Agility @ Scale Whitepaper
February 2010

IBM agility@scale™

Rational. software

Scaling Agile: An Executive Guide

Scott W. Ambler
Chief Methodologist for Agile, IBM Rational

Contents

Executive summary

Agile software development is a highly collaborative, quality-focused approach to software and systems delivery, which emphasizes potentially shippable working solutions produced at regular intervals for review and course correction. Built upon the shoulders of iterative development techniques, and standing in stark contrast to traditional serial or sequential software engineering methods, agile software delivery techniques hold such promise that IBM has begun to adopt agile processes throughout its Software Group, an organization with over 25,000 developers. But how can practices originally designed for small teams (10-12) be "scaled up" for significantly larger operations? The answer is what IBM calls "agility@scale."

There are two primary aspects of scaling agile techniques that you need to consider. First is scaling agile techniques at the project level to address the unique challenges individual project teams face. This is the focus of the Agile Scaling Model (ASM). Second is scaling your agile strategy across your entire IT department, as appropriate. It is fairly straightforward to apply agile on a handful of projects, but it can be very difficult to evolve your organizational culture and structure to fully adopt the agile way of working.

The Agile Scaling Model (ASM) defines a roadmap for effective adoption and tailoring of agile strategies to meet the unique challenges faced by a software and systems delivery team. Teams must first adopt a disciplined delivery lifecycle that scales mainstream agile construction techniques to address the full delivery process, from project initiation to deployment into production. Then teams must determine which scaling factors – team size, geographical distribution, regulatory compliance, domain complexity, organizational distribution, technical complexity, organizational complexity, or enterprise discipline, if any – are applicable to a project team and then tailor their adopted strategies accordingly to address their specific range of complexities.

When scaling agile strategies across your entire IT organization you must effectively address five strategic categories – the Five Ps: People, principles, practices, process, and products (i.e., technology and tooling). Depending on your organizational environment the level of focus on each area will vary. What we are finding within many organizations, including IBM, is that the primary gating factor for scaling agile across your entire organization is your organization's ability to absorb change.

Highlights

In the IBM Rational organization, we've used agile and iterativeIn the IBM Rational organization, we've used agile and iterative techniques internally for many years, and the IBM Global Services and Rational organizations have been working together to help many of our customers apply these techniques within their own environments.

Agile approaches are being used in a wide range of situations, not just the small, co-located team environments that dominate the early agile literature.

Introduction

Agile software development is an evolutionary, highly collaborative, disciplined, quality-focused approach to software development and delivery, whereby potentially shippable working software is produced at regular intervals for review and course correction. Agile software development processes[1] include Scrum, Extreme Programming (XP), Open Unified Process (OpenUP), agile instantiations of Rational Unified Process (RUP), and Agile Modeling (AM), to name a few. In the IBM Rational organization, we've used agile and iterative techniques internally for many years, and the IBM Global Services and Rational organizations have been working together to help many of our customers apply these techniques within their own environments, often under complex conditions at scale. Agile techniques held such promise that beginning in mid-2006 an explicit program was put in place to adopt these processes on a wide-scale basis throughout IBM Software Group, an organization with over 25,000 developers.

Agile software development techniques have taken the industry by storm, with 76% of organizations reporting in 2009 that they had adopted agile techniques, and that on average 44% of the project teams within those organizations had adopted one or more techniques [1]. Agile development is becoming widespread because it works well – organizations are finding that agile and iterative project teams, when compared to traditional project teams, enjoy higher success rates, deliver higher quality, have greater levels of stakeholder satisfaction, provide better return on investment (ROI), and deliver systems to market sooner [2]. By following quality techniques such as refactoring and developer regression testing throughout the lifecycle, agilists are able to progress safely and surely, increasing their productivity. By working closely with stakeholders in an iterative manner they have a better understanding of what stakeholders actually need and are more likely to deliver solutions that people actually want to use for their business purposes. By working in priority order, agile teams are able to provide the greatest return on investment as defined by their stakeholders. In short, agile teams work smarter, not harder, and thereby achieve better results.

As you will learn later in this paper, agile approaches are being used in a wide range of situations, not just the small, co-located team environments that dominate the early agile literature.[2] Agile strategies are being applied throughout the entire software delivery lifecycle, not just the construction (software coding and compiling) phase, and very often in very complex environments that require far more than a small, co-located team armed with a white board or a stack of index

Highlights

cards. Every project team finds itself in a unique situation, with its own goals, abilities, and challenges. What they have in common is the need to adopt, and then tailor, agile methods, practices, and tools to address those unique situations.

This paper looks at our experiences gained while applying agile/iterative strategies and techniques in organizations around the world, often at a scale far larger than the techniques were pioneered for. It begins with our definition of what it means to be agile; it summarizes the Agile Scaling Model (ASM) and explores the scaling factors which your project teams often face; it provides advice for how to adopt agile strategies across your entire IT department; and ends with a discussion of the types of benefits which you may expect to achieve by doing so.

Defining agile

Many people point to the value statements of the Agile Manifesto[3] as a definition for agile development. Although these values are very good foundational philosophies, they were never really meant to be a definition. In fact, the agile community has never really settled on a definition nor does it appear that they will do so any time soon. The Rational organization has its own description for what we call "disciplined agile delivery":

Disciplined agile delivery is performed in a highly collaborative, disciplined, and self-organizing manner within an appropriate governance framework, with active stakeholder participation.

> *Disciplined agile delivery is an evolutionary (iterative and incremental) approach that regularly produces high-quality solutions in a cost-effective and timely manner via a risk and value-driven lifecycle. It is performed in a highly collaborative, disciplined, and self-organizing manner within an appropriate governance framework, with active stakeholder participation to ensure that the team understands and addresses the changing needs of its stakeholders. Disciplined agile delivery teams provide repeatable results by adopting just the right amount of ceremony for the situation which they face.*

Here is a more concise though less robust definition:

> *Disciplined agile delivery is a highly collaborative, evolutionary, self organizing, and governed approach that regularly produces high-quality solutions in a cost-effective and timely manner via a risk and value driven lifecycle.*

I'll return to the elements of this definition a bit later.

