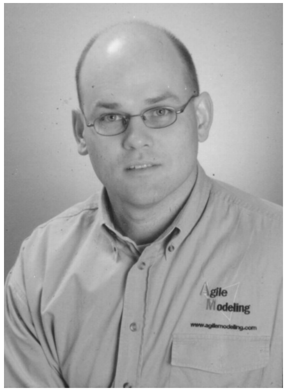
IBM Software Group



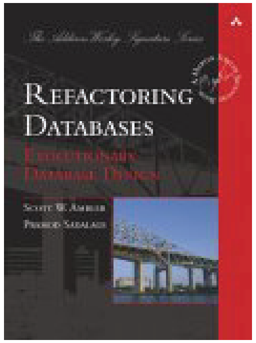
Agility at Scale

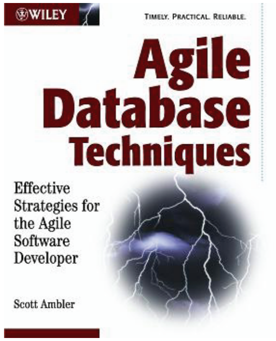
Scott W. Ambler
Practice Leader Agile Development

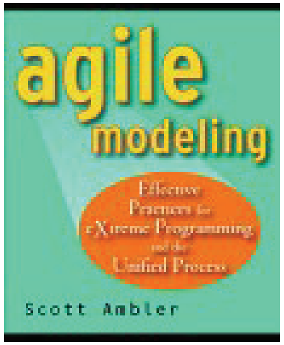
Rational. software

Scott Ambler - Background

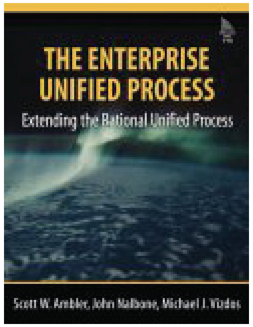
- Practice Leader Agile Development
- Fellow – International Association of Software Architects
- www-306.ibm.com/software/rational/bios/ambler.html

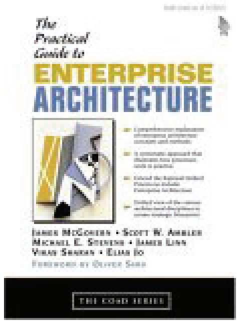


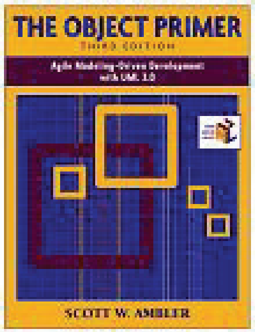

WILEY
TIMELY. PRACTICAL. RELIABLE.
Agile Database Techniques
Effective Strategies for the Agile Software Developer
Scott Ambler

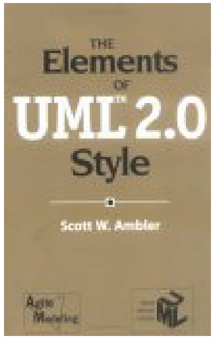

agile modeling
Effective Practices for eXtreme Programming and the Unified Process
Scott Ambler

eclipse
PROCESS FRAMEWORK
COMMITTER


THE ENTERPRISE UNIFIED PROCESS
Extending the Rational Unified Process
Scott W. Ambler, John Nalbone, Michael J. Vizdos







Discussion Topics

- Warning!
- Agile Software Development
- Agile Adoption Rate
- Scaling Agile
- Parting Thoughts

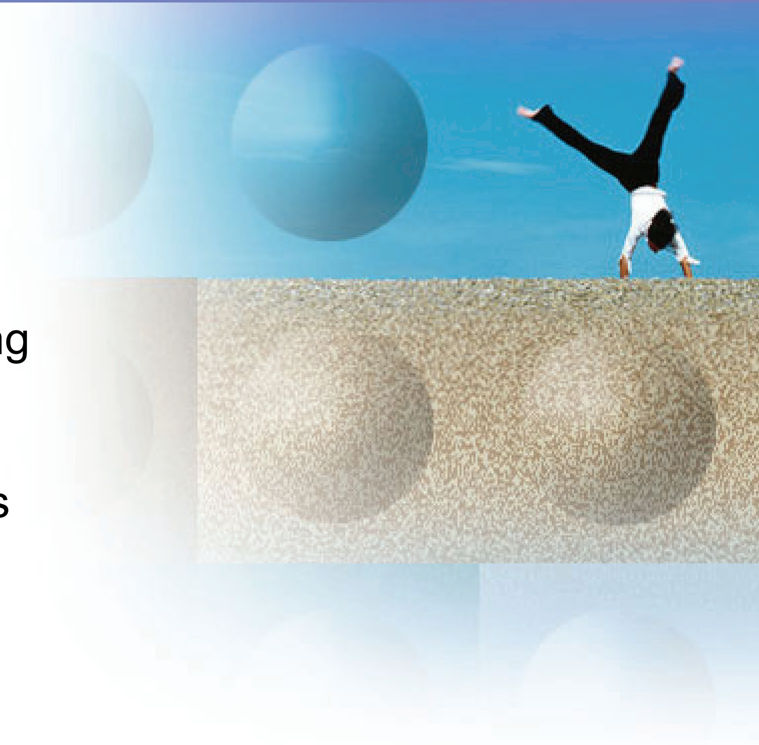
Warning!

