

# Growing Agility in a Large and Distributed Enterprise

Erik M. Gottesman  
*Sapient*  
egottesman@sapient.com

## Abstract

*Many large and distributed organisations encounter challenges when seeking to introduce and institutionalise agile methods. This is often true even when individuals are philosophically aligned to the values and principles which underpin the process. In this report, we share our experiences in establishing agile delivery capabilities at Sapient, an organisation of over 4000 persons distributed across three continents and hundreds of project sites. This report provides an overview of our experiences in establishing and employing our delivery methodology, Sapient/Approach. We present the thought process followed in its definition, the organisational programmes we have implemented for managing adoption amongst our project teams and our viewpoint on the tooling capabilities required to enable and sustain agility in this operational context. We also specifically tackle the often prickly matter of compliance and describe how we have balanced the needs of self-organisation and flexibility at the project level with the needs of the larger organisation.*

## 1. Introduction

Before the first “agile” methods were formalised, industry reports such as the Standish Group’s *CHAOS Chronicles* brought into focus a short list of key enablers for project success, consisting of such influencing factors as user involvement, executive management support, clear business objectives and minimised requirements. Agile methods conspicuously bring the importance of such things into focus, a fact recently recognised by the explicit inclusion of an agile/iterative process on the Standish Group’s top-ten list for 2004 [1]. Due in great part to such visibility, agile methods are gaining significant momentum within the IT industry. However, many large organisations have encountered significant obstacles to adoption and institutionalisation of agile practices.

From an adoption and change management perspective, large organisations struggle with challenges relating to:

- The proliferation of seemingly competing or conflicting agile methods to choose from
- Insufficient infrastructure to support necessary communication and collaboration
- The need for organisational standardisation as a means for delivering a consistent customer experience, managing operational costs and achieving efficiencies of scale

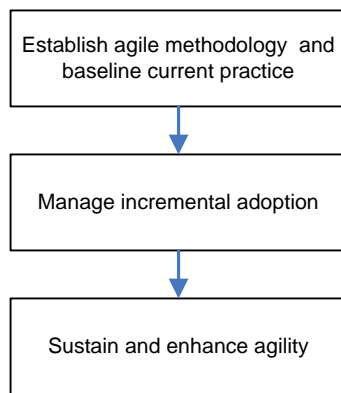
Apart from the inevitable “growing pains” associated with the transition, the use of agile methods in this operational context is also challenged by common “business as usual” practices such as:

- The use of distributed teams either as a matter of preference (e.g. as a means of achieving a lower cost of operations) or due to unavoidable circumstances (e.g. due to skills scarcity and individual mobility constraints).
- Hand-offs between operational teams at specific inflection points in the software project lifecycle (e.g. at “go live” or product end-of-life).

These issues are perhaps not unique to large, geographically distributed organisations. However, they are arguably more important in such contexts and point to the pre-eminence of “team” as a foundational construct in agile methods to the disadvantage of the larger organisation. Based on our own experience in navigating through these issues, we believe there are a number of important points, reflected upon herein, that must be recognised and acted upon when seeking to grow enterprise agility. Accordingly, this experience report is organised into three major discussion sections:

- Section two presents major issues we encountered in the definition of a workable, agile-based methodology for our organisation and how we resolved them.
- Section three describes our approach to managing the adoption of agile methods and key lessons we have learned along the way.
- Section four articulates our point of view on the tools and organisational programmes required to sustain and enhance agility once the initial transition is complete.

Throughout this report, we present our experiences and acquired learning as a set of repeatable patterns for managing the overall agile transformation lifecycle.



**Figure 1. Agile transformation lifecycle**

## 2. Establish a Methodology

We begin our discussion as one might expect, taking a walk down memory lane and recounting some of the trials and tribulations experienced in formulating the current incarnation of our delivery methodology, known as Sapien|Approach (S|A). However, before we dive into that, a bit of organisational context is perhaps in order.

### 2.1. Our History and Impetus for Change

Founded in 1990, Sapien is a global services firm that helps clients innovate their businesses in the areas of marketing, business operations, and technology. Headquartered in Cambridge, Massachusetts, Sapien operates across North America, Europe and India. Just over half of Sapien’s delivery people are based in India, working out of our facilities in Gurgaon and Bangalore. The remainder of our people work in North America and Europe, mostly out of our client’s offices.

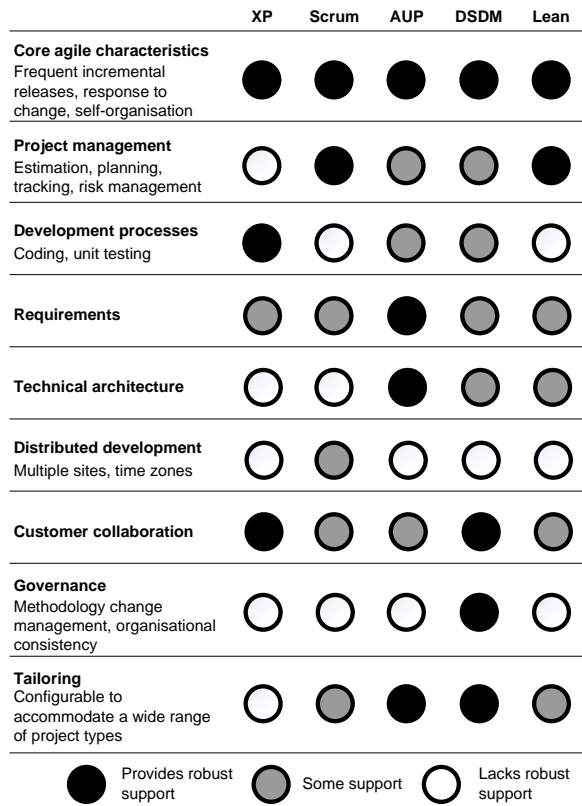
The vast majority of our teams work in a distributed fashion, our U.S. federal government business being a notable exception.

