IBM agility@scale™: Become as Agile as You Can Be

By Scott W. Ambler

Chief Methodologist for Agile and Lean, IBM Rational

Next page

IBM agility@scale™: Become as Agile as You Can Be

Contents

Next page

IBM agility@scale™

Contents

Foreword

Scott Ambler's work on Agile process maturity provides the structure that is needed by many companies seeking to make the transition to Agility while still meeting their overall organizational needs and requirements. In some cases, this means establishing IT governance and compliance while others simply need just enough process to get the job done. Scott's work teaches us that the Agile Manifesto emphasis of Individuals and Interactions over processes and tools, does not rule out mature repeatable process frameworks where they make sense. More importantly, as an Agile thought leader, Scott Ambler gives us the necessary structure to understand Agile process maturity and thereby successfully implement Agile processes that can support part of or, in some cases, the entire systems development lifecycle. Scott raises the bar by helping us implement disciplined agile processes which address one or more scaling factors such as team size or distributed teams that are critical for scaling Agility.

Agile works and the results of many Agile initiatives have been stunning in their success in terms of both quality and productivity. With Scott Ambler's help we can enhance our Agile processes to the point where we can literally be as Agile as we can be while meeting larger organizational needs for process repeatability and Lean Agile frameworks. I encourage you to read this work carefully as it takes us on a journey to successfully implementing Agile processes in ways that yield dramatic results and increased success in systems development.

Bob Aiello
Editor in Chief
CM Crossroads
Author of "Configuration Management Best Practices: Practical Methods that Work in the Real World "

IBM agility@scale™

Contents

Agile software development is an evolutionary, highly collaborative, quality-focused approach to software development where potentially shippable working software is produced on a regular basis. Agile software development processes include Scrum, Extreme Programming (XP), Disciplined Agile Delivery (DAD) and Agile Modeling (AM), to name a few. Although agile approaches are often equated to the development of Web-based applications, in reality they're also being applied to mobile applications, fat-client applications, business intelligence (BI) systems and even mainframe applications. They're being applied by a range of organizations, including but not limited to e-commerce companies, financial companies, manufacturers, retailers and government agencies

Agile software development techniques have taken the industry by storm, with 76 percent of organizations reporting that they had one or more agile projects under way [1]. Agile is becoming widespread because it works — organizations are finding that agile 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, 3].

IBM agility@scale™

Contents

Learn More

- Be As Agile As You Can Be (webcast)

This eBook begins with an overview of the Agile Scaling Model (ASM), a framework we use to provide context to the plethora of agile methodologies available today. It then describes each of the three categories in detail — core agile, disciplined agile delivery, and agility at scale — showing how they build on one another. Then it works through an example of the development of an online bartering system and finishes with some advice for successful agile adoption.

The Agile Scaling Model (ASM)

The Agile Scaling Model (ASM) [4] 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:

1. Core agile development. Core agile methods, such as Scrum and Agile Modeling, are self governing, have a value-driven system development lifecycle (SDLC), and address a portion of the development lifecycle. These methods, and their practices, such as daily stand up meetings and requirements envisioning, are optimized for small, co-located teams developing fairly straightforward systems.

2. Disciplined agile delivery. Disciplined agile delivery processes – which include The Open Unified Process (OpenUP) and the Disciplined Agile Delivery (DAD) method itself – go further by covering the full software development lifecycle from project inception to transitioning the system into your production environment (or into the marketplace as the case may be). Disciplined agile delivery processes are self

IBM agility@scale™

Contents

organizing within an appropriate governance framework and take both a risk and value driven approach to the lifecycle. Like the core agile development category, this category is also focused on small, co-located teams delivering fairly straightforward solutions. To address the full delivery lifecycle you need to combine practices from several core methods, or adopt a method which has already done so.

3. Agility at Scale. This category focuses on disciplined agile delivery where one or more scaling factors are applicable. The eight scaling factors are team size, geographical distribution, regulatory compliance, organizational complexity, technical complexity, organizational distribution, domain complexity, and enterprise discipline. All of these scaling factors are ranges, and not all of them will likely be applicable to any given project, so you need to be flexible when scaling agile approaches to meet the needs of your unique 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.