Highlights

Undisciplined "ad-hoc" teams often run into trouble, and give actual agile teams a bad name.

Because every team finds itself in a unique situation, they must be flexible in the way that they assess their agility. The real goal is to be as effective as possible given the situation.

Criteria to determine if a team is agile

A common problem in many organizations is that undisciplined "ad-hoc" teams often claim to be agile, because they've read an article or two about agile development, and interpret agility to mean any cool, liberated form of undocumented software creativity. These ad-hoc teams often run into trouble, and give actual agile teams a bad name. IBM Rational defines the following five criteria to determine if a team is truly agile:

1. *Working software* - Agile teams produce working software on a *regular basis*, typically in the context of short, stable, time-boxed iterations.
2. *Active stakeholder participation* - Agile teams work *closely* with their stakeholders, ideally on a daily basis.
3. *Regression testing* - Agile teams do, at a minimum, *continuous* developer regression testing.[4] Disciplined agile teams take a Test-Driven Development (TDD) approach.
4. *Organization* - Agile teams are self-organizing, and disciplined agile teams work within an *appropriate* governance framework at a sustainable pace. Agile teams are also cross-functional "whole teams," with enough people with the appropriate skills to address the goals of the team.
5. *Improvement* - Agile teams *regularly* reflect on[5], and disciplined teams also measure, how they work together and then act to improve on their findings in a *timely* manner.

An important aspect of these criteria is that they are flexible. Note the terms used in the description of the criteria – regular basis, closely, continuous, appropriately, regularly, timely; they are all situational in nature. For example, for some teams "regular basis" might be once every week, for other teams in more complex situations once every six weeks. Because every team finds itself in a unique situation, they must be flexible in the way that they assess their agility. The real goal is to be as effective as possible given the situation.

Highlights

Whether you are on a mainframe team writing COBOL code for a bank, software running on millions of mobile phones, or e-commerce code running on the Web, your team should still follow a full delivery lifecycle in a disciplined manner.

Scaling agile strategies at the project level

The Agile Scaling Model (ASM) [3] is a contextual framework for effective adoption and tailoring of agile practices to meet the unique challenges faced by a system delivery team of any size. Figure 1 overviews the ASM, depicting how the ASM distinguishes between three scaling categories: Core agile development, disciplined agile delivery, and agility at Scale. IBM Rational advocates disciplined agile delivery as the minimum that your organization should consider if it wants to succeed with agile techniques – whether you are on a mainframe team writing COBOL code for a bank, software running on millions of mobile phones, or e-commerce code running on the Web, your team should still follow a full delivery lifecycle in a disciplined manner. You may not be there yet, still in the learning stages. But our experience is that you will quickly discover how one or more of the scaling factors is applicable, and as a result need to change the way you work.

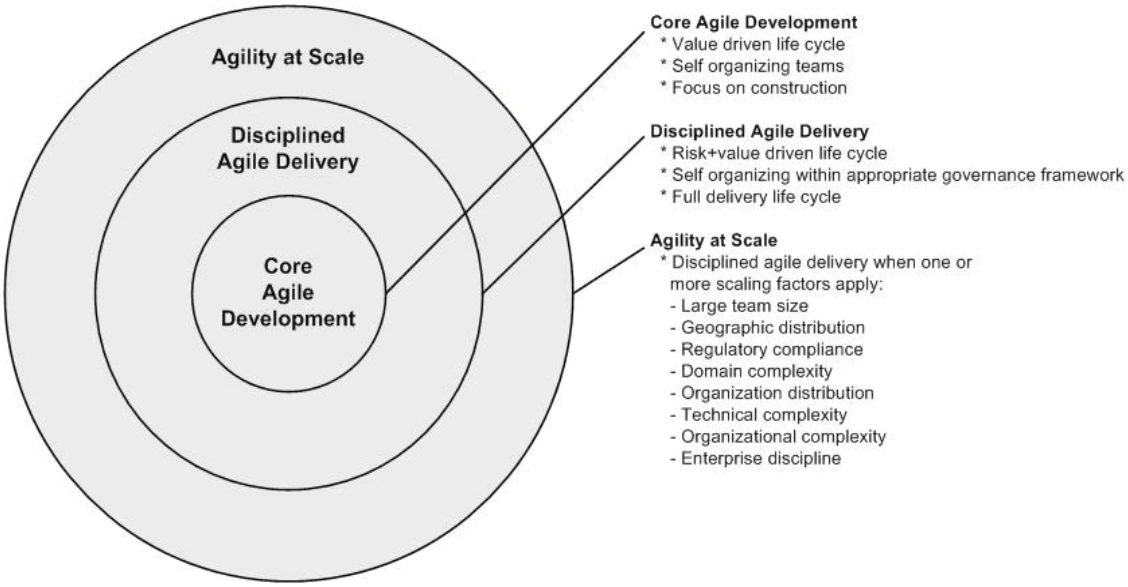

Fig 1. Overview of the Agile Scaling Model (ASM) [6]

Highlights

The first step in scaling agile approaches is to move from partial methods to a full-fledged, disciplined agile delivery process. Mainstream strategies (such as Extreme Programming (XP) or Scrum, which the ASM refers to as core agile development strategies) are never sufficient on their own.

The first step in scaling agile approaches is to move from partial methods to a full-fledged, disciplined agile delivery process. This is a theme echoed by the Software Engineering Institute (SEI) in their work on applying agile and Capability Maturity Model Integration (CMMI) together [20]. Mainstream agile development processes and practices, of which there are many, have certainly garnered a lot of attention in recent years. They've motivated the IT community to pause and consider new ways of working, and many organizations have adopted and been successful with them. However, these mainstream strategies (such as Extreme Programming (XP) or Scrum, which the ASM refers to as core agile development strategies) are never sufficient on their own.