Sapien pioneered the use of fixed-price contracting as a means of driving on-time, on-budget delivery. Over time, we have adapted our approach to respond to the dynamics of our clients’ businesses and now employ a variety of engagement models to ensure the harnessing of change for competitive advantage. Our embrace of agile methods and their positioning at the core of S|A represents a key part of this evolution. Whilst fixed-price contracting brings with it cost predictability and disciplined execution, it is too often overly constrictive when dealing with complex, transformational initiatives that make extensive use of technology in unprecedented ways. Thus, we naturally gravitated toward agile methods as a means of providing flexibility and minimising waste whilst maintaining our historical focus on the client’s success.

### 2.2. Off-the-Shelf Doesn’t Work

When we began to consider the role that agile methods should play in our delivery methodology, we were met with a daunting proliferation of choices: Crystal, Extreme Programming (XP), Scrum, and Dynamic Systems Development Method (DSDM), just to name a few. Each of these, whilst rightfully attesting to being agile, approaches agility in a unique and nuanced fashion. For instance, XP takes a heavily developer-centric view, focusing on engineering practices such as continuous integration and test-driven development (TDD) and collaboration techniques like the planning game and pairing. It does not, however, provide the robustness of project management processes offered by Scrum. Similarly, DSDM provides risk management practices which are conspicuously missing from Scrum. Furthermore, Agile Unified Process (AUP) and Agile Modelling offer tools and techniques for requirements development not found in any of the other aforementioned methodologies. Thus, many of the discrete methodologies beneath the agile umbrella, when taken in isolation, leave something to be desired from an enterprise delivery perspective. Luckily, however, most of the “offenders” are complementary to one another and can be creatively combined to provide the breadth of process necessary to deliver complex projects. You simply need to be selective and understand how elements of each method satisfy specific needs within your organisation. Figure two illustrates some of the high-level distinctions between some of the leading agile methods and can be used to

initiate a conversation around organisational needs and hybrid solution possibilities.



**Figure 2. Key distinctions between some of the leading agile methods**

For us, being selective meant considering our needs across a number of process areas. In the end, we based the core of our development process on XP and Lean Development, but found a great deal of augmentation was required from an analysis and design perspective in order to deal with the complexity of systems we often encounter. So, we dropped XP’s “system metaphor” concept in favour of some of the more robust modelling techniques provided by Agile Modelling. To this, we added Scrum-based project management techniques, but deviated from both Scrum’s guidance on sprint length as well as XP’s approach. We have opted to allow teams to select an iteration length ranging from one to four weeks in duration. The iteration length is chosen based on a decision-making framework that respects such influencing factors as team size and structure, degree of geographic collocation or distribution, the number of integration partners and the nature of their delivery

methods, and the customer’s level of experience and comfort with working in a highly iterative model.

Of course, the need to weave together techniques sourced from multiple agile methodologies is not a new revelation; however, this is not the whole story.

### 2.3. Looking Beyond Agile for Answers

The myriad agile methods existing today offer the basic mechanics required to deliver valuable software solutions early and continuously with a high standard of quality. However, what happens when you seek to institutionalise these methods in organisations consisting of hundreds of teams large and small, working across multiple industries and using sometimes markedly dissimilar technologies? Simply put, today’s agile methods fall short of providing what is needed to cope with such complexity and variation in a consciously reasoned fashion which promotes organisational knowledge. Let us explore further what this means.

Team self-organisation is a central concept amongst agile methods. By *self-organising* we refer to a team empowered with decision-making abilities and motivated toward the achievement of a common goal *through the most optimal means available*. Implicit in the idea of the self-organising team is the individual team member’s ability to seek out and realise innovative new solutions that enable the whole team to converge upon its goal more effectively than otherwise possible. For instance, a team might employ the use of a style-checking tool fused with its continuous integration build as a means of automating a largely mechanical task otherwise performed through peer reviews or pairing. It’s wonderful to see the individual, in this case a developer, champion an innovative technique that improves his team’s delivery effectiveness. However, it would be disappointing if this or similar innovations went unnoticed and were not leveraged by other independent teams within the broader organisation. This is where continuous improvement and quality frameworks play a vital role.

More often than not, agile teams shun such models as Capability Maturity Model Integrated (CMMI), ISO, ITIL and Six Sigma as being bureaucratic and representative of the “old guard” – i.e. favouring processes and tools over individuals and interactions. We, on the other hand, felt this to be a short-sighted view. Is it not possible to combine the best of agile with the best these models have to offer? For instance, ITIL service-level management best practices provided us a means to more seamlessly integrate agile methods into an application support context. Similarly, we drew

structured practices from CMMI for configuration management, product distribution, defining and provisioning training, and identifying and deploying process improvements. CMMI also prompted us to invoke one very specific change in how we do development. As mentioned earlier, the core of the S|A development process is based on practices established in XP. However, we did away with XP's insistence upon pairing. As a services firm we must often work side-by-side with many partners. Effectively integrating such a team often means bringing together many different working styles and organisational cultures. Thus, pairing is not always preferable or possible. In cases wherein pairing is not practiced, other techniques must be employed to ensure the quality of work products. To this end, CMMI provides a wealth of guidance on verification. We'll discuss further the importance of CMMI later in the context of adoption management and process improvement, but for now, let us continue to explore how forces beyond the world of agile methods influenced S|A's definition.