IBM agility@scale™

Contents

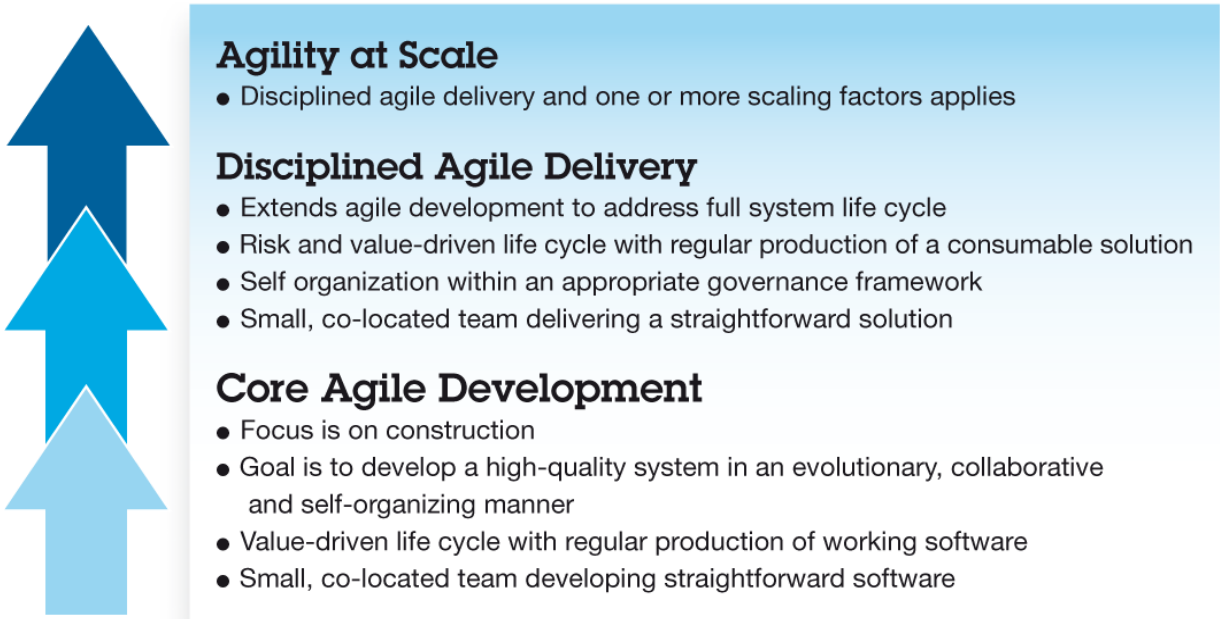

Figure 1: Agile Scaling Model (ASM)

IBM agility@scale™

Contents

Learn More

- Scaling Scrum: Lessons from the Trenches (webcast)

Core Agile Development

Core agile methods address a portion of the software development life cycle. They conform to the values and principles of the Agile Manifesto [5, 6] written in 2001 by a group of 17 software development experts, and are typically described as a cohesive collection of practices. Examples of core agile processes include:

Scrum. The focus of Scrum is project leadership and requirements management. Scrum defines a high-level life cycle for construction iterations (what Scrum calls "sprints"), see Figure 2, and several practices such as a daily stand-up "Scrum" meeting, product owner, product backlog, iteration/sprint planning and potentially shippable software.

Extreme Programming (XP). XP is a collection of practices for software construction, include refactoring, test-first design, pair programming, on-site-customer, continuous integration, whole team and collective ownership. Note that XP is close to being a full-fledged, disciplined agile delivery method but is missing explicit project initiation and release practices.

Agile Modeling (AM). AM is a collection of practices for light-weight modeling and documentation, including requirements envisioning, executable specifications, active stakeholder participation, prioritized requirements, and prove it with code.

Agile Data (AD). AD is a collection of practices for database development, including agile data modeling, database testing, database refactoring, and continuous database integration.

