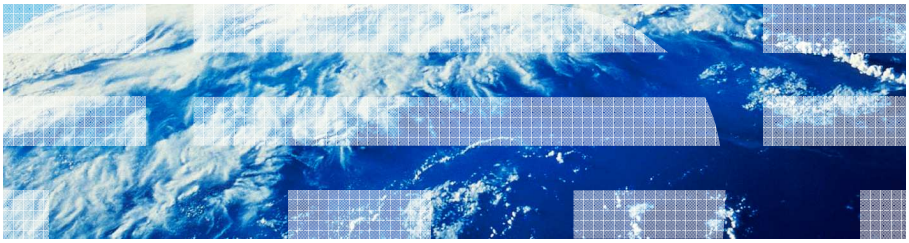
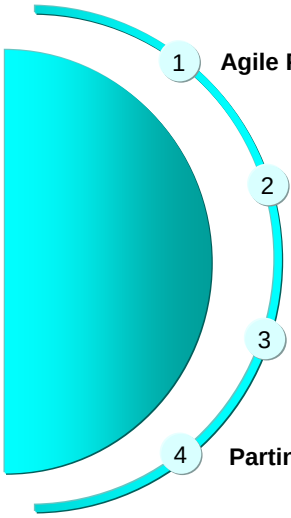
Disciplined Agile Delivery (DAD): The Foundation for Scaling Agile

Scott W. Ambler
Chief Methodologist for Agile and Lean, IBM Rational
www.ibm.com/developerworks/blogs/page/ambler
twitter.com/scottwambler



We hope that you find these slides useful. If you would like to discuss this topic further, please contact IBM.

Agenda



1. Agile Fundamentals
2. Disciplined Agile Delivery (DAD)
3. Agility at Scale
4. Parting Thoughts

What is agile?

- Agile is a highly collaborative, evolutionary, quality focused approach to software development.

- How agile is different:
  - –Focus on collaboration
  - –Focus on quality
  - –Focus on working solutions
  - –Agilists are generalizing specialists
  - –Agile is based on practice, not theory

Focus on collaboration:

Less paperwork and more conversation

Stakeholders actively involved

Focus on quality:

Have a full regression test suite for your systems

Develop loosely-coupled, highly cohesive architectures

Refactor to keep them this way

Focus on working solutions:

Greater feedback makes agile projects easier to manage

Less documentation is required

Less bureaucracy

Agilists are generalizing specialists:

Fewer hand offs between people

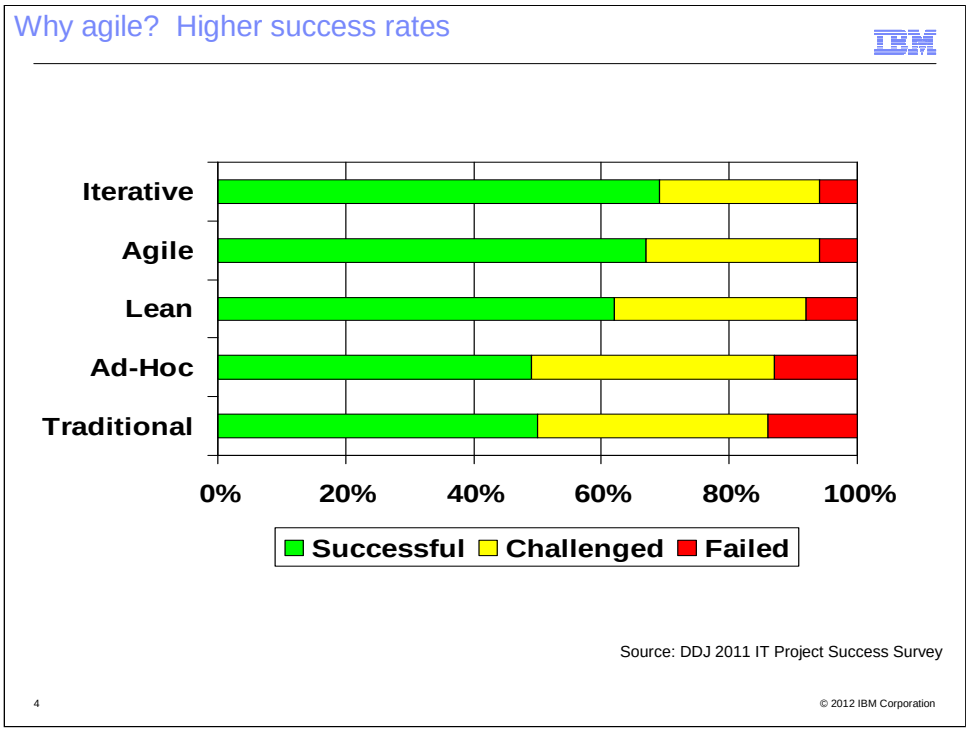
Fewer people required

Specialists find it difficult at first to fit into the team

Agile is based on practice, not theory:

This is a significant change from traditional

You need to see how agile works in practice to truly understand it



Detailed results online at www.ambysoft.com/surveys/

Success as defined by the respondent (this is the same for all slides in this deck). See other slides for how IT professionals define success in practice

Calculated by summarizing the weighted average of each range (i.e. 90-100% averages to 95%) times the number of respondents. Same approach taken for other slides too.

Definitions used in the survey:

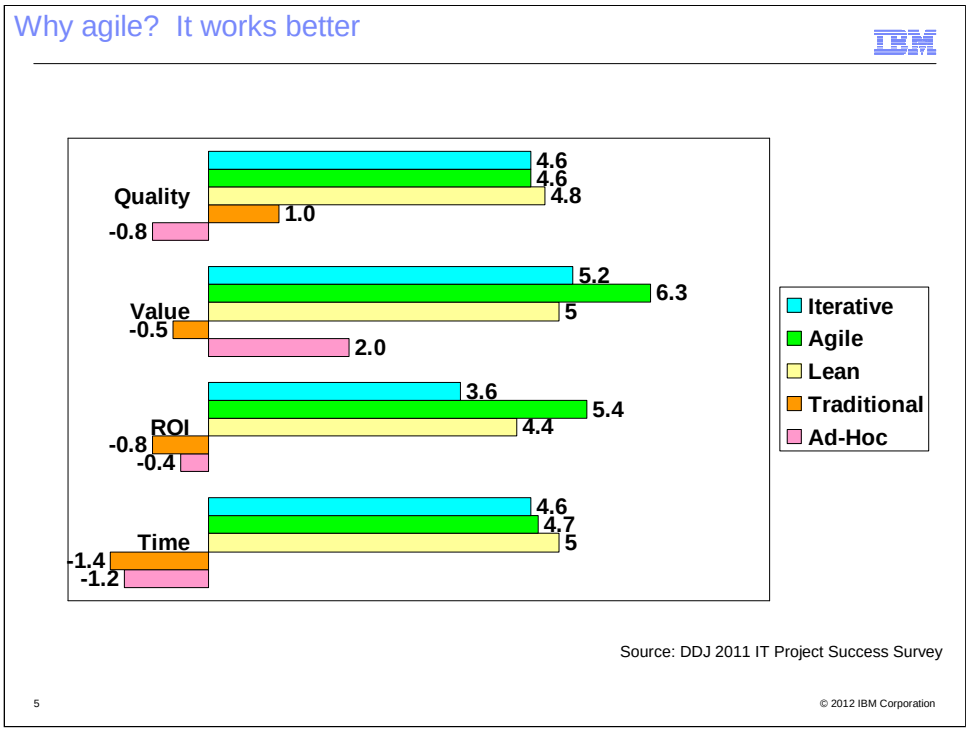
On an ad-hoc software development project the team does not follow a defined process.

On an iterative software development project the team follows a process which is organized into periods that are often referred to as iterations or time boxes. On any given day of the project team members may be gathering requirements, doing design, writing code, testing, and so on. An example of an iterative process is RUP.

On an agile software development project the team follows an iterative process which is also lightweight, highly collaborative, self-organizing, and quality focused. An example of an agile process is OpenUP, Scrum, and XP.