To recognize why this is so, compare the Scrum lifecycle of Figure 2 with the disciplined agile delivery lifecycle [4] of Figure 3. In addition to using sensible terminology (for example, nobody "sprints" through a 10 kilometer race), the disciplined agile delivery lifecycle expands upon the Scrum lifecycle in three important ways:

1. It has explicit project phases, recognizing that agile delivery is really iterative in the small and serial in the large [5] – Figure 3 explicitly recognizes that there is additional effort to coordinate teams at inception and additional effort to package/transition and release the product to production which the Scrum lifecycle of Figure 2 doesn't take into account.
2. It specifies practices as well as the project management framework. Scrum components/aspects of the project management framework and leaves practice selection to the teams. Disciplined agile includes initial requirements and architecture envisioning at the beginning of the project to increase the chance of building the right product in the right manner as well as system release practices.
3. It includes more robust practices. The lifecycle of Figure 3 explicitly reworks the product backlog of Figure 2 into the more accurate concept of a ranked work item list. Not only do delivery teams implement functional requirements, they must also fix defects (found through independent testing or by users of existing versions in production), provide feedback on work from other teams, take training courses, and so on. Instead of leaving these issues up to the development teams to work through, disciplined teams start with a strategy which addresses them from the very beginning.

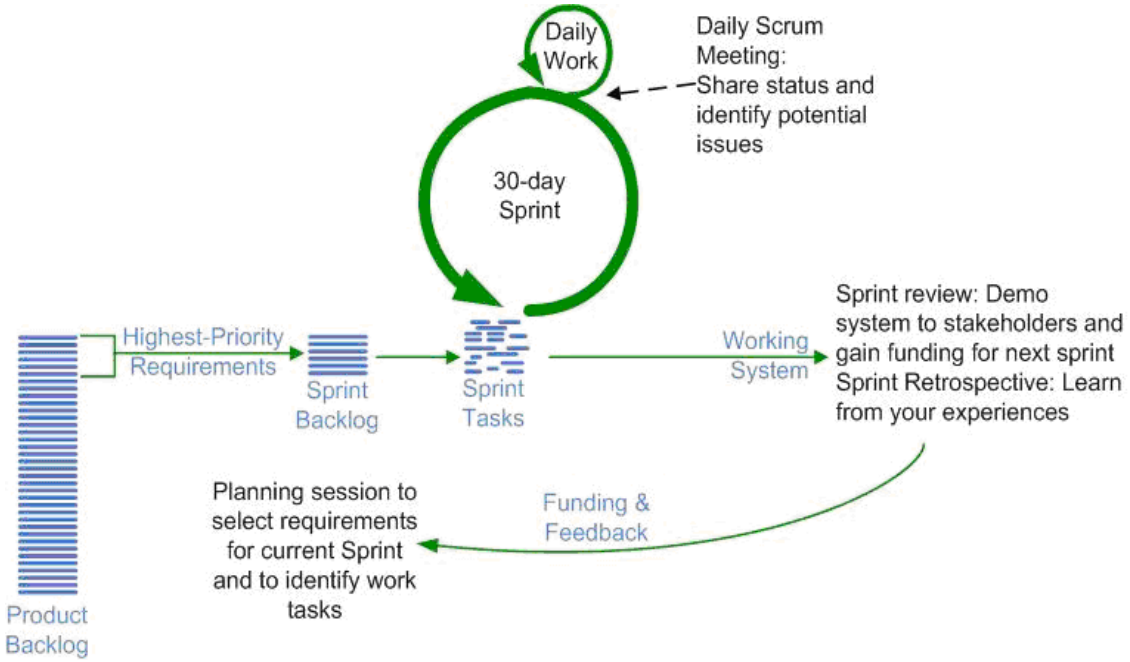

Fig 2. Scrum construction lifecycle

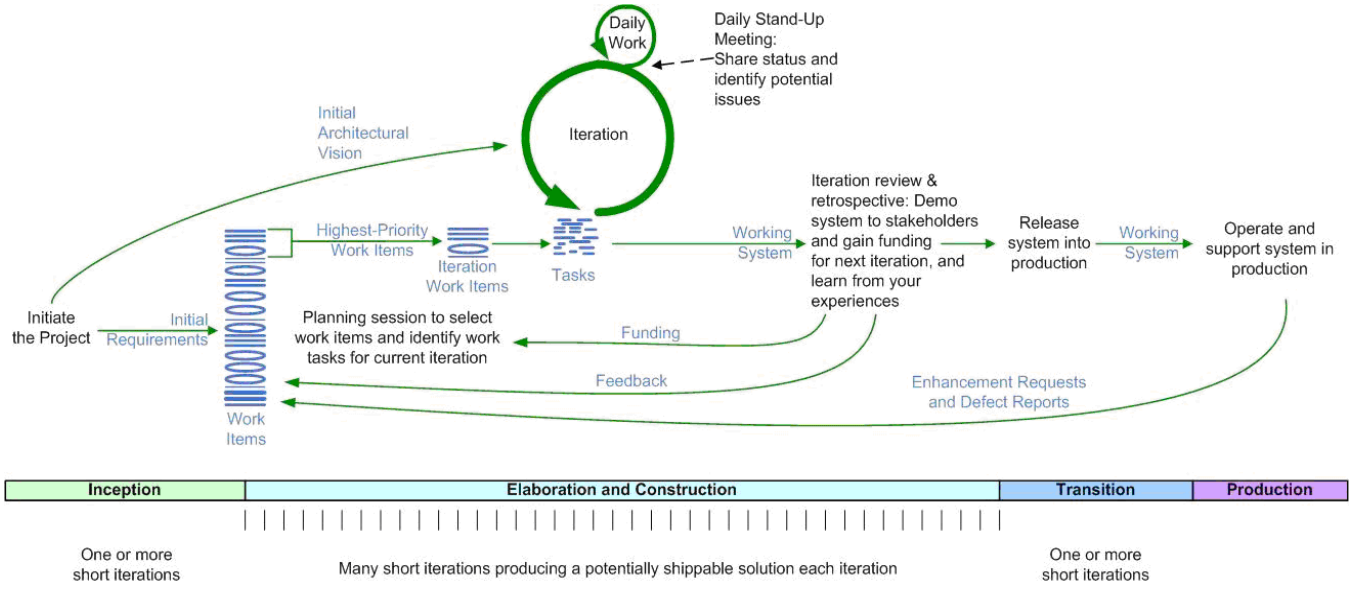

Fig 3.Agile system delivery lifecycle

Highlights

Many organizations will develop their own disciplined agile delivery process(es) by combining Scrum, practices from XP, and (sometimes unknowingly) practices from other processes such as Agile Modeling. This strategy works, although it can be expensive and time consuming compared to starting with a full disciplined agile delivery process. Many teams I've worked with would have greatly benefited by starting with an existing, more disciplined process such as the Open Unified Process (OpenUP), Dynamic System Development Method (DSDM), or the Eclipse Way, all widely available methodologies for team-based software delivery.