IBM agility@scale™

Contents

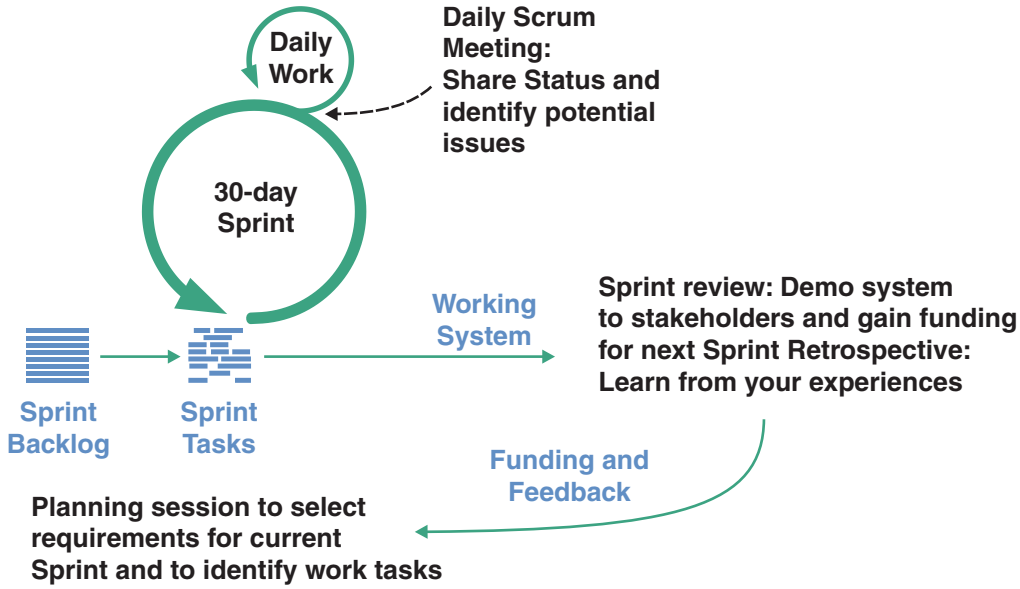

Figure 2. Scrum construction life cycle

IBM agility@scale™

Contents

Learn More

- Take the Agile Fitness Survey

Disciplined Agile Delivery

Disciplined agile delivery processes extend core agile development methods to address the full system delivery life cycle (SDLC). As the criteria suggest (see sidebar: Are You Really Agile?), they also tend to "dial up" certain aspects of agile development, such as testing, measurement and process improvement. 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 life cycle. It is performed in a highly collaborative and self-organizing manner, 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 they face.

Examples of Level 2 agile processes include:

Open Unified Process (OpenUP). OpenUP [5], the definition of which is available via open source, combines and extends practices from Scrum, XP, AM and Rational Unified Process (RUP) for co-located agile teams that are building business applications. OpenUP practices include whole team, daily stand-up meeting, prioritized work items, risk-driven life cycle, TDD, active stakeholder participation and continuous integration.

Dynamic System Development Method (DSDM). DSDM is an agile delivery process originally based on Rapid Application Development (RAD), which is often used to

IBM agility@scale™

Contents

develop user-interface intensive applications. DSDM practices include prototyping, test throughout the life cycle, reversible changes and feasibility study.

Disciplined Agile Delivery (DAD). DAD [6] is a hybrid method which combines strategies and practices from several software methods, including Scrum, XP, AM, Agile Data, and the Open Unified Process (OpenUP). My apologies for any confusion around using the same name for both the ASM category and the method..

Figure 3 depicts the life cycle of the Disciplined Agile Delivery (DAD) method. This life cycle expands upon the Scrum construction life cycle in several important ways. First, it includes an explicit project inception phase where you do some initial modeling, start putting together your team, and gain initial project funding. Second, it extends the product backlog concept to include not only functional requirements but also defects and other work items such as providing feedback on work from other teams, taking training courses, and so on. Third, it includes explicit transition/release and production phases.