- I’m spectacularly blunt at times
- Many new ideas will be presented
- Some may not fit well into your existing environment
- Some will challenge your existing notions about software development
- Some will confirm your unvoiced suspicions
- Don’t make any “career-ending moves”
- Be skeptical but open minded

Discussion Topics

- Warning!
- Agile Software Development
  - What is Agile?
  - How Agile is Different
  - Agile Mythbusters
  - Agile Practices
- Agile Adoption Rate
- Scaling Agile
- Parting Thoughts

What is Agile?



- An iterative and incremental (evolutionary) approach performed in a highly collaborative and self-organizing manner with **just the right amount of ceremony** to produce high quality software in a cost effective and timely manner which meets the changing needs of its stakeholders.

- Core principles
  - “Fits just right” process
  - Continuous testing and validation
  - Consistent team collaboration
  - Rapid response to change
  - Ongoing customer involvement
  - Frequent delivery of working software

How Agile is Different

- Focus on collaboration:
  - Less paperwork and more conversation
  - Stakeholders actively involved
- Focus on working software:
  - Greater feedback makes agile projects easier to manage
  - Less documentation is required
  - Less bureaucracy
- Agilists are generalizing specialists:
  - Less hand offs between people
  - Less 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

Mythbusters

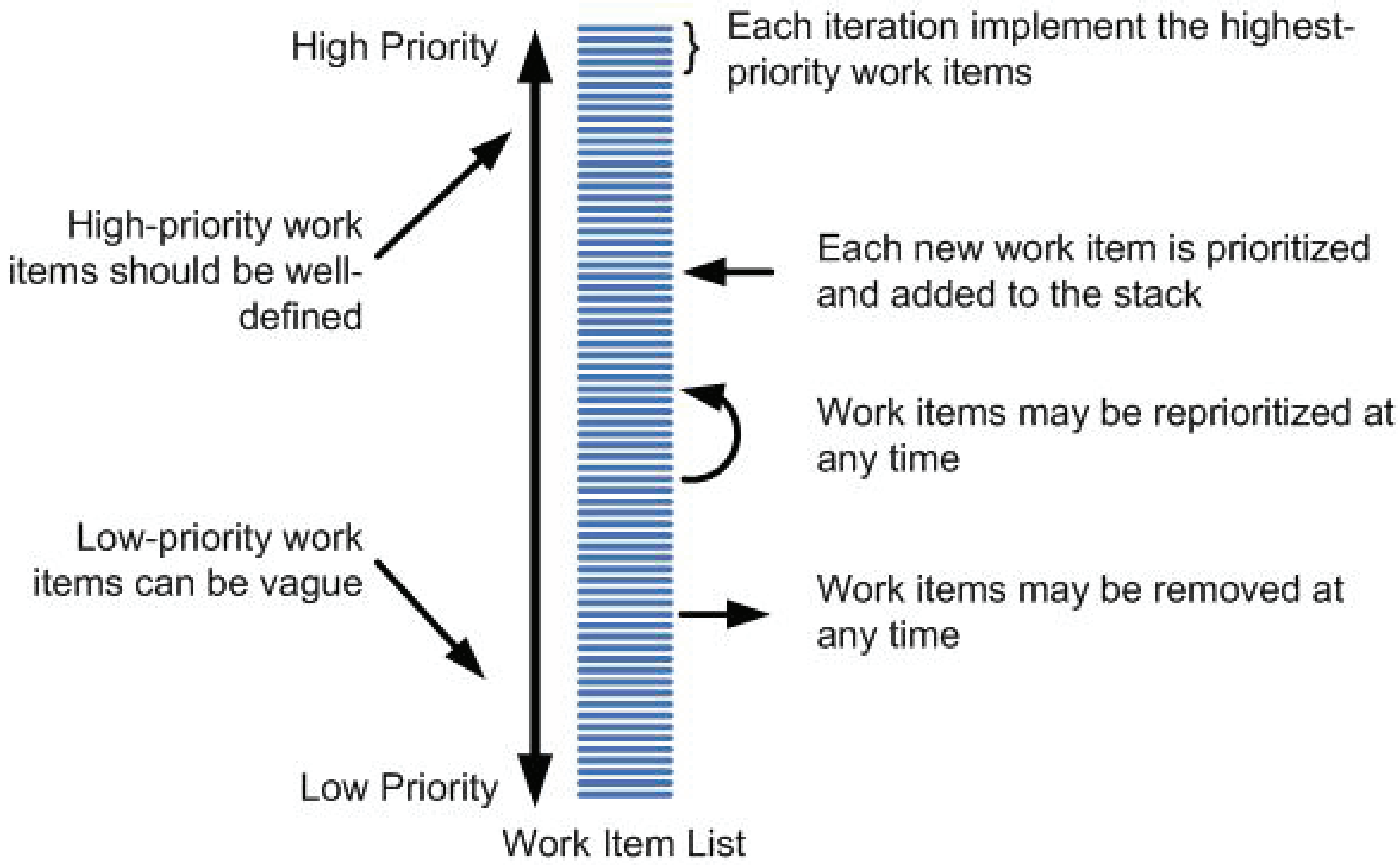
Myth	Reality
1. No Documentation	1. Agile Documentation
2. Undisciplined	2. Requires great discipline
3. No Planning	3. Just-in-time (JIT) planning
4. Not Predictable	4. Far more predictable
5. Does Not Scale	5. Eclipse is agile
6. Is a Fad	6. It’s quickly becoming the norm
7. Silver Bullet	7. It requires skilled people
8. RUP isn’t agile	8. RUP is as agile as you make it
9. Not Fixed Price	9. Agile provides stakeholders control over the budget, schedule, and scope