In the early days, projects managed via agile techniques were small in scope and relatively straightforward. Today, the picture has changed significantly, and larger organizations want to apply agile development to a broader set of projects.

The second step to scaling agile is to assess the degree of complexity your team faces. In the early days, projects managed via agile techniques were small in scope and relatively straightforward. Small, co-located teams using mainstream processes still get the job done in these situations. Today, the picture has changed significantly, and larger organizations want to apply agile development to a broader set of projects. They require large teams; they want to leverage a distributed work force; they want to partner with other organizations; they need to comply with regulations and industry standards; they have significant technical or cultural environmental issues to overcome; and they want to go beyond the single-system mindset and truly consider cross-system enterprise issues effectively. Not every project team faces all of these scaling factors to the same extent, but these are the primary factors that add complexity to your situation. That's why your disciplined agile delivery process needs to adapt.

In addition to scaling your lifecycle to address the full range of needs for solution delivery, there are eight more scaling factors that may be applicable, as shown in Figure 4. Each factor represents a range of possibilities, from simple to complex. For each factor the simplest situation is on the left-hand side and the most complex situation on the right-hand side. When a project team finds that all eight factors are close to the left (simple), then their project can be managed in a disciplined agile delivery mode. But when one or more scaling factors moves to the right, they are in an agility at scale situation. To address these scaling factors you will need to tailor your disciplined agile delivery practices and in some situations adopt a handful of new practices to address the additional risks that you face at scale.

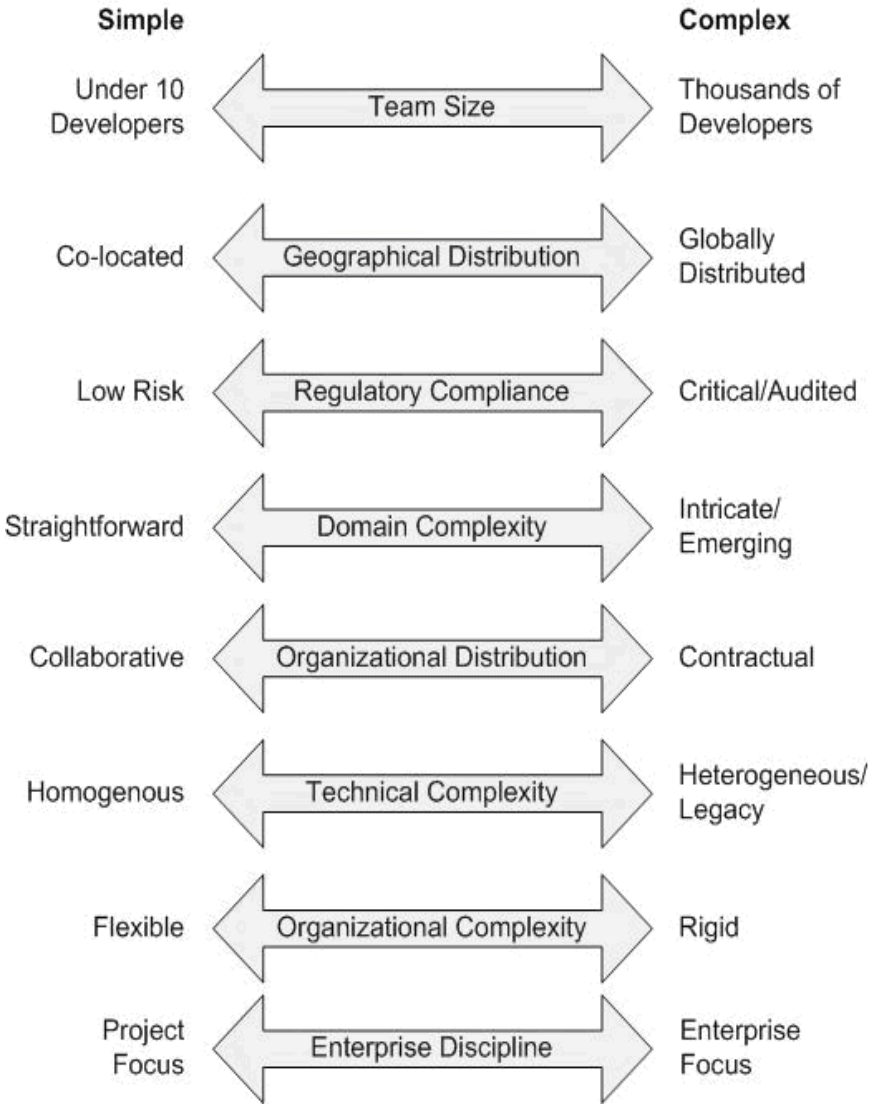

Fig 4.Potential scaling factors for disciplined agile delivery

Within the range of complexities shown in Figure 4, teams will need to tailor practices and tools to reflect their situation. The first four scaling factors listed – team size, geographical distribution, regulatory compliance, and organizational distribution – are relatively straightforward to address via disciplined work, adoption of appropriate technology, and tailoring of practices to reflect the realities of each scaling factor. The other four scaling factors – domain complexity, technical complexity, organizational complexity, and enterprise discipline – are more difficult to address because environmental complexity often reflects systemic challenges within your organization and enterprise discipline requires a level of maturity that many organizations struggle to achieve (although most desire such discipline).

Highlights

While it may be tempting to begin adopting agile techniques via a small pilot project or two, the best way to ensure success is to assign a crack team with a mission critical project using agile/iterative techniques.