On a traditional software development project the team follows a staged process where the requirements are first identified, then the architecture/design is defined, then the coding occurs, then testing, then deployment. Traditional processes are often referred to as "waterfall" or simply "serial" processes.

Lean is a label applied to a customer value-focused mindset/philosophy. A lean process continuously strives to optimize value to the end customer, while minimizing waste which may be measured in terms of time, quality, and cost. Ultimately the Lean journey is the development of a learning organization. Examples of Lean methods/processes include Kanban and Scrumban.



Detailed results online at www.ambysoft.com/surveys/

Bottom Line: Agile teams produce higher quality work, are quicker to deliver, are more likely to deliver the right functionality, and more likely to provide greater ROI than traditional teams

Tower of Babel: Agile terminology

Disciplined Agile	XP	Scrum	Other
Team lead	Coach	Scrum master	Project manager
Iteration	Iteration	Sprint	Timebox
Daily stand up meeting	Daily stand up	Scrum Meeting	Coordination meeting
Retrospective	Retrospective	Sprint retrospective	Reflection meeting
Product owner	Customer	Product owner	Stakeholder representative
Team Member	Extreme programmer	Team member	Developer
Iteration review	-	Sprint review	-

Important: Each agile method has its own terminology. It is highly unlikely that the industry will ever standardize the terminology.

6

Don’t worry too much about the terminology, instead worry about the philosophies and strategies surrounding agile/lean.

What does it mean to be agile?

Disciplined agile teams:

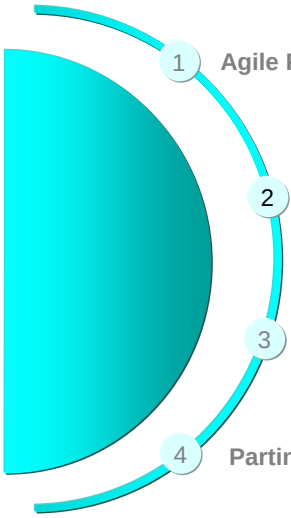
1. Value: Produce a consumable solution on a regular basis which provides value to stakeholders.
2. Validation: Do continuous regression testing, and better yet take a Test-Driven Development (TDD) approach.
3. Stakeholders: Work closely with their stakeholders, or a stakeholder proxy, ideally on a daily basis.
4. Self organization: Are self organizing and work within an appropriate governance framework.
5. Improvement: Regularly reflect on, and measure, how they work together and then act to improve on their findings in a timely manner.

Survey says:

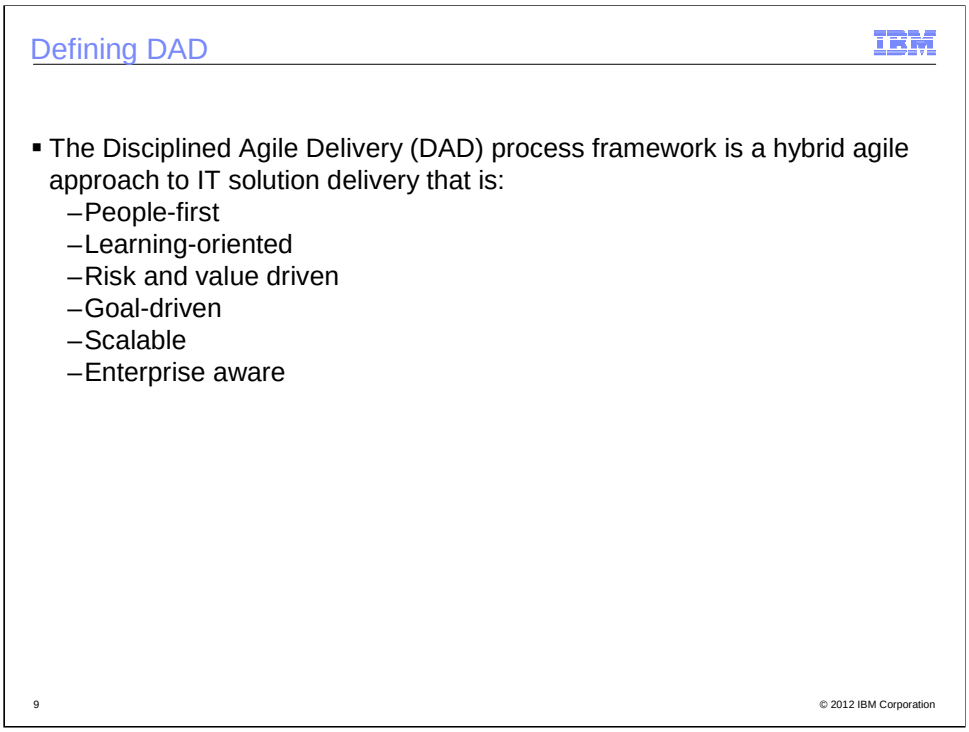
- Value: 94%
- Validation: 87%
- Stakeholders: 95%
- Self organization: 56%
- Improvement: 88%
- All 5: 53%
- Everything but self organization: 72%

Source: 2010 How Agile Are You Survey, www.ambysoft.com/surveys/

Agenda



1. Agile Fundamentals
2. **Disciplined Agile Delivery (DAD)**
3. Agility at Scale
4. Parting Thoughts



From this definition, you can see that the DAD process framework has several important characteristics. These characteristics are:

•People first

•Learning oriented

•Agile

•Hybrid

•IT solution focused

•Goal-driven

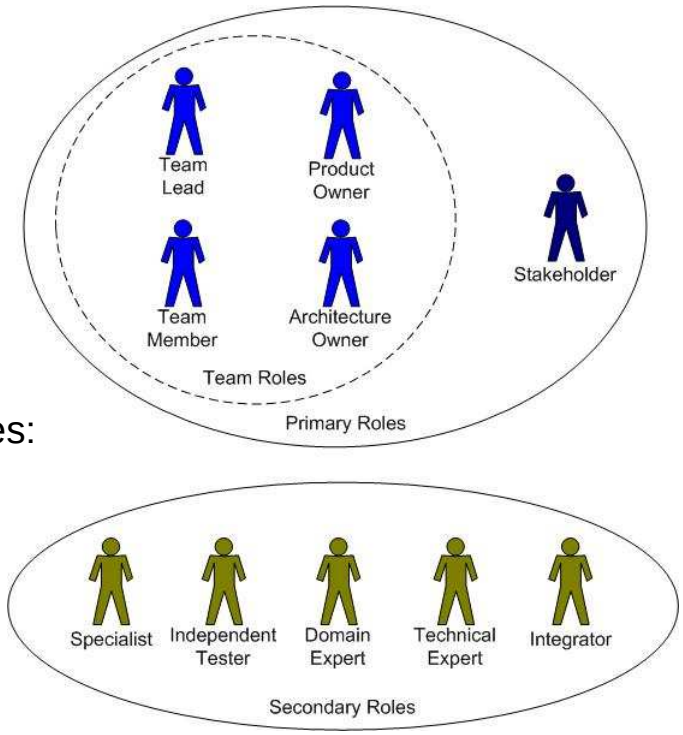
•Delivery focused

•Risk and value driven

•Enterprise aware

People First: Potential roles on disciplined agile teams

- Primary roles:
  - –Stakeholder
  - –Team Lead
  - –Product Owner
  - –Agile Team Member
  - –Architecture Owner
- Secondary/optional roles:
  - –Domain Expert
  - –Technical Expert
  - –Independent Tester
  - –Integrator
  - –Specialist



10

1. Stakeholder. The people who affect the success of your system and are affected by it.
2. Team Lead. Responsible for the success of the project and for employing the process to build a system or product
3. Product Owner. Defines and promotes the vision, goals, and capabilities of the product so team can make decisions. Owns the Product Backlog/Work Item List and defines acceptance criteria for work items; Determines the scope/content of the release (release planning); Defines acceptance criteria for the release and determines when the system is ready for release
4. Agile Team Member. This is everyone else on the team
5. Architecture Owner. Responsible for the architecture of the system or subsystems that the team is working on. Mentors and guides the developers in architectural issues, and leads them through technical issues. Understands the architectural direction and standards of their organization and helps to ensure that the team adheres to them appropriately
6. Domain Expert. Has detailed knowledge about one or more aspects of the problem domain.
7. Technical Expert. Has detailed technical knowledge needed
8. Independent Tester. Focuses on complex testing efforts, working parallel but independent of the team.
9. Integrator. Responsible for building the entire system from its various subsystems.
10. Specialist. Sometimes component subteams require people focused on narrow specialties.