With a view toward encouraging collaboration and eliminating the "distance" between business users and developers, we have embraced Model-Driven Development (MDD). Through the use of models, MDD seeks to establish a common syntax understandable by techies and non-techies alike. Moreover, the use of process automation tools within an MDD environment eliminates some of the manual transformation steps (*muda*) inherent in traditional software development.<sup>1</sup> When seen through an agilist's lens, MDD, with its emphasis on improving collaboration and increasing productivity, is entirely compatible with agile methods. Of course, integrating MDD techniques into S|A required us to reconcile some of the differences between MDD's model-driven techniques and XP's test-driven approach.<sup>2</sup>

Without going into exhaustive detail, it's worth mentioning a few other things from outside the agile toolset which have found a place of importance in S|A:

- Lean and Theory of Constraints provide powerful tools for identifying and managing sources of risk (variation) and minimising

---

<sup>1</sup> "Muda" is Japanese for "waste." Used in the context of lean thinking, muda refers to anything that interrupts flow, or consumes resources and produces no value. In the case of MDD, flow is improved and cycle time reduced through the replacement of manual transformations between various models and actual working code with automatic transformations, facilitated by MDD tools.

<sup>2</sup> For a more detailed discussion on this topic, you can refer to: Gottesman, E., *Model Driven Development through the Agile Looking Glass*, Agile India 2006, Agile Software Community of India, Bangalore, India, 5-6 May, 2006.

work-in-progress (WIP). We have found these concepts to be particularly useful within the contexts of estimation and progress tracking. The commonalities between lean's concept of flow, throughput in Theory of Constraints language, and velocity point to this utility.

- Function points and the COCOMO II model for software estimation greatly enhance the organisation's ability to understand team productivity and focus in on needed process and organisational changes to improve product output. These tools pick up where velocity leaves off and provide a means for comparing output across teams, which is essential from the perspective of portfolio project management.
- User-centred design (UCD) techniques fuel agile teams' capabilities to deliver working software early and continuously with a first-class user experience. Not surprisingly, the "user" in UCD meshes nicely with the "user" in XP's user stories and the rapid prototyping techniques common for years in UCD circles bear a strong resemblance to their software engineering analogues – spike solutions in the case of XP, and the RAD methods which preceded them. Beyond simple user-interface spiking, UCD also provides ethnographic research techniques which, when used during the most initial stages of problem definition, help teams design a better solution by understanding the people who will interact with it. Some specific interaction evaluation techniques we have found to be most useful include:
  - Analysing analogue products or systems for the ways they've functioned, succeeded and failed
  - Exploring the social and cultural notions underlying a project (e.g. the family, the home, identity, agency, community, etc.)
  - Identifying relevant user types and understanding where they talk to each other about the product
  - Performing user interviews, analysing advertising materials and conducting site visits.

Lastly, when transitioning to agile methods, one should not believe that every practice in place before the transition is unnecessary or wrong. Your organisation must have been doing some things right to begin with; otherwise you wouldn't be in business,

would you? We point this out only because we have seen a disturbing tendency amongst organisations to think that in order to be agile you must unlearn and divorce yourself from *all* of your old behaviours. This simply isn't true. You must let go of some things. For us, this included evolving our attitude toward fixed-pricing. We also had to get very disciplined about managing WIP and deferring decisions until the last responsible moment in order to avoid speculation and waste. Conversely, over the course of Sapien's history, we had developed some highly effective processes of our own design for such things as aligning stakeholders around a common vision, eliciting customer requirements and running project management offices (PMOs). All of this was good and required neither disposal nor significant overhaul in order to integrate with the new techniques introduced.

Figure five (which appears at the end of this report) provides a diagrammatic cause-and-effect view of many of the principles, techniques and disciplines which have informed and influenced S|A and the outcomes they drive. While we must again stress the need for each organisation to craft a methodology best suited to its purpose and vision, we do believe that many of the sources we have drawn upon satisfy the basic requisites for enterprise agility in the broadest sense.

## 2.4. Baby Steps

Organisations large and small are awakening to the reality that with increasing process automation and globalisation, the pace of business is accelerating. Many such organisations are now turning to agile methods as a means of injecting new life into their delivery teams and reconfiguring them for competition in an environment where quality and cycle time are top-of-mind like never before. When we first challenged our teams to adopt these "new" agile practices, we made a mistake. We initially positioned S|A as a fundamentally different way of delivering for our clients. In essence, we branded it a methodology that was innovative and bore little resemblance to how we were accustomed to working. We even went so far as to deliberately change familiar terminology as a means of underscoring how "different" and "better" our new methodology was. Now, it's frequently said that people produce the best results when they're performing work that they enjoy and that leverages all of their skills. However, it's very difficult to obtain a person's best work when his or her sense of understanding is undermined – i.e. don't pull the rug out from under the feet of your teams (especially if you expect them to be self-organising). The transition

process should be approached incrementally, as an evolutionary process and not as a break-point. We all can only deal with so much change at a time, so it's wise to, as some say, "eat your own dog food." By taking a deliberately agile approach to the introduction of agile practices, one can introduce change without being disruptive while calming fear and resistance along the way.<sup>3</sup> A piloting strategy is an important element of this approach, which we will discuss in greater detail later on.

## 2.5. Methodology is More Than Process

Recently, a large financial services client of ours related to us its experience with adopting agile methods. The client had surveyed the agile landscape, as we had, and settled on the common XP plus Scrum approach in response to an organisational mandate to reduce development cycle times as quickly as possible. They proceeded to send their developers and project managers to XP and Scrum trainings where their staff learned how to do such things as write unit tests first, program in pairs, conduct stand-up meetings and retrospect at each iteration's end. The results, however, were not a roaring success. Teams tripped over themselves and failed to demonstrate the return on investment that management had hoped for. The problem was not a lack of alignment or motivation on the part of the teams, nor was it an issue of management support. It was also not a problem of process; rather, it was a problem of organisation.