I've found that to be successful scaling agile techniques across your entire IT department that you must address five areas, what I call the "5 Ps" of IT: people, principles, practices, products, and processes.

Scaling agile across your entire IT department

While it may be tempting to begin adopting agile techniques via a small pilot project or two, Walker Royce, Chief Software Economist at IBM Rational offers a bold suggestion: *The best way to ensure success is to assign a crack team with a mission critical project using agile/iterative techniques.* Perhaps this explains much of the success of disciplined agile teams. However, successful process improvement across all or most of an IT entire organization can prove difficult in practice, often because casting a wider net draws a wider range of challenges.[7]

I've found that to be successful scaling agile techniques across your entire IT department that you must address five areas, what I call the "5 Ps" of IT: people, principles, practices, products, and processes.[8] Here they are, in order of importance:

1. *People* - People and the way they work together have a greater effect on the outcomes of a project than the processes they're following or the products (tools and technologies) that they're using.
2. *Principles* - An effective set of principles, some organizations use the term philosophies, provides a foundation to help you keep things together even when the environment is shifting underneath you.
3. *Practices*- A practice is a self-contained, deployable component of a process [14]. Examples of agile practices include test-driven development (TDD), daily stand-up meetings, requirements envisioning, database refactoring, continuous integration, shared vision, and user-story driven development to name a few. The prevailing strategy within the agile community is for project teams to adopt and then tailor these small, cohesive practices to meet the unique needs which their project team finds themselves in.
4. *Products* - The IBM Jazz platform (www.jazz.net) provides a tailorable tooling eco-system which reflects the realities of agility at scale. Although simple, point-specific products work well in the straightforward situations faced by disciplined agile delivery teams, such teams working at scale quickly find that they need integrated tools which support collaboration and which are instrumented to support automated reporting (to support appropriate governance).
5. *Processes* - The previous 4Ps do not exist in a vacuum, we need some sort of glue to help piece all of this together. Minimally this glue is a lifecycle, such as the one in Figure 3, although more often than not it is a process or method.

Highlights

We help the organization make the appropriate improvements by leveraging training materials, process definitions, and tooling guidance, all of which can be tailored to the unique needs of an organization to give them a head start on their process improvement efforts.

Within IBM, we've found that teams who explicitly track their progress at adopting improvements are more successful than those who don't.

Understanding the five Ps of IT, and being prepared to address them is a good start, but note that any medium to large organization is doing many things in parallel, making planning and coordination difficult. IBM Rational takes a measured improvement approach to help organizations improve their system delivery effectiveness. This strategy typically includes an initial "health check" assessment called which helps you to navigate and select the right subset of practices, define your current capability (an "as-is" measure), a target capability improvement (a "to-be" measure), and a roadmap for you to get from where you are today to your target improvement with measurable feedback all along the route. We then help the organization make the appropriate improvements by leveraging training materials, process definitions, and tooling guidance, all of which can be tailored to the unique needs of an organization to give them a head start on their process improvement efforts. The measured improvement approach is intended to resolve the two predominant failure patterns of past process improvement initiatives, either self-inflicting too much process (rather than a subset of incremental practices) or employing subjective rather than objective measures of progress.

You must also explicitly manage your process improvement efforts. It's fairly easy to succeed at a handful of pilot projects; it's a bit more difficult to permanently adopt meaningful process improvements across your IT organization. A common strategy is for a team to regularly reflect on their approach to identify potential improvements, and then hopefully act on those improvements. Within IBM, we've found that teams who explicitly track their progress at adopting improvements are more successful than those who don't. We've developed tooling called IBM Rational SelfCheck which helps teams do exactly this [15]. Reflections/retrospectives at the team level work well in practice, but you need more at the IT department level. You also need a funded, continuous improvement program across your delivery organization that leverages a mix of iteration reflections and practitioner input to constantly improve your baseline agile delivery process. Without that, agile teams in various lines of business may re-invent the wheel and go through unnecessary pain.

Highlights

Many organizations are starting with lean principles to provide a guiding foundation for delivery efforts.

The relationship between Agile and Lean

As I discussed earlier, you want to adopt a set of principles that reflect your unique situation to provide a guiding foundation for your delivery efforts. Many organizations are starting with lean principles to provide such guidance. In *Implementing Lean Software Development* [10], Mary and Tom Poppendieck show how the seven principles of Lean Manufacturing can be applied to optimize the whole IT value stream. These principles are:

1. *Eliminate waste* - Lean thinking advocates regard any activity that does not directly add value to the finished product as waste. The three biggest sources of waste in software development are the addition of unrequired features, project churn and crossing organizational boundaries (particularly between stakeholders and development teams). To reduce waste it is critical that development teams be allowed to self organize and operate in a manner that reflects the work they're trying to accomplish. Walker Royce argues in "Improving Software Economics" [21] that the primary benefit of modern iterative/agile techniques is the reduction of scrap and rework late in the lifecycle.
2. *Build in quality* - Your process should not allow defects to occur in the first place, but when this isn't possible you should work in such a way that you do a bit of work, validate it, fix any issues that you find, and then iterate. Inspecting after the fact, and queuing up defects to be fixed at some time in the future, isn't as effective. Agile practices which build quality into your process include test driven development (TDD) and non-solo development practices such as pair programming and modeling with others.
3. *Create knowledge*- Planning is useful, but learning is essential. You want to promote strategies, such as iterative development, that help teams discover what stakeholders really want and act on that knowledge. It's also important for a team to regularly reflect on what they're doing and then act to improve their approach.
4. *Defer commitment* - It's not necessary to start software development by defining a complete specification, and in fact that appears to be a questionable strategy at best [11]. You can support the business effectively through flexible architectures that are change tolerant and by scheduling irreversible decisions to the last possible moment. Frequently, deferring commitment requires the ability to closely couple end-to-end business scenarios to capabilities developed in multiple applications by multiple projects.

