an explosion in the adoption of Agile methodologies that exploit this confusion. What is needed is the practices that the IT shop would institutionalize to hedge the absence of many merits that traditional waterfall still carries. This diminishes role of methodologies and heightens the need for a best practice culture.

About TCS' Global Consulting Practice

TCS' Global Consulting Practice (GCP) is a key component in how TCS delivers additional value to clients. Using our collective industry insight, technology expertise, and consulting know-how, we partner with enterprises worldwide to deliver integrated end-to-end IT enabled business transformation services.

By tapping our worldwide pool of resources - onsite, offshore and nearshore, our high caliber consultants leverage solution accelerators and practice capabilities, balanced with our knowledge of local market demands, to enable enterprises to effectively meet their business goals.

GCP spearheads TCS' consulting capacity with consultants located in North America, UK, Europe, Asia Pacific, India, Ibero-America and Australia.

About Tata Consultancy Services (TCS)

Tata Consultancy Services is an IT services, business solutions and outsourcing organization that delivers real results to global businesses, ensuring a level of certainty no other firm can match. TCS offers a consulting-led, integrated portfolio of IT and IT-enabled services delivered through its unique Global Network Delivery Model™, recognized as the benchmark of excellence in software development.

A part of the Tata Group, India's largest industrial conglomerate, TCS has over 143,000 of the world's best trained IT consultants in 42 countries. The company generated consolidated revenues of US $6 billion for fiscal year ended 31 March 2009 and is listed on the National Stock Exchange and Bombay Stock Exchange in India.

For more information, visit us at www.tcs.com.

Contact us at

For more information about TCS' consulting services,
email us at global.consulting@tcs.com
or visit www.tcs.com/consulting

TCS Design Services | M | 0110

Subscribe to TCS White Papers
TCS.com RSS: http://www.tcs.com/rss_feeds/Pages/feed.aspx?f=w
Feedburner: http://feeds2.feedburner.com/tcswhitepapers

TATA CONSULTANCY SERVICES

www.tcs.com