Whilst development and project management behaviours had changed, their operational context – their organisational structure – did not adequately change in response. While developers, testers and customers were working together in new ways, teams continued to operate in a "stove-piped" fashion, with development, quality assurance and application support each operating as silos. Likewise, the client found that despite formal training in agile processes, many of the new techniques failed to "stick." Many factors contributed to this. They say that "old habits die hard" and that certainly is true when it comes to software development. It's not uncommon for developers and project managers to slip back into comfortable, well-entrenched behaviours when

---

<sup>3</sup> We found the use of well-publicised "success stories" to be significantly important as an internal communications technique for maintaining momentum during our transition. Similarly, the broad communication of "lessons learnt" – essentially broadcasting the outputs of individual team retrospectives *beyond just the team* – was found to be of equal importance. Both of these tools help to demonstrate the value of agile methods whilst remaining open and honest about the challenges of enacting behavioural change.

situations become difficult. Many agilists might prescribe a coach as the panacea in these circumstances; however, this is often unfeasible in a large enterprise. Firstly, during a time of transition, highly experienced agile coaches tend to be in very short supply within the organisation (for obvious reasons). Given this lack of critical mass within the organisation, many companies begin to look outside themselves for this kind of support. This too is a problematic proposition when your chosen methodology is not a “by-the-book” approach, but an amalgamation of many different source methodologies specifically designed to thrive in your company.

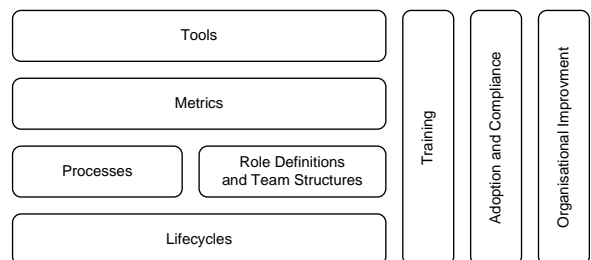
The example above provides an interesting illustration of how *not* to approach wide-scale deployment of agile methods within a large enterprise. What we found particularly troubling about this client’s experience, was the treatment of agile transition as a *process change event* as opposed to a *programme of organisational transformation*. We believe and our experience has shown that when preparing for such a transition, in addition to process, one must also consider the following:

- Roles, responsibilities and team structures must be aligned to the processes proposed. Some old roles may become obsolete and go away whilst others will change and new ones will appear. In particular, we found the absence of an architect role in XP to be ill-suited to our business. The size and complexity of many of our engagements, their tendencies to involve numerous integration partners and systems, and the reality that customers and end-users cannot always be accessible to the team prompted us to define a role. In S|A, the architect has explicit ownership for the overall integrity of the solution and responsibility for providing the team with advanced domain expertise. This is not to say that architects are not also “coaches” in the XP sense of the term; nor is our definition of architect one that implies the aloof ivory-tower expert, removed from his or her team.<sup>4</sup>
- Organisational structures must support cross-functionality. This may require disbanding functionally aligned, horizontal work groups (e.g. product planning, quality assurance, etc.)

and re-organising in a more vertically oriented fashion (e.g. along products lines of business).

- Organisational structures must allow teams to be self-organising. Working in a self-organising model does not mean that management is eliminated from the equation or kept out of close contact with the teams. Rather, it implies an environment pervaded by empowerment and trust. Senior managers must allow teams to make decisions on a day-to-day basis and teams are expected to pro-actively escalate issues and risks, when appropriate.
- Teams must be structured in a fashion that minimises time-traps and promotes information flow as a means for driving productivity. Team size and geographic distribution both complicate this. As a means of scaling teams, we use the concept of a *track* or sub-team. In practice, this is similar to a “Scrum of Scrums” or what Mike Cohn refers to as a “Meta-Scrum.” However, there is a right way and a wrong way to do this on a distributed team. Tracks which span multiple geographies, for instance, are a certain anti-pattern, as doing so introduces wait-states (as the result of distance, lower-bandwidth forms of collaboration, time differences, etc.) which increase drag and counteract team velocity. Taking a cue from lean principles, each track should ideally operate as an independent work cell that can be rapidly deployed, reconfigured and repurposed with all necessary resources in close proximity.

Thus, as shown in figure three, S|A consists not only of processes commonly associated with agile software development, but also of metrics, tools, trainings and organisational programmes to support adoption and standards compliance. We see all of these as essential elements that, when combined with guidance which informs their interactions, form the basis for an enterprise-capable agile methodology.



**Figure 3. Conceptual architecture of S|A**

<sup>4</sup> For a more detailed discussion on this topic, you can refer to: Gottesman, E., *Life Beyond XP?*, International Conference on Project Management Leadership, QAI India, Bangalore, India, 13-14 May, 2005.

### 3. Manage Adoption

Once the major architectural elements of an enterprise-agile methodology are in place, it's not as simple as 'flipping the switch.' The lack of a comprehensive adoption strategy can be crippling on a number of accounts. Failure to adequately set expectations amongst stakeholders at both the project and management levels can result in undue frustration and angst as projects go "off the rails" – as they can and will. Being up-front with your constituency and setting the expectation that there will be challenges helps to make some of the inevitable difficult conversations a little bit less difficult.

We have found adoption and change management to be one part of the transformation lifecycle that demands facilitation by a dedicated organisational function outside of the project teams themselves. Herein, we will simply refer to this organisational function as the *S|A Team*. As we explain how the S|A Team engages our project teams, we hope you will come to appreciate why this function is so essential within an enterprise context.