Highlights

5. *Deliver quickly* - It is possible to deliver high-quality systems quickly. By limiting the work of a team to its capacity, which is reflected by the team's velocity,[9] you can establish a reliable and repeatable flow of work. An effective organization doesn't demand teams do more than they are capable of, but instead asks them to self-organize and determine what they can accomplish. Constraining these teams to delivering potentially shippable solutions on a regular basis motivates them to stay focused on continuously adding value.
6. *Respect people* - The Poppendiecks also observe that sustainable advantage is gained from engaged, thinking people. The implication is that you need a governance strategy that focuses on motivating and enabling IT teams—not on controlling them [12].
7. *Optimize the whole* - If you want to be effective at a solution you must look at the bigger picture. You need to understand the high-level business processes that individual projects support—processes that often cross multiple systems. You need to manage programs of interrelated systems so you can deliver a complete product to your stakeholders. Measurements should address how well you're delivering business value, because that is the sole reason for your IT department.

Agile Modeling's practices of light weight, initial requirements envisioning followed by iteration modeling and just-in-time (JIT) model storming work because they reflect deferment of commitment regarding what needs to be built until it's actually needed.

Lean thinking is important to agility in several ways. First, lean provides an explanation for why many of the agile practices work. For example, Agile Modeling's practices of light weight, initial requirements envisioning followed by iteration modeling and just-in-time (JIT) model storming work because they reflect deferment of commitment regarding what needs to be built until it's actually needed, and the practices help eliminate waste because you're only modeling what needs to be built. Second, these principles offer insight into strategies for improving your software process. For example, by understanding the source of waste in IT you can begin to identify it and then eliminate it. Third, these principles provide a philosophical foundation for scaling agile approaches. Fourth, value stream mapping – a technique common within the lean community whereby you model a process and then identify how much time is spent on value-added work versus wait time – helps calculate overall time efficiency of what you're doing. Value stream maps are a straightforward way to illuminate your IT processes, providing insight into where significant problems exist.[10]

Highlights

It is best for teams actually involved with the details of project progress to measure key improvement factors and not simply rely on gut feel (and not rely on outside consultants hired to achieve the benefits!)

What improvements should you realistically expect?

I am often asked what kind of benefits to expect from adopting agile approaches. While the surveys[11] I have conducted over the years consistently show that agile and iterative approaches provide better quality, greater stakeholder satisfaction, better return on investment, and better time to value than traditional techniques, [2] I am invariably asked: "How much better?" and "How much will we benefit?" As you may suspect, I can't provide exact answers to these questions, but I can provide a framework for understanding the potential benefits of adopting agile across your IT organization.

You've probably heard some "wild claims" in agile case studies of productivity improvements of 50%, 75%, 100%, and sometimes more. If in your organization you see lower productivity improvements than this, don't worry; this doesn't suggest failure. Although I have no doubt that many of these claims are reasonably correct (I myself have seen some impressive improvements at organizations that I've work with), it is best for teams actually involved with the details of project progress to measure key improvement factors and not simply rely on gut feel (and not rely on outside consultants hired to achieve the benefits!).

Both IBM Rational and the Accelerated Solution Delivery Practice[12] within IBM Global Services have been helping organizations improve their internal IT processes for years, in particular based on iterative and/or agile strategies. Depending on the situation which the client finds itself in, we may recommend either a continuous process improvement strategy, which IBM Rational focuses on; an accelerated improvement strategy, which the ASD practice focuses on; or a combination of both approaches as part of a broader initiative. First, with the continuous strategy you adopt a few techniques at a time, absorb them, learn from your experiences, and then iterate. Table 2 summarizes what we believe to be realistic expectations by taking this approach, based on our actual experiences helping our customers.

Highlights

Table 2. Realistic improvements when adopting agile widely

Success Criteria	Year 1	Year 2	Year 3
Quality	3-5% fewer defects	3-5% fewer defects	3-5% fewer defects
Labor costs	2-3% improvement	4-5% improvement	4-5% improvement
Time to Value	5% faster	10% faster	5% faster
Delivered Fiunctionality	5% improved accuracy	5% improved accuracy	5% improved accuracy

There are several important points that I need to make about the numbers shown in Table 2:

1. They are for large-scale agile adoption across most or all of an IT organization. Not everyone is going to be the highly skilled, highly motivated people you put on your pilot projects.
2. They are very conservative, as I'm a firm believer in under promising and over delivering. We've had several customers who have done much better than this using an aggressive adoption approach which received significant support from all aspects of the organization. So, as the agile community likes to say, "your mileage may vary" (YMMV).
3. The results are for year on year. For example, you should hopefully see a 3-to-5% improvement in quality the first year, another 3-to-5% improvement the next year, and so on. The most last process improvement occurs gradually over time, in small increments, a Japanese concept called *kaizen*.

The primary determinant of success is your leadership. Whatever your current situation, you need to choose to make the often difficult changes that enable your organization to improve.

The primary determinant of success is your leadership. Whatever your current situation, you need to choose to make the often difficult changes that enable your organization to improve. Change is uncomfortable, the implication being that not everyone is going to be happy with moving to agile. You will not achieve even these conservative benefits unless you're willing to make them happen. Furthermore, you will need to parameterize and measure the impact of the changes. As the process change moves forward, one of the keys will be the ability to prove "that the gain is worth the pain."

Highlights

The strategy with greatest potential – increasing the flexibility in your approach to IT and the way in which you make IT investments – has the greatest cultural impact on your organization because it often requires a paradigm shift in how the business perceives IT.

Table 3 presents a complementary view to Table 2 by examining four potential improvement strategies (which could be combined). As with Table 2, the figures in Table 3 reflect the experiences of IBM Rational consultants helping organizations to improve their approach to IT [21]. Improving automation, collaboration, and improving your process are all relatively short term endeavors with modest, although not unsubstantial, potential for productivity increase. The strategy with greatest potential – increasing the flexibility in your approach to IT and the way in which you make IT investments – has the greatest cultural impact on your organization because it often requires a paradigm shift in how the business perceives IT.