The secondary roles often occur only at scale.

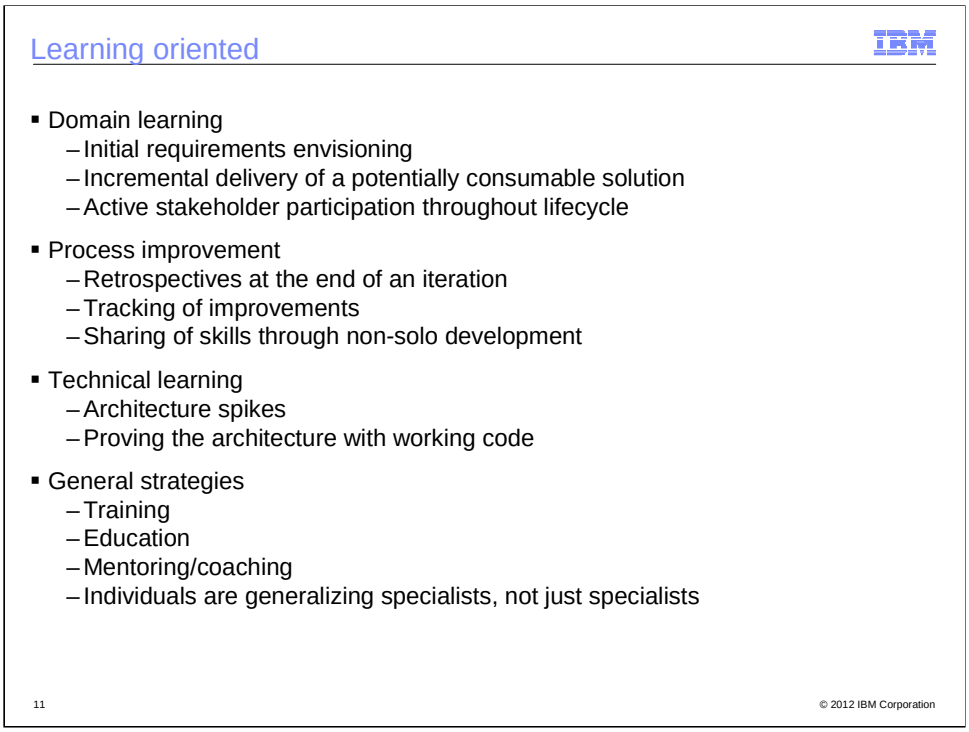
Agile roles are different than traditional roles, even though they may sound similar to what you are used to.

Many organizations struggle to adopt agile effectively because they're not willing to make the changes necessary to support these new roles. This is an example of the "Organizational Complexity" scaling factor.

Agile principle # 5: Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

Note that the Architecture Owner role isn't likely needed on smaller teams (say 10 or less).

Diagram used with permission from the forthcoming book "Disciplined Agile Delivery (DAD): An Introduction" by Scott Ambler and Mark Lines publication date June 2012)

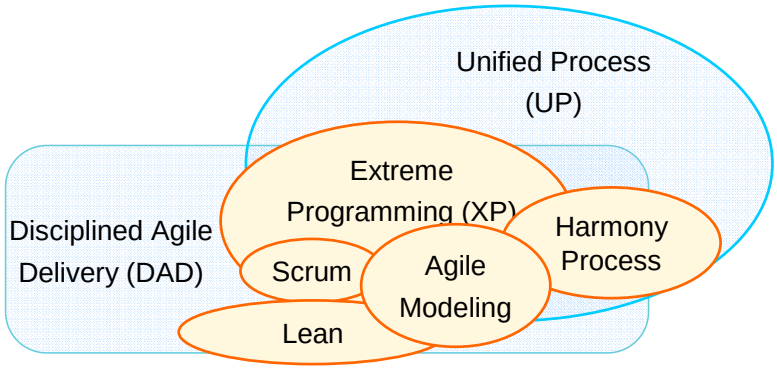


In the years since the Agile Manifesto was written we've discovered that the most effective organizations are the ones that promote a learning environment for their staff. There are three key aspects which a learning environment must address. The first aspect is domain learning – how are you exploring and identifying what your stakeholders need, and perhaps more importantly how are you helping them to do so? The second aspect focuses on learning to improve your process at the individual, team, and enterprise levels. The third aspect is technical learning which focuses on understanding how to effectively work with the tools and technologies being used to craft the solution for your stakeholders.

The DAD process framework suggests several strategies to support domain learning, including initial requirements envisioning, incremental delivery of a potentially consumable solution, and active stakeholder participation through the lifecycle. To support process-focused learning DAD promotes the adoption of retrospectives where the team explicitly identifies potential process improvements, a common agile strategy, as well as continued tracking of those improvements. Within IBM software group we've found that agile teams which held retrospectives improved their productivity more than teams which didn't, and teams which tracked their implementation of the identified improvement strategies were even more successful. Technical learning often comes naturally to IT professionals, many of whom are often eager to work with and explore new tools, techniques, and technologies. This can be a double-edged sword – although they're learning new technical concepts they may not invest sufficient time to master a strategy before moving on to the next one, or may abandon a perfectly fine technology simply because they want to do something new.

There are many general strategies to improve your learning capability. Improved collaboration between people correspondingly increases the opportunities for people to learn from one another. Luckily high collaboration is a hallmark of agility. Investing in training, coaching, and mentoring are obvious learning strategies as well. What may not be so obvious is the move away from promoting specialization amongst your staff and instead fostering a move toward people with more robust skills, something called being a generalizing specialist. Progressive organizations aggressively promote learning opportunities for their people outside their specific areas of specialty as well as opportunities to actually apply these new skills.

Disciplined Agile Delivery (DAD): A Hybrid Framework



DAD is a hybrid process framework. DAD adopt best practices and philosophies from several methodologies

DAD is a process framework which you can tailor to meet the needs of the situation which your team finds itself in.

DAD is also a hybrid methodology which adopts best practices, philosophies, and strategies from a range of proven sources including Scrum, XP, UP (e.g. Rational Unified Process, Open Unified Process, Agile Unified Process), Agile Modeling, and others.

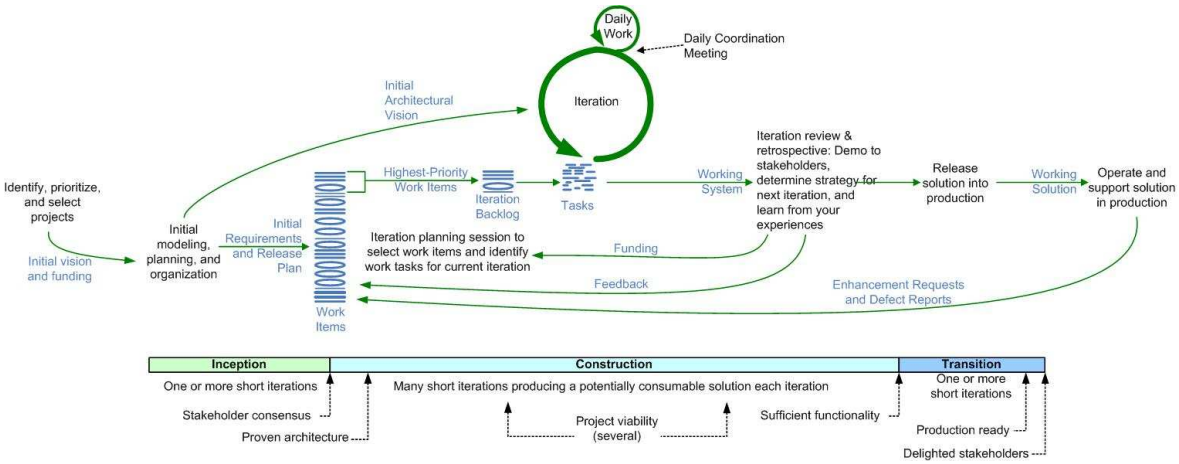
Scrum is effectively a subset of XP.