The S|A Team engages each project team at its inception as a means of helping the project get off to a successful start. However, that is not where their interaction ends. S|A consultants remain connected to project teams throughout the lifecycle. We see this as essential to growing organisational knowledge of what works and what doesn't work and feeding that knowledge back into the delivery engine. Herein we will talk in detail of what exactly happens during the S|A Team's first exchange with a project and what interactions follow from that. Throughout our discussion of agile adoption management, we invite you to align with our view that agile methods are not lacking in discipline. This assumption is key to understanding why we even bother talking about adoption management and agile in the same breath. Allow us to illustrate with a few brief examples:

- TDD carries with it the implication of 100% unit test coverage and a broken unit test is as good as a broken build – all development stops and the team converges on the issue and drives it to closure before moving on. Teams conduct themselves in this fashion in order to prevent the accumulation of "technological debt" which undermines maintainability and increases development costs. Thus, TDD is unquestionably one of the most rigorous software engineering techniques in use today.

- Agile teams follow strictly time-boxed iterations. Time-boxing versus scope-boxing ensures that teams produce a steady stream of customer value. Agile thought leaders caution teams against wild and arbitrary variations in iteration length as this disrupts the team's "rhythm" and makes velocity-based planning difficult (not to mention the frustration this creates for team members and customers alike).
- Pair programming has its own unique "protocol." Two people sit in front of a computer at the same time. One person always drives while the other observes. The observer thinks strategically about the software being developed and asks pertinent questions about the expressiveness of the code. They write unit tests first. They switch roles within pairs. They rotate between pairs frequently and regularly. Developers pair in this fashion as a means of injecting quality into everything they do. It serves as a form of continuous peer review, catching defects early and helping to maintain focus on simple design. If you are not doing all of these things, then you are not pair programming.

Because agile methods can be associated with clear behavioural expectations, we can codify and share them with each other. Articulating the *intent* of an agile practice is in itself a form of process reuse that leads to efficiencies. As suggested earlier, the responsibility for aligning behaviours amongst members of a team often falls upon the agile coach (at least until the team is well-versed enough that most or all its members can serve as models for these behaviours). However, relying only upon agile coaches for the purpose of "spreading the gospel" imposes significant limitations on the speed with which agile methods can be introduced and the extent to which they can be scaled within the organisation; this is also further complicated by geographic distribution. Thus, a more robust organisational infrastructure is needed to ensure broad and consistent dissemination of agile practices.

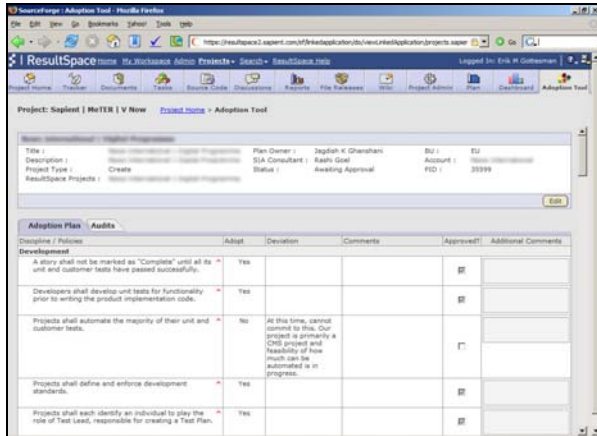
#### 3.1. Engage Teams Early

At the start of every Sapient project, a member of the S|A Team conducts a project kick-off meeting with representatives of the project team.<sup>5</sup> During this meeting they collaborate to create a *process adoption plan*. Each process adoption plan consists of

---

<sup>5</sup> These meeting may take place either in person or remotely with the support of teleconferencing services and virtual collaboration tools.

organisation-standard policies that capture the essence of S/A. Given the geographically distributed nature of Sapient, we built a web-based collaboration and workflow tool to facilitate this process and some of the critical conversation which accompanies it. Figure four show an example of what a team’s process adoption plan looks like within the tool.



**Figure 4. Example of a team’s S/A adoption plan in our custom-built adoption tool**

By engaging the project team right at its start, both project and organisation leaders gain essential insights into where peoples’ understanding of best practice is lacking *before the project gets substantially underway*. We have seen that when left to their own devices, projects run the risk of establishing processes that are either:

- inadequate (thus leading to project-level sub-optimisation);
- over-engineered (particularly in cases where most of the team is inexperienced in the use of agile methods or resistant to certain practices for one reason or another);
- significantly divergent from organisational norms, making it difficult for skills and experience obtained on one project to be effectively leveraged by another.

Most agile methods provide techniques for dealing with the first two of these outcomes (such as team retrospectives), but the third – that of organisationally divergent processes – is not something explicitly addressed in today’s agile body of knowledge. Further, this issue is of paramount concern to us as if left unchecked it can lead to organisation-level sub-optimisation. We argue that organisational sub-optimisation must be treated with equal if not greater

contempt than team sub-optimisation since the stakes are generally much higher. Senior management may be led to question further investment in agile techniques, and in the most extreme of circumstances, a failure to invoke some degree of practice standardisation can result in a deliberate backlash against agile methods and retreat into command-and-control models which provide the organisation with the “mirage of control” at great expense to innovation. Thus, by engaging project teams as they are just leaving the starting blocks, we identify knowledge and skills gaps early, when we can still bring to bear the full resources of the organisation. Wherever such gaps are identified, response plans utilising mentorship, formal training or even process automation can be initiated while such actions are still likely to meaningfully influence the project’s ultimate success.