Table 3. Comparing potential improvement strategies

Improvement Strategy	Cost to implement	Potential Improvement	Timeframe	Cultural Impact
Improve Automation	<5%	5-25%	Weeks	Very Low
Improve Collaboration	5-10%	15-35%	Months	Low
Improve Process	10-35%	25-100%	Quarters	Some
Increase Flexibility and Investment Value	25-50%	2x-10x	Years	High

Using an accelerated approach

The ASD practice has been delivering a mix of rapid/agile/lean development services with clients for over ten years, including high-performance agile delivery centers, turn-key project delivery, and assessments of troubled client implementations. They've frequently used an accelerated approach, where you adopt a larger number of agile practices at once and support this adoption by bringing mentors to help guide agile delivery and transfer skills to the staff. This leads to much higher productivity improvements, but it requires you to partner closely with the ASD mentors at all levels within your organization and commit to a more aggressive accelerated program – in other words, there's quicker gain from greater focus. Their analysis over hundreds of agile projects that they've been involved with is that the more agile techniques you use, the better the aggregate results.

Highlights

Look at What's Changing: A Lesson from Physics

The best indicators in software are measurements of what's changing. For example, an easy way to determine the productivity improvement of an agile team is to calculate its acceleration, the change in its velocity over time [16]. Agile teams already calculate their velocity, the number of points of functionality that they can deliver each iteration, for estimation and planning purposes, and acceleration is simple calculation based on that information. Acceleration doesn't tell you the exact levels of productivity, something that is expensive to calculate, but it is a virtually free estimate of the change in productivity. Better yet, the acceleration across your entire department can be easily calculated as a weighted average and then monetized by multiplying the number of people involved by their fully burdened cost.

IBM Rational and the ASD practice have been working together to optimize our collective assets, and jointly engage where the client desires a mix of practice improvement, tooling and want to take a more aggressive approach to optimization.

Table 2 addresses the factors which you may want to consider when calculating productivity. If your organization supports a domain where delivery time is paramount, then your calculation of productivity improvement would be highly weighted towards the time-to-value statistics and your process improvement efforts would be similarly skewed towards techniques for reducing overall delivery time. If all of these factors are equally important, then your productivity improvement in the first year could potentially be 19% [13] – we've seen more than double this on pilot projects, for the reasons discussed earlier, although across an entire IT department the average seems to be 6 to 8% per annum. The critical observation is that the way you calculate productivity improvement is situational.

Part of being a leader is that sometimes you need to take a leap of faith that your vision – in this case, your move to adopt the scaling of agile techniques across your IT department – is a good one. There are no easy fixes, no "silver bullets" to slay the IT productivity werewolf, regardless of what some of the agile marketers may imply. Slow and steady wins the process improvement race.

What challenges should you expect?

As I discussed earlier, the adoption of agile approaches within most organizations, including IBM, typically begins with a grass roots movement. The people involved self select themselves, they're often highly motivated to try new things and learn from their experiences, and more often than not they're often amongst your most highly skilled people. Then, when you "officially" start supporting agile adoption you often choose straightforward pilot projects, put together teams of these motivated and skilled people, and give them the support that they need to succeed. And succeed they do. But soon the situation changes. Suddenly, the projects aren't so straightforward, and you're trying to roll out agile approaches to people who may not be highly skilled or motivated to change.

Our experience is that changing your organizational culture is the primary challenge when adopting agile techniques at scale [17], just like it's the primary challenge with other type of process improvements. The difficulty is your organi-

Highlights

zational culture reflects the people, your organizational goals, the way that people are organized, and the ways that they prefer to work – all of these issues are near and dear to the hearts of the people involved. A common refrain heard from groups that prefer the status quo is "Yes, agile is wonderful, but to allow us to address X they must still continue to produce Y just like other teams." For example, the quality assurance group may still want to be responsible for comprehensive testing at the end of the lifecycle and therefore require a detailed requirement speculation [18], not realizing that agile delivery teams do much of the testing themselves much earlier in the project. Or the data management group may insist that they produce a detailed logical data model and physical data model during the analysis and design phases of the agile project to ensure that corporate standards are followed and existing data sources leveraged appropriately, not realizing that analysis and design are so important to agile teams that they do these activities all the way through the lifecycle – in an evolutionary manner – and would rather have someone knowledgeable about data issues involved throughout the entire project as an agile team member. These requests are not unreasonable; clearly your traditional teams have performed this way for years. But on the agile landscape, these methods can hamper a team's ability to actually achieve the promised benefits. Instead of giving in to these requests, in other words taking the easy road to mediocrity, you must instead choose the "hard road" and work with those tradionally minded teams to help them also become agile. It will be better for everyone involved in the long run.

Clearly your traditional teams have performed for years. But on the agile landscape, traditional methods can hamper a team's ability to actually achieve the promised benefits.

Parting thoughts

Although many organizations have succeeded with agile approaches to system delivery, that doesn't make agile a silver bullet with which you can easily slay the IT productivity lycanthrope. I have described how to scale agile approaches on two fronts: for individual project teams and for adopting it across your IT organization. To succeed at scaling agile for project teams you must first recognize the need to apply agile throughout the entire delivery lifecycle, not just construction. Then, depending on the situation that the team finds itself in, you may need to tailor the agile practices which you have adopted for applicable scaling factors – team size, geographical distribution, regulatory compliance, domain complexity, organizational distribution, technical complexity, organizational complexity, or enterprise discipline. Disciplined agile teams focus on producing repeatable results, not on the bureaucratic façade of following repeatable practices. To succeed at scaling agile strategies across your IT organization you must address the following five areas: People, principles, practices, process, and products (technology and tooling).

Highlights

Nobody gets a gold star for being agile; the goal is to get better, not to become agile. Considering that the focus of this paper, I realize this sounds contradictory. But if your team can succeed with agile techniques, you will certainly become more effective at software and systems delivery.

Acknowledgements

I'd like to thank Alan W. Brown, Paul Gorans, David Lubanko, Mike Perrow, Walker Royce, Rick Weaver, and Elizabeth Woodward for their feedback, which was incorporated into this white paper.