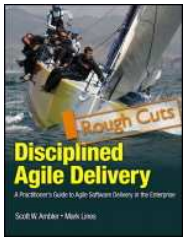
Agile Modeling overlaps a bit with XP (collective ownership, non-solo work, following conventions, …)

Agile methods such as Scrum, XP, and AM include concepts popularized by UP in the 1980s and 1990s, and UP has evolved to address many of the “new” concepts popularized in agile methods.

Diagram used with permission from the forthcoming book “Disciplined Agile Delivery (DAD): An Introduction” by Scott Ambler and Mark Lines publication date June 2012)

The Disciplined Agile Delivery life cycle – Basic





The Disciplined Agile Delivery (DAD) process framework is a people-first, learning-oriented hybrid agile approach to IT solution delivery. It has a risk-value lifecycle, is goal-driven, scalable, and is enterprise aware.

13

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.

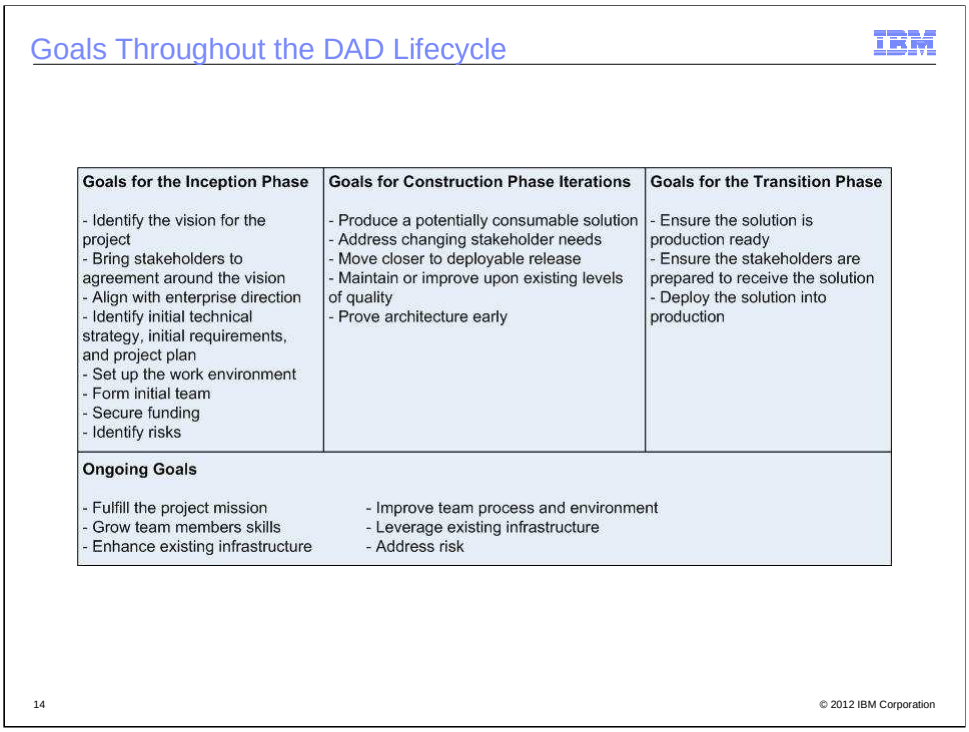
The basic DAD lifecycle expands upon the Scrum construction lifecycle in three important ways:

It has explicit project phases, recognizing that agile delivery is really iterative in the small and serial in the large.

It includes a full range of practices. This 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.

It includes more robust practices. The lifecycle of this figure explicitly reworks the product backlog in the previous slide into the more accurate concept of a ranked work item list. Not only do agile 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.

Diagram used with permission from the forthcoming book “Disciplined Agile Delivery (DAD): An Introduction” by Scott Ambler and Mark Lines publication date June 2012)



DAD addresses the project lifecycle from the point of initiating the project through construction to the point of releasing the solution into production. We explicitly observe that each iteration is NOT the same. Projects do evolve and the work emphasis changes as we move through the lifecycle. To make this clear, we carve the project into phases with light-weight milestones to ensure that we are focused on the right things at the right time. Such areas of focus include initial visioning, architectural modeling, risk management and deployment planning. This differs from mainstream agile methods, which typically focus on the construction aspects of the lifecycle. Details about how to perform initiation and release activities, or even how they fit into the overall lifecycle, are typically vague and left up to you.

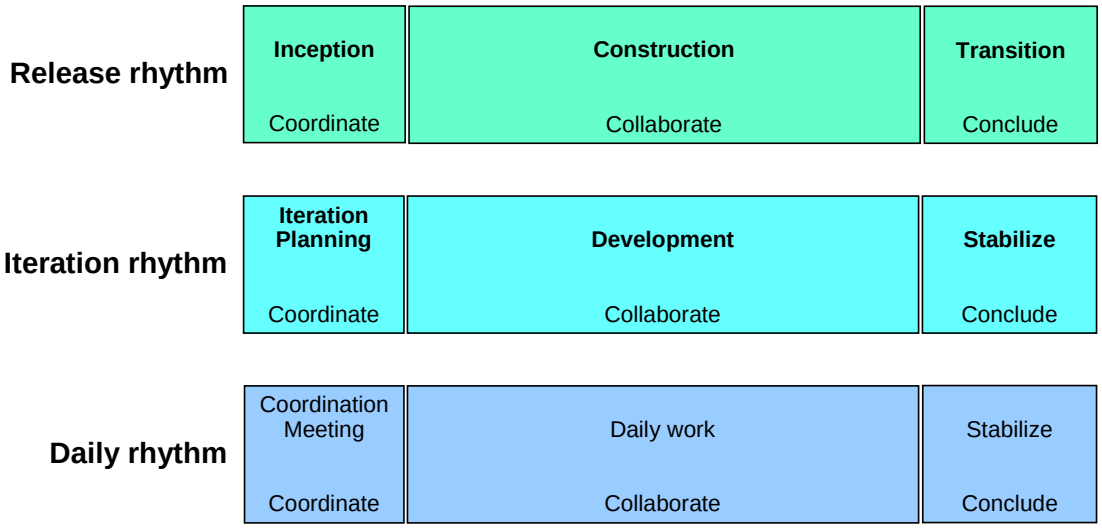
There are of course many ways which these goals can be addressed, so simply indicating the goals is of little value. This goals driven, suggestive approach provides just enough guidance for solution delivery teams while being sufficiently flexible so that teams can tailor the process to address the context of the situation that they find themselves in.

Furthermore, there are several personal goals of individuals, such as specific learning goals, the desire for interesting work, for compensation, and for public recognition of their work. There are also specific stakeholder goals which will be unique to your project.

Diagram used with permission from the forthcoming book “Disciplined Agile

Concept: The Agile 3C rhythm

- The coordinate-collaborate-conclude rhythm occurs at several scales on a disciplined agile delivery (DAD) project:



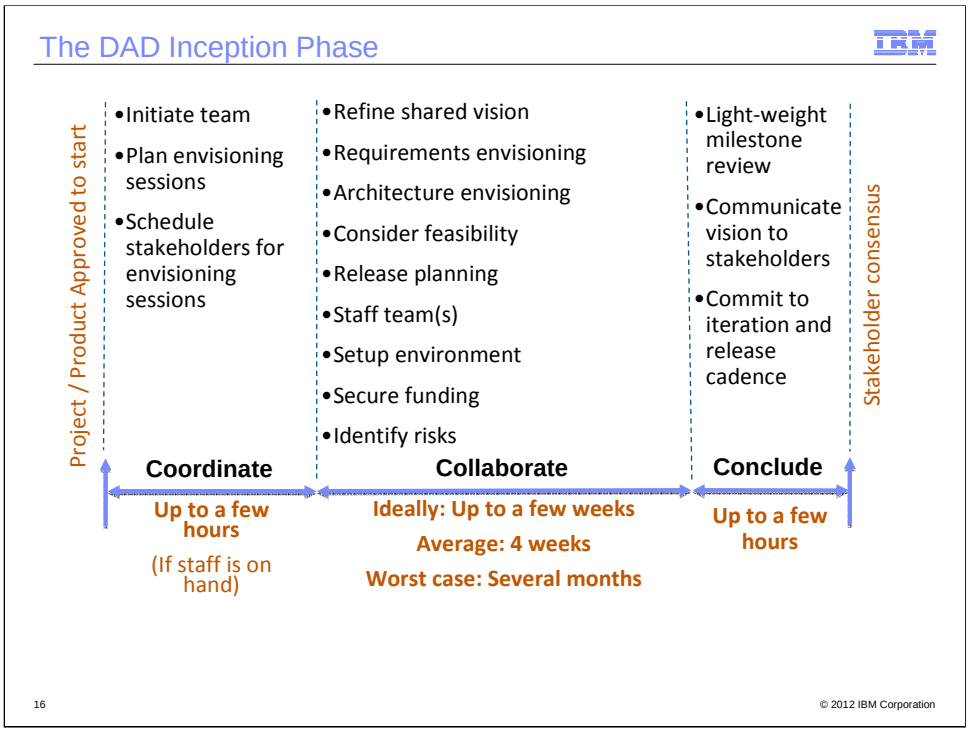
15

Consider things from the point of view of a team member. The daily stand up meeting is followed by work that, at the end of the day, is stabilized and committed. This cyclic rhythm takes place in another rhythm of an iteration. At a still higher level, you have the cyclic rhythm of Inception, Construction, and Transition.

Rhythms are important to keep a large team in sync as well as avoiding burnout.

Suggested reading:

Software Development Rhythms: Harmonizing Agile Practices for Synergy. Kim Man Lui and Keith C. C. Chan (2008)



There are several activities that occur iteratively throughout Inception:

•Build the initial team – people build solutions

•Develop a shared vision – what are the goals? Who are the stakeholders? What are you trying to achieve?

•Initial requirements envisioning – What is the scope?

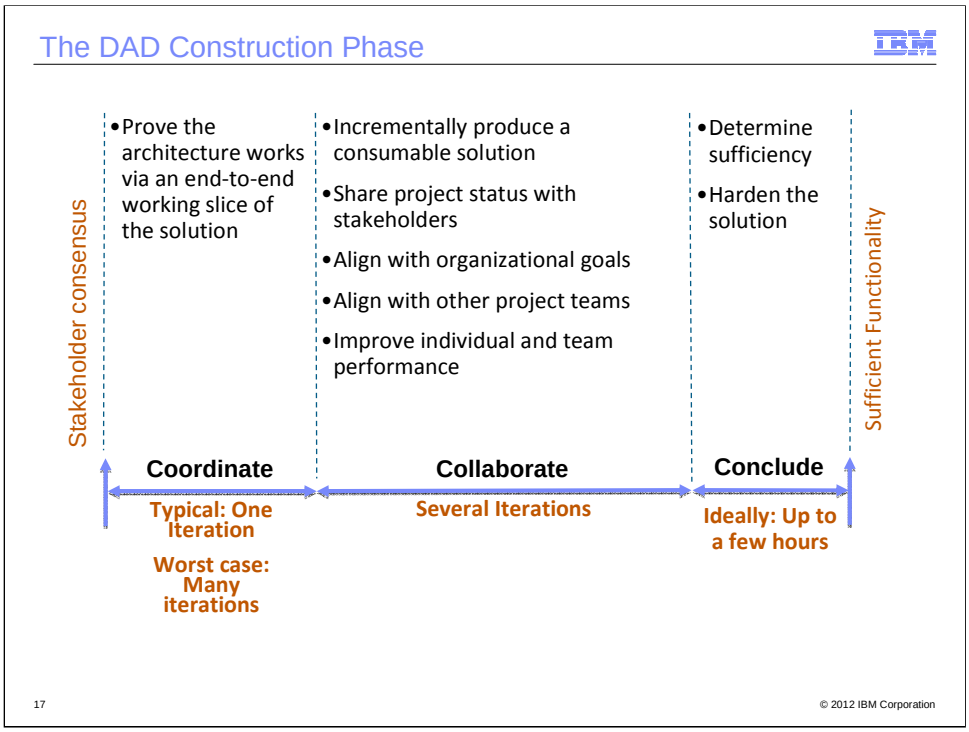
•Initial architecture envisioning – What is the technical vision?

•Align with enterprise strategy – How does what we're doing fit into the overall organizational strategy?

•Setup the environment – The team needs tools, a place to work, and other resources

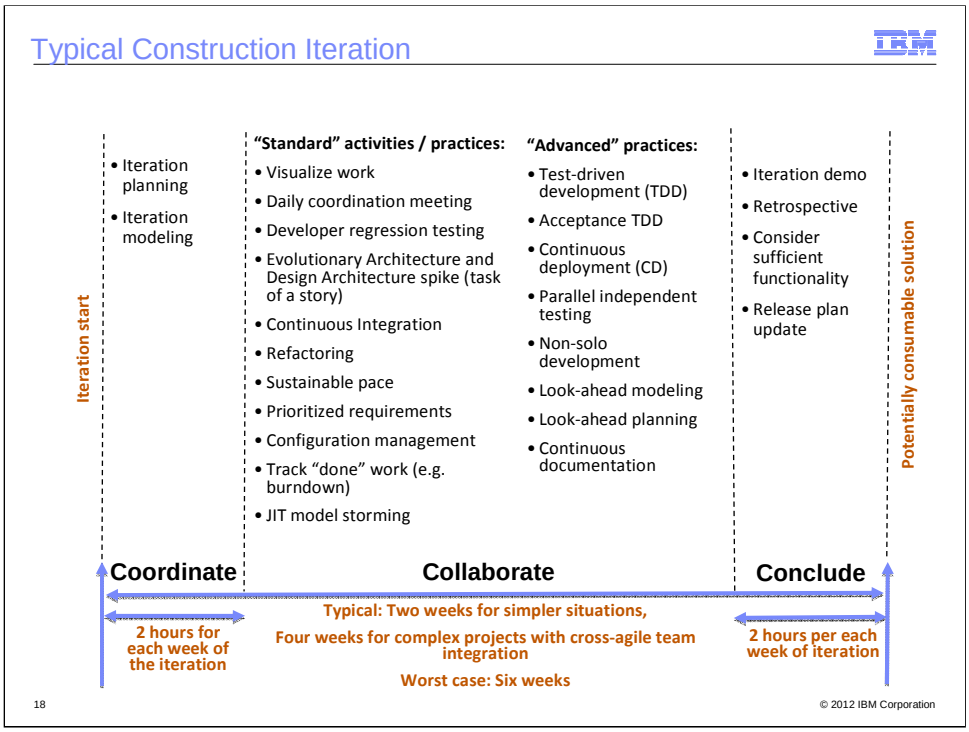
•Initial high-level release planning – How long will the project take? What will be the cost?

Diagram used with permission from the forthcoming book "Disciplined Agile Delivery (DAD): An Introduction" by Scott Ambler and Mark Lines publication date June 2012)



The way that you address these goals will vary depending on your situation.

Diagram used with permission from the forthcoming book “Disciplined Agile Delivery (DAD): An Introduction” by Scott Ambler and Mark Lines publication date June 2012)



Iterations are also referred to as time boxes and sprints (in Scrum).

At the beginning of the iteration the team plans what they are going to do and how they're going to do it that iteration. This may include some modeling.

Throughout the iteration the team does the work to address the work items for that iteration.