### 3.2. Provide Teams with Options

We also recognise that one size really does not fit all, a view validated by a recent Forrester Research survey which showed that 80% of all large companies (those with market capitalisations greater than US \$1 billion) are unable to satisfy the needs of their development organisations with a single standardised methodology [2]. Thus, the S/A consultant works with the project members to tailor the policies to accommodate the project’s unique circumstances. By having these conversations, we challenge our project teams to think critically about what processes actually make sense for them. This is doubly important because first, it helps us as an organisation learn how our project teams are thinking about the processes and applying them and second, it results in a defined process that is truly *owned by the project team* – not by ivory-tower “process jockeys” who lack insight into the on-the-ground realities and constraints of the project.

### 3.3. Team Members as Change Agents

A sense of true ownership, beyond helping to avoid the hostility that can result from a team’s feelings that process is being “shovelled upon them,” also serves organisational needs for growth. By having these conversations with our project teams, we drive accountability for follow-through and mentorship. For instance, it’s not uncommon for teams to “push back” on the adoption of automated functional testing when their development consists primarily of COTS package configuration and integration. However, when a team signs on to do this we know that they’re likely to do a

fair bit of trailblazing. Over the course of the project, they'll face challenges in fully automating their functional testing and they may or may not get there 100%. Nonetheless, they are aligned around a common goal. In the course of working toward achievement of that goal they may produce reusable assets which can immediately benefit the organisation, but even if they don't, the mere experience they gain will help to grow the next generation of agile project leaders within the organisation. In essence, you will make champions out of your most vocal sceptics.

### 3.4. Succeed or Fail Fast – Everywhere

Most agile project managers will tell you, one of the greatest benefits of using agile methods is the ability to checkpoint progress early and frequently and make course corrections as needed. For instance, using the principle of *Yesterday's Weather*, we might determine that a team is unlikely to complete all of its committed stories by the end of the current release. Following this realisation, we might have a conversation with the customer (e.g. in an iteration-end checkpoint), and determine an appropriate course of action. We need to remain disciplined about time-boxing, so we might insist upon not adding additional iterations to the current release and pushing out the release date. Thinking in terms of the "iron triangle" of quality, schedule, and cost, what options are left? We don't want to compromise on quality so we are forced to consider cost. One reaction we've all probably experienced would be to increase the size of the team (a technique also known as "throwing bodies at a problem"). However, that has its own issues. First, the customer may be unwilling to pay for a larger team. Second, even if willingness to pay (either on the part of the customer or supplier) is not an issue, we cannot assume a larger team will be more productive. There has to be another option. We must recognise that cost is not connected only to people – it's also a function of process (among other things). Here, we look to the wisdom of the team retrospective. We reflect upon what is and is not working well and change course from a process standpoint (typically though not exclusively through the employ of automation). In doing so, we may find the savings necessary to put the project back on track. So, we are inspired to try different processes. Sometimes we make good choices and other times we don't. Either way we're going to get feedback quickly – i.e. by the end of the next iteration (if not sooner).

Many agile practitioners have warned against persons from outside of the project team attending team retrospectives, suggesting that this violates the

"safe space" principle which is central to ensuring participation and openness. We have taken a somewhat counter approach. Our conspicuous "open-door" policy provides the opportunity for S/A Team members to take part our project teams' retrospectives. Through this ongoing, active relationship, organisational knowledge is cultivated around what constitutes our canonical set of agile best practices. If something is found to work exceptionally well for one project, we have "eyes and ears" within the organisation that can pick up on that success and broadcast it to other teams. Similarly, we gain visibility into situations wherein certain practices are unsuccessful, unfeasible or fail or add commensurate value and can inject this knowledge, as appropriate, into our interactions with each project team we engage.

We have also found that changes come in all sizes. Some of the structural and process changes we've proposed have been so transformative that they could not be implemented as a single change. An early example of this was TDD, which can be excruciatingly challenging for a team if it is not practising continuous integration or its members lack sufficient experience in refactoring. Thus, introducing TDD actually involved introducing a number of related practices, some sequentially and others concurrently. Given the spectrum of technologies in use by our project teams we had to proceed with a plan. It is in these kinds of situations that a pilot strategy is of central importance. A pilot may be deemed appropriate if:

- the institutionalisation of the (process or structural) change is sufficiently complex;
- or further investigation is required in order to substantiate the change's benefits.

You can think of a pilot as a spike solution for your organisation – a time-limited experiment that will produce knowledge and help to validate a new operational concept's value (or lack thereof). Pilot duration should be just barely sufficient to satisfy the needs of the organisation. Smaller changes may be possible to evaluate within the span of a single iteration using one lone project. Larger changes may require multiple iterations and multiple concurrent pilots. Piloting is an inclusive process that requires openness and buy-in for success. All stakeholders should make clear before the pilot begins their expectations and definitions of what will constitute a successful outcome. Once the pilot is in progress, you must listen to what the piloting team has to say since they're the one who are "feeling" the impact of the change. Further, wherever possible objective measures

can help make the pilot outcomes less open to interpretation and dispute.

## 4. Sustain Change

When we say “sustain change” we are perhaps inviting criticism for speaking in contradictory terms. On one hand we suggest that new-found agility must be nurtured and kept alive, whilst on the other hand we suggest moving on. We invite you to look beyond this oxymoron and consider how a “sustain change” attitude is inherent to agility and essential in an enterprise context.

When teams buy into implementing defined processes, they provide a set of baseline expectations against which their performance can be gauged. Periodic assessments of actual practice provide opportunities to judge conformance to standards, identify improvement opportunities, and extract best practices. Importantly, such assessments may be used to reward teams and reinforce good behaviours (but should not be used for disciplinary purposes). What we are of course suggesting here, is a system of ongoing compliance and process improvement like that embodied in models such as CMMI, the mere mention of which is probably enough to send some agile purists running for cover (or running to get their guns). In this section, we describe how CMMI-based continuous improvement practices coexist with agile methods in S|A and why this juxtaposition is of immense importance for a large, distributed organisation.