About the Author

Scott W. Ambler is Chief Methodologist/Agile with IBM Rational and he works with IBM customers around the world to improve their software processes. He is the founder of the Agile Modeling (AM), Agile Data (AD), Agile Unified Process (AUP), and Enterprise Unified Process (EUP) methodologies. Scott is the (co-)author of 19 books, including Refactoring Databases, Agile Modeling, Agile Database Techniques, The Object Primer 3rd Edition, and The Enterprise Unified Process. Scott is a senior contributing editor with Information Week. His personal home page is www.ibm.com/software/rational/leadership/leaders/#scott and his Agile at Scale blog is www.ibm.com/developerworks/blogs/page/ambler.

Endnotes

[1] Throughout this paper the term process shall also include the terms "method" and "methodology." These terms are used interchangeably within the IT industry and for the sake of simplicity I have chosen to use the term "process."

[2] This difference is discussed in, for example, Stober, T. and Hansmann, W. (2010). *Agile Software Development: Best Practices for Large Software Development Projects.* New York: Springer Publishing, and in Larman, C. and Vodde, B. (2009). *Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum.* Upper Saddle River, NJ: Addison Wesley.

[3] For a more detailed discussion of the Agile Manifesto, see "Examining the Agile Manifesto" at www.ambysoft.com/essays/agileManifesto.html

[4] Regression testing, essentially, tests whether changes to existing software have introduced new problems.

[5] A common strategy to do so via retrospectives, see www.retrospectives.com

[6] The ASM is described in detail in the white paper "The Agile Scaling Model (ASM): Adapting Agile Methods for Complex Environments" which can be downloaded at ftp://ftp.software.ibm.com/common/ssi/sa/wh/n/raw14204usen/RAW14204USEN.PDF

[7] My follow-up whitepaper to this one, entitled "Agility@Scale: Disciplined Strategies for Scaling Agile Delivery", will go into the details of scaling across your organization and tailoring agile practices to reflect the realities of various scaling factors. It will be available in the first quarter of 2010 at www.ibm.com

[8] This shouldn't be confused with the 5Ps of marketing: product, price, place, promotion, and people.

[9] This is the number of "points" of functionality which a team delivers each iteration.

[10] I've created value stream maps with several customers around the world where we analyzed their existing processes which some of their more traditional staff believed worked well only to discover they had efficiency ratings of 20-30%. You can't fix problems which you are blind to.

[11] My surveys are performed in a completely open manner. The original questions as they were asked, the source data (without identifying information), and summary slide decks are available free of charge from www.ambysoft.com/surveys/ so that you can analyze the results for yourself.

[12] See http://www.ibm.com/services/us/index.wss/offering/gbs/a1029597

[13] Calculated as 1.05*1.03*1.05*1.05. This assumes that you focus on all four success criteria and that they are all weighted equally as important in your organization.

References

1. Dr. Dobb's Journal's July 2009 State of the IT Union Survey - www.ambysoft.com/surveys/stateOfITUnion200907.html
2. Dr. Dobb's Journal's 2008 Project Success Survey - www.ambysoft.com/surveys/success2008.html
3. Ambler, S.W. (2009). *The Agile Scaling Model (ASM): Adapting Agile Methods for Complex Environments* - ftp://ftp.software.ibm.com/common/ssi/sa/wh/n/raw14204usen/RAW14204USEN.PDF
4. Ambler, S.W. (2005). *The Agile System Development Life Cycle (SDLC)* - www.ambysoft.com/essays/agileLifecycle.html
5. Ambler, S.W. (2004). *The Object Primer 3rd Edition: Agile Model Driven Development with UML 2.0*. New York: Cambridge University Press.
6. Kruchten, P. (2009). *The context of software development*. pkruchten.wordpress.com/2009/07/22/the-context-of-software-development/
7. Dr. Dobb's Journal November 2009 State of the IT Union Survey. www.ambysoft.com/surveys/stateOfITUnion200911.html
8. Ambler, S.W. (2004). *Agile Database Techniques: Effective Strategies for the Agile Development*. New York: Wiley Publishing.
9. Ambler, S.W., Nalbone, J., and Vizdos, M. (2004). *The Enterprise Unified Process: Enhancing the Rational Unified Process*. Boston: Addison Wesley.
10. Poppendieck, M. and Poppendieck, T. (2006). *Implementing Lean Software Development: From Concept to Cash*. Boston: Addison Wesley.
11. Ambler, S.W. (2003). Examining the "Big Requirements Up Front (BRUF) Approach". www.agilemodeling.com/essays/examiningBRUF.htm
12. Ambler, S.W. & Kroll, P. (2007). *Lean Development Governance*. www.software.ibm.com/webapp/iwm/web/preLogin.do?lang=en_US&source=swg-ldg
13. Agile Manifesto, www.agilemanifesto.org
14. IBM Practices home page. www.ibm.com/developerworks/rational/practices/index.html
15. Kroll, P. and Krebs, W. (2008). *Introducing IBM Rational Self Check for Software Teams*. www.ibm.com/developerworks/rational/library/edge/08/may08/kroll_krebs/index.html
16. Ambler, S.W. (2009). *Examining Acceleration*. https://www.ibm.com/developerworks/mydeveloperworks/blogs/ambler/entry/metric_acceleration_examined
17. Ambler, S.W. (2009). *Not Agile Yet? Exploring the Excuses*. www.ddj.com/architect/222002704
18. Ambler, S.W. (2009). *The Danger of Detailed Speculations*. https://www.ibm.com/developerworks/mydeveloperworks/blogs/ambler/entry/detailed_speculations
19. Measured Capability Improvement Framework Home Page. www.ibm.com/software/rational/mcif/
20. Glazer, H., Dalton, J., Anderson, D.J., Konrad, M.D., and Shrum, S. (2008). *CMMI or Agile: Why Not Embrace Both!* www.sei.cmu.edu/reports/08tn003.pdf
21. Royce, W. (2009). "Improving Software Economics: Top 10 Principles of Achieving Agility at Scale." download.boulder.ibm.com/ibmdl/pub/software/rational/web/whitepapers/Royce_SoftwareEconomics_whitepaper3.pdf

IBM Corporation

Software Group

Route 100

Somers, NY 10589

U.S.A.

Produced in the United States of America

March 2010

RAW14211-USEN-00