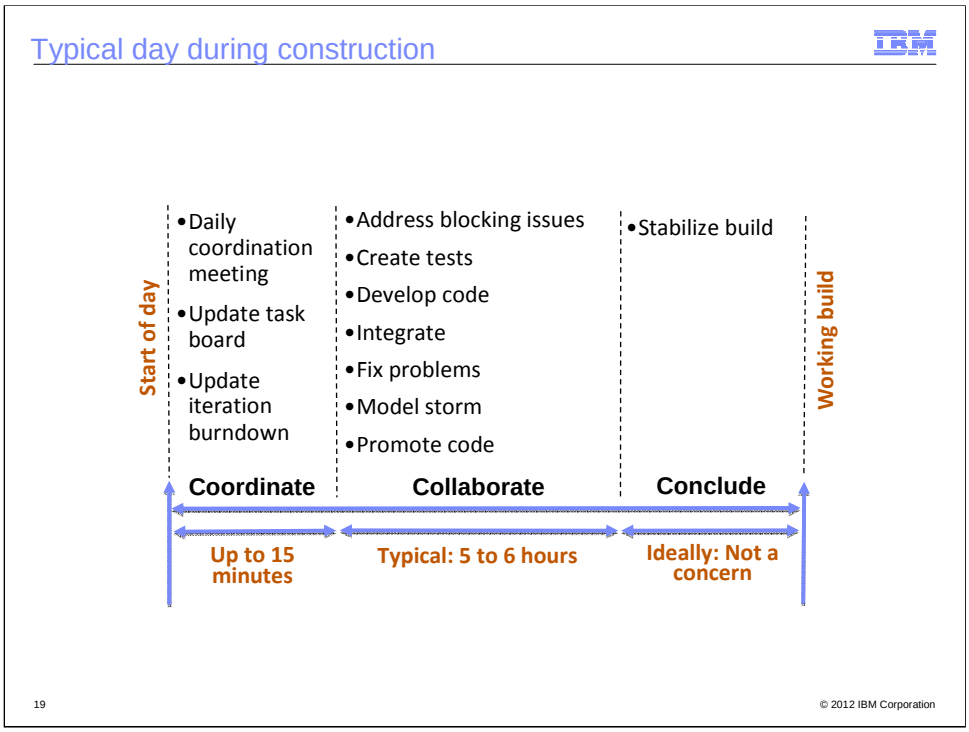
Toward the end of the iteration the team stabilizes the solution, ensuring that it works and is sufficiently tested. The team will also demo the solution to key stakeholders to show what they accomplished and to get feedback. They may also hold a reflection meeting, such as a retrospective, to identify potential ways to improve their process.

The iteration rhythm is determined by the iteration length. Fixed iteration length helps drive the reliable rhythm of a project. Without this rhythm, you are constant revising, re-estimating, and reconciling

Agile principle #7: Working software is the primary measure of progress.

Agile principle #12: At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly

Diagram used with permission from the forthcoming book "Disciplined Agile Delivery (DAD): An Introduction" by Scott Ambler and Mark Lines publication date June 2012)



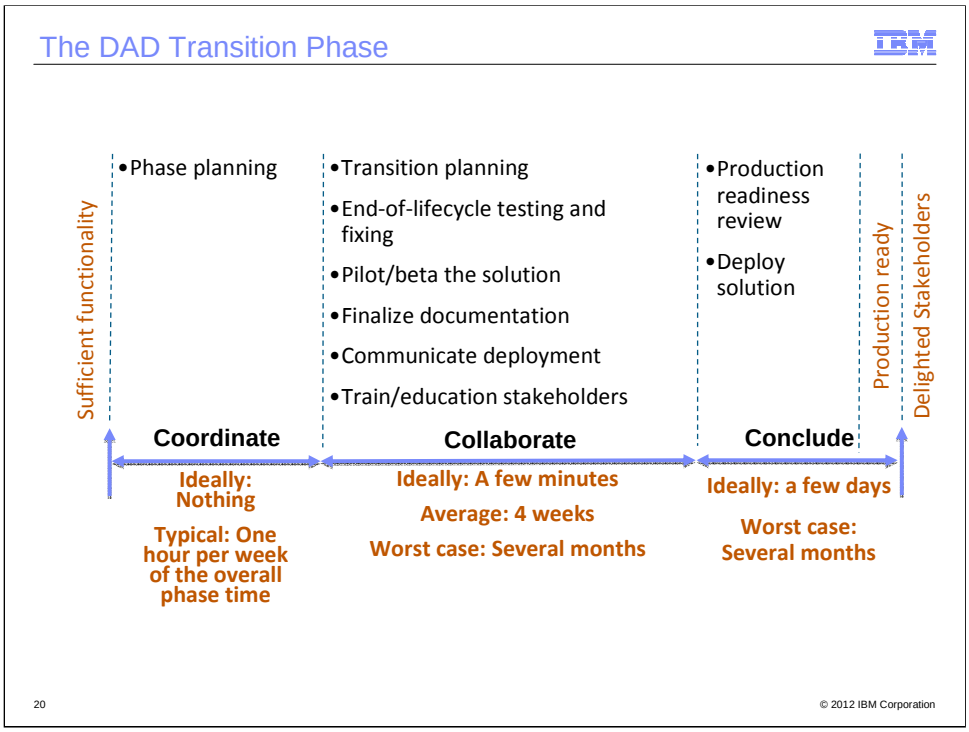
The day starts with a short meeting to coordinate everyone's activities.

The majority of the day is focused on working on the tasks.

At the end of the day you hopefully have a working build of the solution. You may also have working builds throughout the day as well.

Notice that activities around picking up email, attending non-team related meetings, … aren't included. These sorts of activities are outside the scope of a "DAD day" although clearly they still need to occur. Many DAD teams will leave enough time in the morning and at the end of day for people to deal with email. Interactions with other teams, such as reviewing their work and providing feedback, should be scheduled on an as-needed basis.

Diagram used with permission from the forthcoming book "Disciplined Agile Delivery (DAD): An Introduction" by Scott Ambler and Mark Lines publication date June 2012)



Coordinate:

•For teams that deploy continuously, on the order of days or a few weeks, then your construction efforts will be so disciplined that you don't need to do much planning for this phase because you simply follow your regular procedures, many of which are likely to be automated

Collaborate:

•This could be a huge range of effort.

•For continuous deployment environments, this may be virtually nothing more than doing a final build and regression test run (which in itself could run for days or weeks depending on the complexity of the solution).

•For environments where you don't deploy very often this often gets drawn out because you likely won't have automated this effort and because the deployments include more functionality (thereby increasing the complexity of the deployment effort, potentially exponentially).

Conclude:

-Do light-weight review for production readiness.

-This is one deployment step that you don't want to automate typically, but instead someone(s) must make a conscious decision to deploy.

Diagram used with permission from the forthcoming book "Disciplined Agile Delivery (DAD): An Introduction" by Scott Ambler and Mark Lines publication date June 2012)

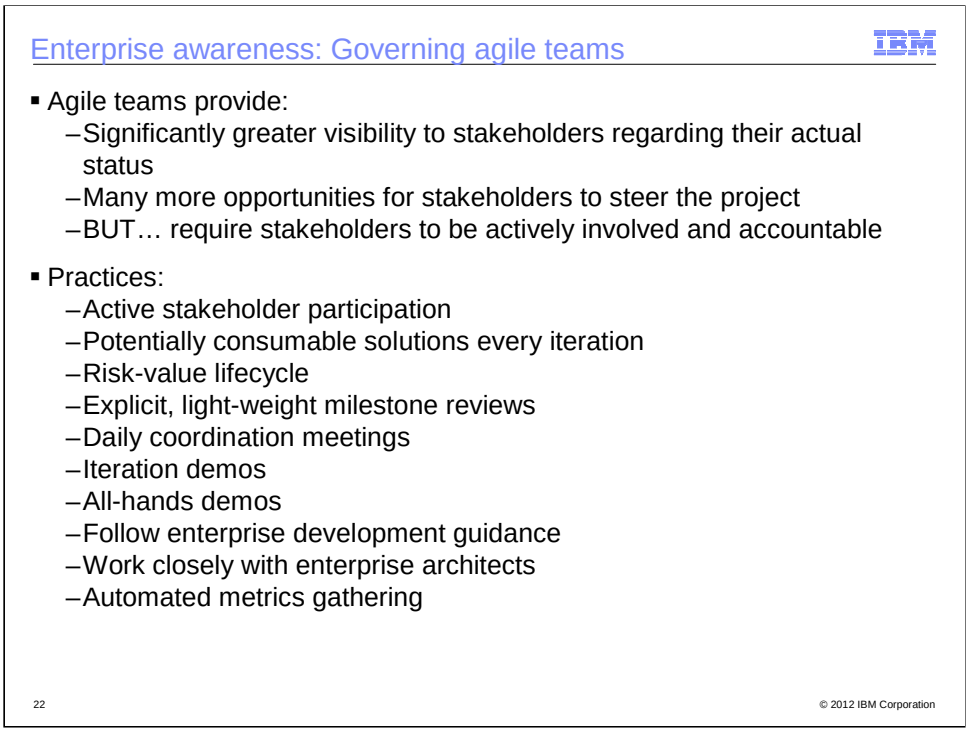
Enterprise awareness: Optimizing the whole