IBM agility@scale™

Contents

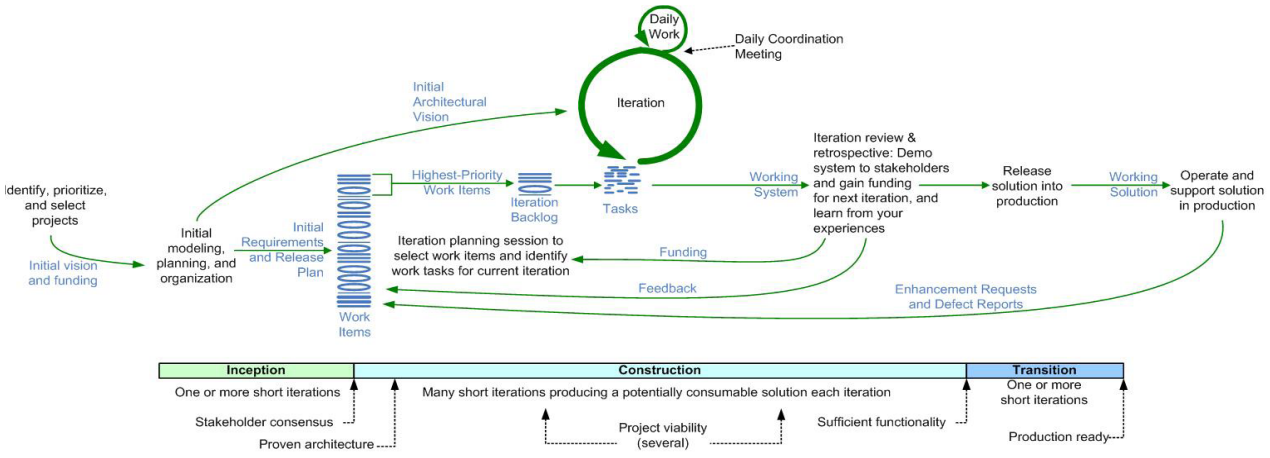

Figure 3. Disciplined Agile Delivery (DAD) system-development life cycle

IBM agility@scale™

Contents

Are You Really Agile?

A common problem in many organizations is that undisciplined "ad hoc" teams will claim to be agile, often simply because they're not writing any documentation and have read an article or two about how cool agile is. Unfortunately, those ad hoc teams often run into trouble and give actual agile teams a bad name. I suggest the following criteria to determine whether a team is agile:

1. Value. Agile teams provide value to their stakeholders on a regular basis.

2. Validation. Agile teams do, at a minimum, continuous developer regression testing. Disciplined agile teams take a Test-Driven Development (TDD) approach.

3. Active stakeholder participation. Agile teams work closely with their stakeholders, ideally on a daily basis.

4. Self organization. Agile teams are self-organizing, and disciplined agile teams work within an appropriate governance framework.

5. Improvement. Agile teams regularly reflect on, and disciplined teams also measure, how they work together, and then act to improve on their findings in a timely manner.

Several of the terms in the above criteria are underlined to indicate where your strategy needs to be flexible. For example, some agile teams will produce working software every two weeks, whereas others may be in a more complex situation and may do so only every two months. Different situations require different strategies, implying that one process size does not fit all. Sadly, the 2010 How Agile Are You? Survey [7] found that only 53% of "agile" teams met all five of these criteria, with self organization as the most challenging criterion to fulfill.

IBM agility@scale™

Contents

Learn More

- Adapt Agile Methods for Complex Environments (whitepaper)

Agility at Scale

In the early days of agile, the applications where agile development was applied were smaller in scope and relatively straightforward. Today, organizations apply agile strategies to a broader set of projects. This is what IBM agility@scale™ is all about —explicitly addressing the complexities that disciplined agile delivery teams face in the real world. Figure 4 gives an overview of the eight scaling factors of agile development.

Each factor has a range of complexities, and each team will have a different combination and therefore will need a process, team structure and tooling environment tailored to meet its unique situation. Case agile processes on the ASM work best when basically all factors are at the left-hand side (the low-complexity side), although they can potentially be tailored to address greater complexity with strategies from higher-level processes. Disciplined agile delivery processes typically assume that one or more of the factors are slightly to the right, although you are considered to be 'at scale' when one or more factors are significantly to the right (the high-complexity side).

When it comes to tooling, many agile teams will find that they can make do with open source tools when most scaling factors are to the left. But, when they find themselves in agility at scale situations they soon discover that they need to adopt more sophisticated tools. To succeed at scaling agile, you will need tools that integrate easily, are sufficiently instrumented to provide the metrics required for effective governance, support distributed development, enhance collaboration between disparate team