### 4.1. Audit without Angst

More often than not, audits conjure up images of the “process police” – an army of grim-faced, clipboard-wielding drones that are, at best, a slight distraction to teams and at worst, a significant drain on the teams’ energy and productivity. Thus, the relationship between teams and their auditors is generally cool if not downright adversarial. However, audits can prove valuable to both project teams as well as the organisation-at-large if approached with a collaborative problem-solving spirit.

At Sapien, project audits are conducted on a monthly basis (which happens to align with the allowed upper limit on iteration length in S|A). To remove some of the stigma associated with audits, we treat them as activities of retrospection, current-practice assessment and forward-looking improvement. They are not (and must not) devolve into a blame game. Early on we struggled with implementing our desired model as we had staffed our adoption and

compliance team (a track of the S|A Team) with newly hired “quality professionals.” These individuals came from a traditional software quality assurance (SQA) background. Whilst they were well-versed in the procedures of audits due to their professional exposure to quality frameworks like ISO and CMMI, they brought with them the unfortunate view that development and SQA should be forever separate with a thick wall between them to preserve objectivity. As you might imagine, this led to some rather heated arguments, requiring senior management intervention to diffuse the situation. Since then, we have fully reconstituted this team, leveraging persons with more varied backgrounds – a combination of traditional SQA and delivery experience, new hires and Sapien veterans. Nonetheless, the “quality professionals” did get one thing right: objectivity is important (even if they went about achieving it in an unproductive way). A string of recent posts to the Agile Management email group has significantly progressed this discussion within the agile community [4]. One contributor, for example, proposed that project managers regularly observe one another’s projects and debrief on what was seen. This certainly suggests value in obtaining an objective “second opinion.” Others responded favourably to this and similar posts, implying that objective, fact-based discussion concerning a project’s implementation of agile practices in relation to standard of behaviour is indeed compatible with the values enshrined in the Agile Manifesto and does not suggest a shift away from individuals and interactions in overwhelming favour of processes and tools. We maintain that audits are a necessary function in any large and growing organisation choosing to adopt agile methods.<sup>6</sup> However, to be effective, agile audits must:

- view auditors within the organisation as trusted “players” – i.e. that they are pragmatic and not dogmatic in their approach
- allow auditors and project team members to “meet on even ground” and are equally empowered to escalate in instances of disagreement;
- leave the inference of fault at the door;

---

<sup>6</sup> Elsewhere, we hypothesize the conditions under which the need for audits would be obviated in an enterprise context. We have chosen not to discuss this here as we feel those conditions remain a distant reality for most large organisations. For more on this topic, see: Gottesman, E. *Driving Agile Adoption Through Audits*, Agile Journal, CMC Media, Inc., 8 September, 2006

- and focus on uniformity of practice as a means of driving consistency in customer experience and not an end in and of itself.

## 4.2. Metrics Matter

Metrics are not foreign to agile teams. In fact, they are of central importance to many agile practices. Unit test coverage, for instance, is a useful measure of a team's discipline around TDD. Similarly, velocity and effort burn-down are indispensable tools for the agile project manager. However, there appears to be a tendency amongst agilists to reject many software project metrics that are equally applicable to traditional waterfall projects and agile project alike. Similarly, agile purists eschew formal measurement and analysis frameworks and statistical quality management techniques (e.g. Six Sigma) as burdensome, heavyweight process overhead that adds no customer value. However, in reality, metrics do count (no pun intended). Customers care about metrics and organisations do too. Customers are interested in a array of measures beyond just velocity – defect trends, incident resolution times and availability stats just to name a few. Likewise, organisations interested in maintaining their competitive edge should be cognisant of metrics like productivity, estimation accuracy and customer satisfaction. As we see it, the problem is not with metrics. You do need to identify the right ones – the ones that matter to you based on your organisational objectives – but so much work has been done on that already that you really don't need to look too hard (hint: we've already mentioned some the really "big" ones). The issue, in our view, has much more to do with the perception that they are a burden and will be used against the interests of the project when it suits the customer (or the organisation). On the matter of metrics being a burden, we feel strongly that agile teams must come of age and move beyond some of the simple tools they have become so attached to. Sticky notes and whiteboards indisputably have their place in an agile project environment and we hope they're here to stay. However, these tools are inadequate in a distributed enterprise context. Even spreadsheet-based tools fall short of providing the levels of integration, automation and accessibility we require. ALM 2.0 offers a much more complete vision of the tooling capabilities necessary to enable agile methods to flourish in an enterprise context [5, 6]. As for the suggestion that metrics expose the project team to abuse from customers or management, let us simply say that whatever abuse comes as a result of metrics can be no worse than what empirical observation and

subjective perception could dish out (and at least in the case of metrics you know that it's coming).<sup>7</sup>

## 5. Summary

Awareness of agile methods has significantly increased in recent years, yet the data suggests that many organisations remain cautious when it comes to the question of adoption [2]. Why is this and does it not serve our interests to introspect upon how the agile movement may have contributed (knowingly or unknowingly) to the industry's slow up-take? We firmly believe this is a useful line of inquiry. Over the past few years we have internally grappled with the challenges of institutionalising agile methods. We've also had countless conversations with our clients to "sell" them on the idea that agile methods work and deliver key benefits above would otherwise be possible. In both cases, it hasn't been easy. Organisational inertia certain plays a role, but what we have now come to believe (as should be evident to you the reader), is that we need to pick up and move forward from where agile methods left off. Whether you consider this part of agile's ongoing evolution or the dawn of a new fluid discipline, there's work to be done. In order to remain relevant in a competitive enterprise environment, agile methods must be adapted and augmented to effortlessly accommodate scale, globally distributed teams and continuous improvement at an organisational level. What we can and must do as software development professionals is embrace syncretism. We need to explore all aspects of the problem – in this case, the problem of maximising customer satisfaction within a complex environment of stakeholder needs and constraints – merge several originally discrete disciplines or traditions, and in doing so, assert an underlying unity. By reconciling such disparate or even opposing forces as agility, distributed delivery, ALM, standardisation, compliance, statistical management and lean, we will ultimately arrive at a more powerful and flexible solution provided we keep an open mind.