- Follow corporate conventions:
  - Standards and guidance for the architecture
  - Coding standards
  - Data guidelines
  - User interface (UI) guidelines
  - And many more
- Enhance the organizational ecosystem:
  - Reusing and leveraging the existing infrastructure is great
  - Enhancing and building out the infrastructure is better
  - Work closely with the enterprise architecture (EA) team
- Share learnings:
  - Personal and team improvement is great
  - Organization-level improvement is better
  - Your agile center of competency should harvest and share improvements
- Interact with other (potentially non-agile) teams:
  - Enterprise architecture
  - Data management
  - Governance
  - Quality assurance
  - Project management office (PMO)

DAD teams work within your organization's enterprise ecosystem, as do other teams, and explicitly try to take advantage of the opportunities presented to them – to coin an environmental cliché "disciplined agilists act locally and think globally." This includes working closely with: enterprise technical architects and reuse engineers to leverage and enhance. the existing and "to be" technical infrastructure; enterprise business architects and portfolio managers to fit into the overall business ecosystem; senior managers who should be governing the various teams appropriately; data administrators to access and improve existing data sources; and IT development support people to understand and follow enterprise IT guidance (such as coding, user interface, security, and data conventions to name a few). In other words, DAD teams should adopt a "whole enterprise" mindset.

With the exception of start-up companies, agile delivery teams don't work in a vacuum. There are often existing systems currently in production, and minimally your solution shouldn't impact them although hopefully your solution will leverage existing functionality and data available in production. There are often other teams working in parallel to your team, and you may wish to take advantage of a portion of what they're doing and vice versa. There may be a common vision which your organization is working towards, a vision which your team should contribute to. There will be a governance strategy in place, although it may not be obvious to you, which hopefully enhances what your team is doing.

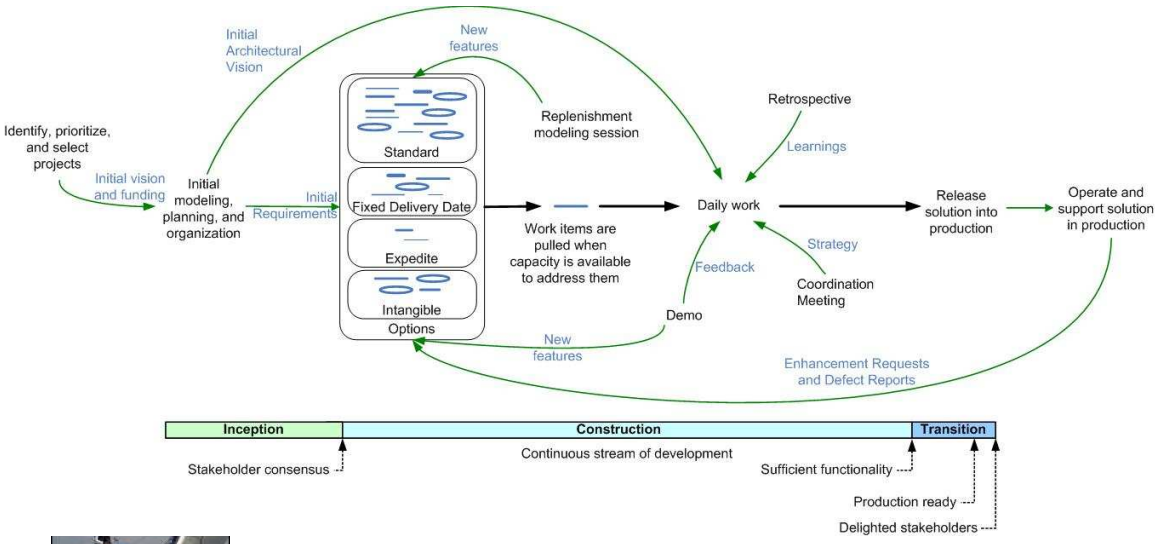
Enterprise awareness is an important aspect of self discipline because as a professional you should strive to do what's right for your organization and not just what's interesting for you. Unfortunately this isn't always the case. Some IT "professionals" will choose to work with new technologies, and even implement them in the solutions that they produce, not because those technologies are what's most appropriate for the organization but instead they help to improve the person's resume. Or they'll choose to build something from scratch, or use new development tools, or create new data sources, when perfectly good ones already exist within the organization.



Effective governance strategies should enhance that which is being governed. An appropriate approach to governing agile delivery projects, and we suspect other types of efforts, is based on motivating and then enabling people to do what is right for your organization. What is right will of course vary, but this typically includes motivating teams to take advantage of, and to evolve, existing corporate assets; following common guidelines to increase consistency, and working towards a shared vision for your organization. Appropriate governance is based on trust and collaboration. Appropriate governance strategies should enhance the ability of DAD teams to deliver business value to their stakeholders in a cost effective and timely manner.

Unfortunately many existing IT governance strategies are based on a command-and-control, bureaucratic approach which often proves ineffective in practice. Chapter 9 explores appropriate governance, the impact of traditional governance strategies, and how to adopt an appropriate governance strategy in greater detail.

The Disciplined Agile Delivery life cycle – Advanced



The Disciplined Agile Delivery (DAD) process framework is a people-first, learning-oriented hybrid agile approach to IT solution delivery. It has a risk-value lifecycle, is goal-driven, scalable, and is enterprise aware.

23

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.

The advanced DAD lifecycle is evolved from the basic DAD lifecycle in several important ways:

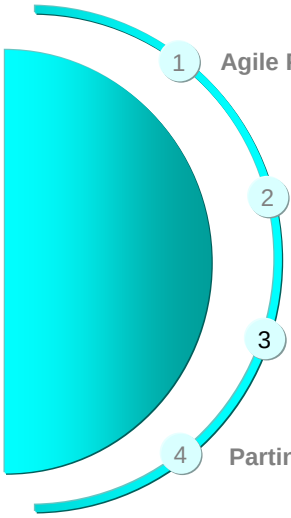
1. Work items are managed as a categorized pool, not a priortized stack. Work is pulled from the stack as capacity to perform it is available.

2. The cadences of various activities (planning, demos, retrospectives, and so on) are detached from each other. These activities are performed when needed, not based on a specific time during an iteration