IBM agility@scale™

Contents

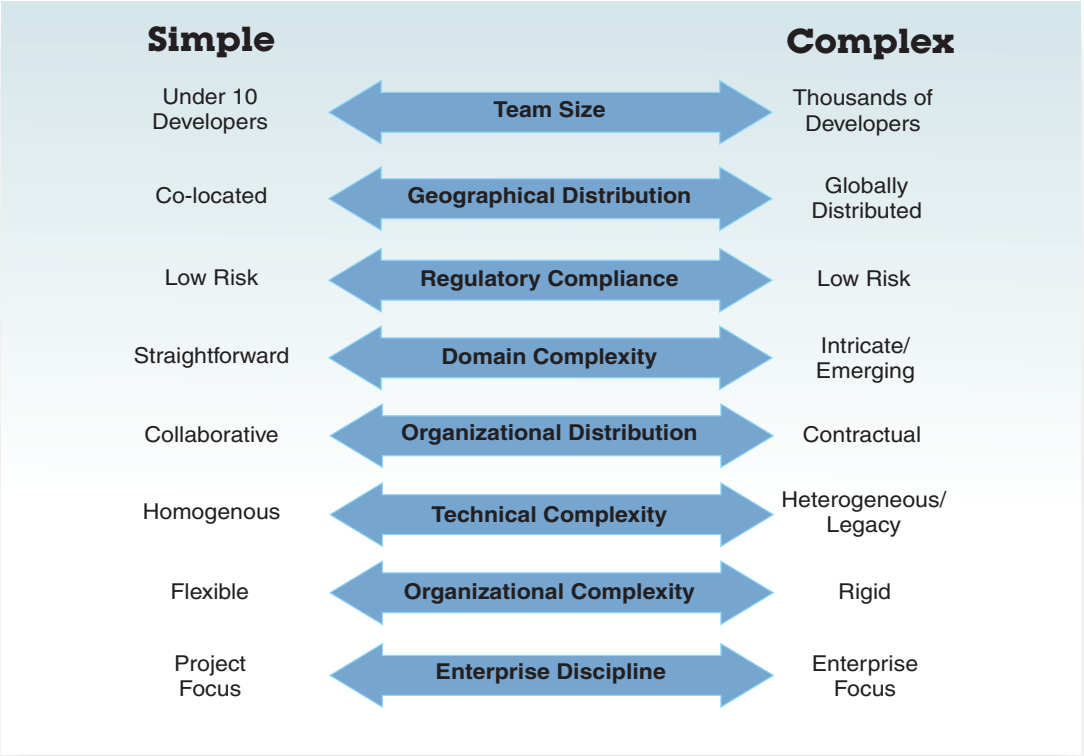

Figure 4. Potential scaling factors for software development

IBM agility@scale™

Contents

members, and automate as much of the work as possible to comply with regulations. Jazz-compliant tools, see www.jazz.net, are very good options for agile teams working at scale.

Agile Online Bartering

The best way to understand agile software development at scale is by example. SWA International, a fictitious company, wanted to extend its existing e-commerce offerings by adding bartering functionality. With the current economic times as they are, the company found that fewer people were online purchasing products from them. Offline, SWA had noticed that some of its customers were trying to barter, offering their services to help pay down their existing debt to SWA. The company felt the time was ripe for online bartering.

SWA put together a team that would eventually grow to 25 people, composed of two subteams, one in Toronto and the other in San Francisco, plus several people working from their home offices in other cities. The company chose to follow the Disciplined Agile Delivery (DAD) method with two-week iterations. The first iteration was spent doing initial-requirements envisioning, with four senior business staff, four senior developers on the project, two operations staff members and the chief technology officer (CTO).

The goals were not only to understand the scope of the new system but also to come to stakeholder concurrence regarding what that scope was, thereby reducing business risk. In parallel, the CTO and senior developers did high-level architecture envisioning to identify the subsystems and their interfaces, critical information required to split the work appropriately between the

IBM agility@scale™

Contents

two subteams — distributed agile teams prefer to organize themselves around the architecture instead of around a job function (e.g., having the developers in one location, the testers in another, the modelers in another and so on). One member of the business staff was an expert in regulatory issues, an important issue because SWA was entering a new line of business and publicly traded in the U.S. (hence, Sarbanes-Oxley compliance was an issue). Two of the businesspeople became product owners, one for each subteam, for the rest of the project. The product owner is the person who prioritizes the requirements and provides detailed information about the business to developers.

The second iteration focused on developing an end-to-end working skeleton of the system to prove that the architecture actually works. To simplify this effort, the work was performed in Toronto, the headquarters for SWA International, with three members of the San Francisco team involved. The team built a simple Web page that allowed an end user to select an item from a list and then make a bartering offer for it. The offer was then persisted by the system, and an e-mail notice was sent to the owner of the item for which the offer had been made. The team ran load tests showing that the system worked with 500 concurrent users for a period of 10 hours — a critical performance requirement for the system. At the end of the iteration, they demonstrated the system to the original stakeholders who had been involved in setting the requirements envisioning for the previous iteration, proving that the architecture for the system worked. At the end of the first month, they had reduced both their business and technical risks substantially.