## 6. References

- [1] The Standish Group, *CHAOS Chronicles*, 2004.

---

<sup>7</sup> This touches upon the topic of transparency as a means of engendering trust. While we are tempted to explore how this manifests in agile methods through such things as XP's "big visible charts," we'll resist for now and save this topic for another time.

[2] Schwaber, C., *What's New In Application Development Processes and Methodologies*, Forrester Research, Inc., 14 September, 2006.

[3] *Business Technographics® North American and European Enterprise Software and Services Survey*, Forrester Research, Inc., November 2005.

[4] Agile Management email group,  
<http://groups.yahoo.com/group/agilemanagement/>

[5] Parker, K., *ALM 2.0: Application Development On The Cusp*, Agile Journal, CMC Media, Inc., 8 September, 2006.

[6] Schwaber, C., *The Changing Face Of Application Life-Cycle Management*, Forrester Research, Inc., 18 August, 2006.

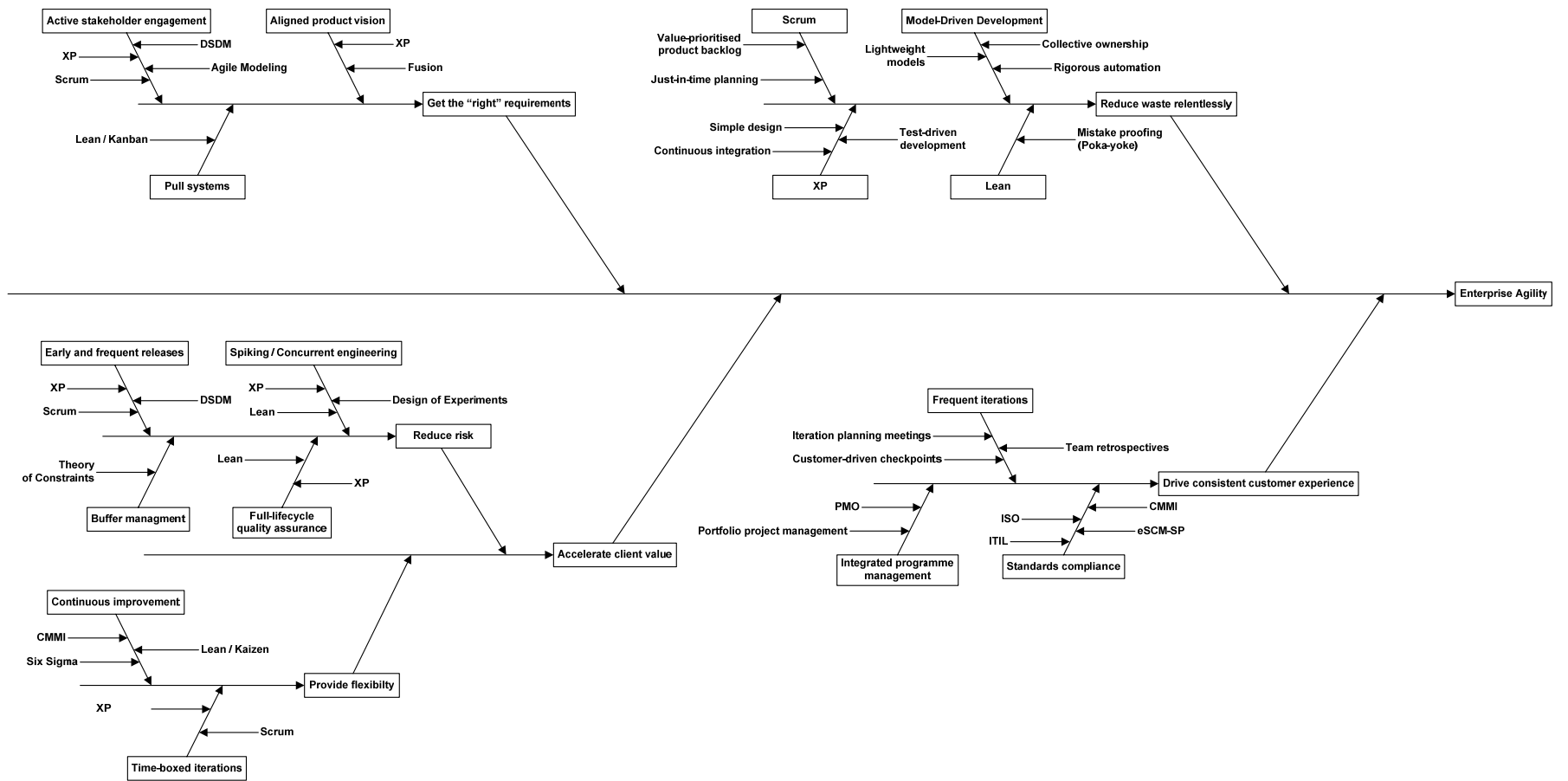


Figure 5. Diagrammatic view of practices, their outcomes and how they contribute to enterprise agility