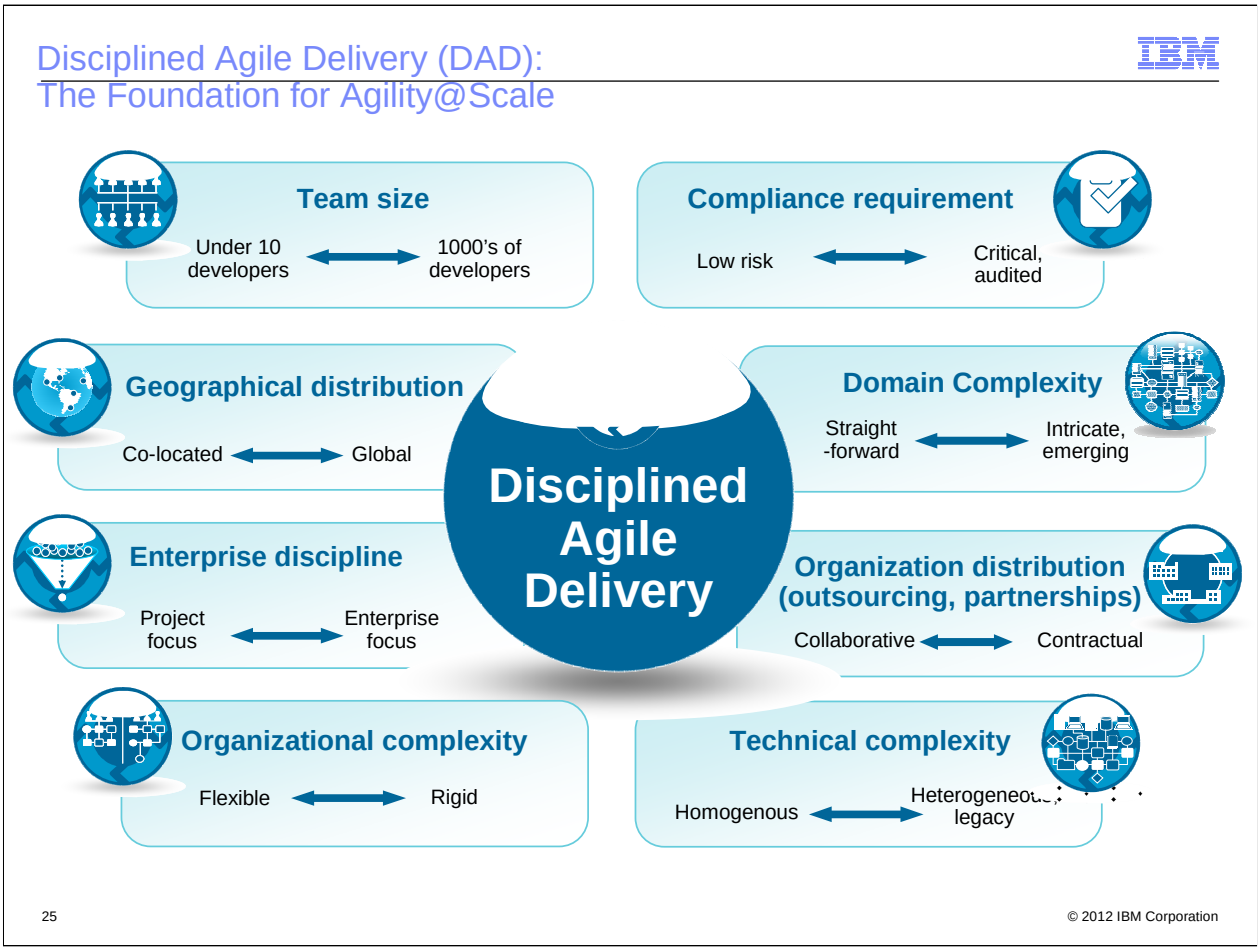
3. Transition should be very short, with most transition activities automated and any test/fix efforts performed during construction.

Diagram used with permission from the forthcoming book “Disciplined Agile Delivery (DAD): An Introduction” by Scott Ambler and Mark Lines publication date June 2012)

Agenda



1. Agile Fundamentals
2. Disciplined Agile Delivery (DAD)
3. **Agility at Scale**
4. Parting Thoughts



In the early days, agile development was applied to projects that were small in scope and relatively straightforward. Today, organizations want to apply agile development to a broader set of projects. Agile needs to adapt to increasing complexity. Agility@Scale is about explicitly addressing the complexities that disciplined agile delivery teams face in the real world. The agile scaling factors are:

Geographical distribution. What happens when the team is distributed within a building or across continents?

Team size. Mainstream agile processes work well for small teams (10-15), but but what if the team is fifty people? One hundred people? One thousand people?

Compliance requirement. What if regulatory issues – such as Sarbanes Oxley, ISO 9000, or FDA CFR 21 – are applicable?

Domain complexity. What if the problem domain is intricate (such as bio-chemical process monitoring or air traffic control), or is changing quickly (such as financial derivatives trading or electronic security assurance). More complex domains require greater exploration and experimentation, including but not limited to prototyping, modeling, and simulation.

Organization distribution. Sometimes a project team includes members from different divisions, different partner companies, or from external services firms.

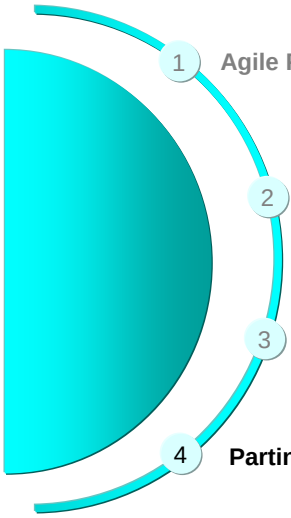
Technical complexity. Working with legacy systems, multiple platforms, or blending disparate technologies can add layers of technical complexity to a solution. Sometimes the nature of the problem is very complex in its own right.

Organizational complexity. The existing organizational structure and culture may reflect traditional values, increasing the complexity of adopting and scaling agile strategies. Different subgroups within the organization may have different visions as to how they should work. Individually, the strategies can be quite effective, but as a whole they simply don't work together effectively.

Enterprise discipline. Organizations want to leverage common infrastructure platforms to lower cost, reduce time to market, and to improve consistency. They need effective enterprise architecture, enterprise business modeling, strategic reuse, and portfolio management disciplines. These disciplines must work in concert with, and better yet enhance, the disciplined agile delivery processes.

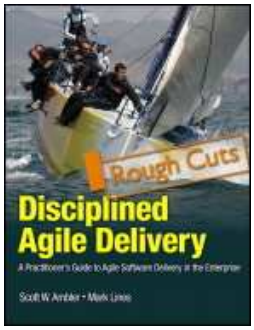
Each scaling factor has a range of complexities associated with it. Each team faces a different combination of factors, and therefore needs a process, team structure, and tooling environment tailored to meet their unique situation.

Agenda



1. Agile Fundamentals
2. Disciplined Agile Delivery (DAD)
3. Agility at Scale
4. **Parting Thoughts**

Get involved



Join the discussion today at

www.DisciplinedAgileDelivery.com

scott_ambler [at] ca.ibm.com

twitter.com/scottwambler

www.ibm.com/developerworks/mydeveloperworks/blogs/ambler/

www.ibm.com/rational/agile/

www.jazz.net

Thank you!

Some agile whitepapers on IBM.com

- The Agile Scaling Model (ASM): Adapting Agile Methods for Complex Environments
  - ftp://ftp.software.ibm.com/common/ssi/sa/wh/n/raw14204usen/RAW14204USEN.PDF
- Scaling Agile: An Executive Guide
  - ftp://public.dhe.ibm.com/common/ssi/sa/wh/n/raw14211usen/RAW14211USEN.PDF
- Achieving Agility at Scale
  - ftp://public.dhe.ibm.com/software/pdf/dk/rational/Achieving_Agility_at_Scale_-_Alan_W_Brown.pdf
- Improving Software Economics: Top 10 Principles of Achieving Agility at Scale
  - ftp://public.dhe.ibm.com/common/ssi/ecm/en/raw14148usen/RAW14148USEN.PDF
- Enable the Agile Enterprise Through Incremental Adoption of Practices
  - http://public.dhe.ibm.com/common/ssi/ecm/en/raw14077usen/RAW14077USEN.PDF
- Disciplined Agile Delivery: An Introduction
  - http://public.dhe.ibm.com/common/ssi/ecm/en/raw14261usen/RAW14261USEN.PDF

Disciplined Agile Delivery (DAD) offerings

DAD training (PMI approved, registered under provider number 1107)

- Introduction to disciplined agile delivery: Self-paced virtual class (16 PDUs)
- Advanced disciplined agile delivery: 3 days (21 PDUs)

Related Training

- Applying DAD with User Stories (RV037)
- Applying DAD with Use Cases (RV036)
- For more info, visit the Rational Agile Training Page at www.ibm.com/jct03001c/services/learning/ites.wss/us/en?pageType=page&c=Z340950B41583X73

DAD Services

- DAD with RTC quick start
- IBM Rational Rapid Deployment for Agile Delivery
- Collaborative Lifecycle Management (CLM) for IT

DAD Products

- The DAD process template for Rational Team Concert (RTC)