IBM agility@scale™

Contents

For the next four construction iterations, the work was split between the Toronto and San Francisco teams, with the Toronto team focusing on the financial processing (SWA takes a fee for each swap), tax calculation and remittance, and management reporting, while the San Francisco team focused on the trading functionality. Both teams needed to coordinate their work on a daily basis, with the architecture owner on each team serving as the primary contact point for technical issues (often putting together the right technical people on each team to resolve an issue), and the product owners coordinating the requirements. A person living in Maui was the "independent test team," integrating the entire system every day and performing exploratory testing on it to look for bugs the developers had missed[8]. Even though the development team was doing comprehensive regression testing of its own, many problems still got past them, so the defects the independent tester found were reported back to the development teams via their Jazz-based tools. The first version of the system was delivered into production after three months, and subsequent releases every six weeks (three iterations) after that.

Become as Agile as You Can Be

Many organizations have been successful at adopting agile software development approaches in part because the greatest focus until now has been on pilot projects or on a handful of projects within an organization. However, successful process improvement across an entire organization can prove difficult to implement in practice, often simply because by casting a wider net you run into a wider range of challenges. I've found that the following strategies can help increase your chances of success at improving your software process.

IBM agility@scale™

Contents

Learn More

- Organize Global Agile Teams (Agile 101 class)

1. Recognize that the true goal is to improve. The reality is that nobody is going to give you a little gold star for being agile. They might, however, reward you for becoming more effective at system delivery. Agile techniques can often help with this, but we need to remember that there are still some pretty good ideas out there in the traditional community, too.

2. Have a plan. For your process-improvement efforts to be successful, you should first determine what your goals are, what your current situation is and what challenges you face.

3. Gain some experience. Adopt agile approaches on one or more medium-risk pilot project(s) to gain organizational experience as well as build expertise among your staff. It's important to expect to run into a few problems because pilot projects never go perfectly.

4. Explicitly manage your process-improvement efforts. A common agile strategy is for a team to reflect regularly on its approach so as to identify potential improvements, and then to act on those improvements. Teams that explicitly track their progress at adopting improvements are more successful than those that don't [9].

5. Invest in your staff. You need to train, educate and mentor your staff in agile philosophies, processes, practices and tooling. Focus on the people involved with the pilots at first and train them on a just-in-time (JIT) basis. Don't forget senior management, project management and anyone interfacing with the pilot team — because these individuals need to change the way that they work, too.

IBM agility@scale™

Contents

Parting Thoughts

Many organizations have succeeded at applying agile at scale, and you can, too. If you keep your wits about you and stay away from some of the rhetoric surrounding core agile methods you should be okay. Minimally, you want a disciplined agile system delivery approach that addresses the full life cycle, not just parts of it. Remember that you will often find yourself in a scaling situation and that different teams will experience differing scaling factors -- that's okay because it is fairly straightforward to scale agile strategies with effective practices and tooling. With a realistic approach to process improvement and with a bit of help from the outside, you can increase your return on investment (ROI), quality, stakeholder satisfaction and time to value through agility at scale.

References and Suggested Resources

1. Dr. Dobb's Journal's July 2009 State of the IT Union Survey.

2. Dr. Dobb's Journal's 2008 Project Success Survey.

3. Dr. Dobb's Journal's 2010 Project Success Survey.

4. Ambler, S.W. (2009). The Agile Scaling Model (ASM): Adapting Agile Methods for Complex Environments.

5. Kroll, P. and MacIsaac, B. (2006). Agility and Discipline Made Easy: Practices from OpenUP and RUP. Boston: Addison-Wesley.

6. Ambler, S.W. (2009). Disciplined Agile Delivery (DAD).

7. Ambler, S.W. (2010). 2010 How Agile Are You? Survey Results.

8. Ambler, S.W. (2003). Agile Testing and Quality Strategies: Discipline Over Rhetoric.

9. Kroll, P. and Krebs, W. (2008). Introducing IBM Rational Self Check for Software Teams.

10. IBM Agile Development — Rational Home Page.

IBM agility@scale™

Contents

Learn More

- IBM Agile Development

Scott W. Ambler

Chief Methodologist for Agile and Lean,
IBM Rational

About the Author

Scott W. Ambler is Chief Methodologist for Agile and Lean 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), Disciplined Agile Delivery (DAD), and Enterprise Unified Process (EUP) methodologies.

Ambler 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. He is a senior contributing editor with Dr. Dobb's Journal. His personal home page is **ibm.com**/software/rational/leadership/leaders/#scott , and his Agility@Scale blog is **ibm.com**/developerworks/blogs/page/ambler

IBM Global Services
Route 100
Somers, NY 10589
U.S.A.

Produced in the United States of America
November 2010

Previous page