Mainstream Agile Practices

- Regular Deployment of Working Software
- Pair Programming
- Refactoring
- Continuous Integration
- Configuration Management
- Test Driven Development (TDD)
- Agile Testing
- Agile Documentation

Mainstream Practice: Working in Priority Order

www.agilemodeling.com/essays/agileRequirements.htm


High Priority
Each iteration implement the highest-priority work items
High-priority work items should be well-defined
Each new work item is prioritized and added to the stack
Work items may be reprioritized at any time
Low-priority work items can be vague
Work items may be removed at any time
Low Priority
Work Item List

Discussion Topics

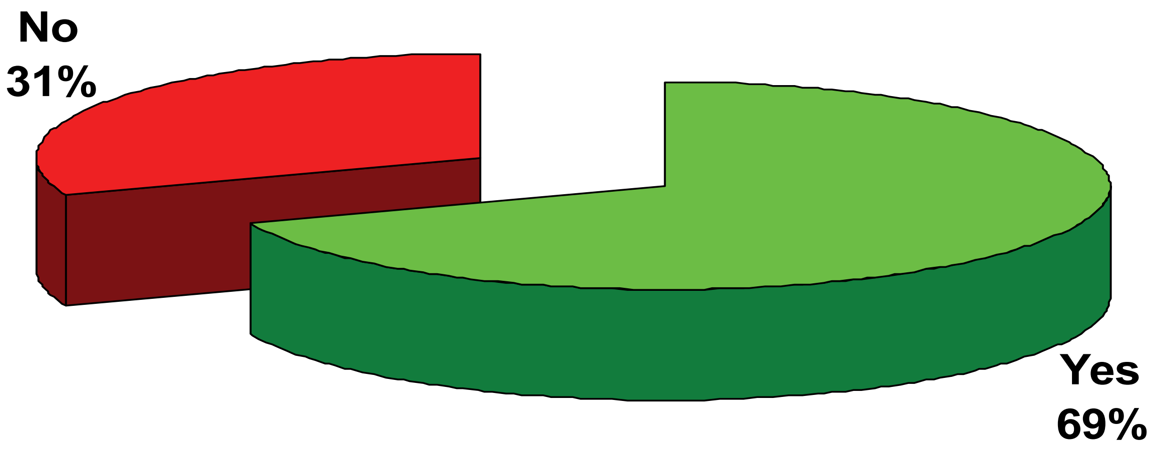
- Warning!
- Agile Software Development
- Agile Adoption Rate
  - Survey overview
  - Adoption rates
  - Success rates
  - Iteration lengths
- Scaling Agile
- Parting Thoughts

Dr. Dobb’s Journal Agile Adoption Survey

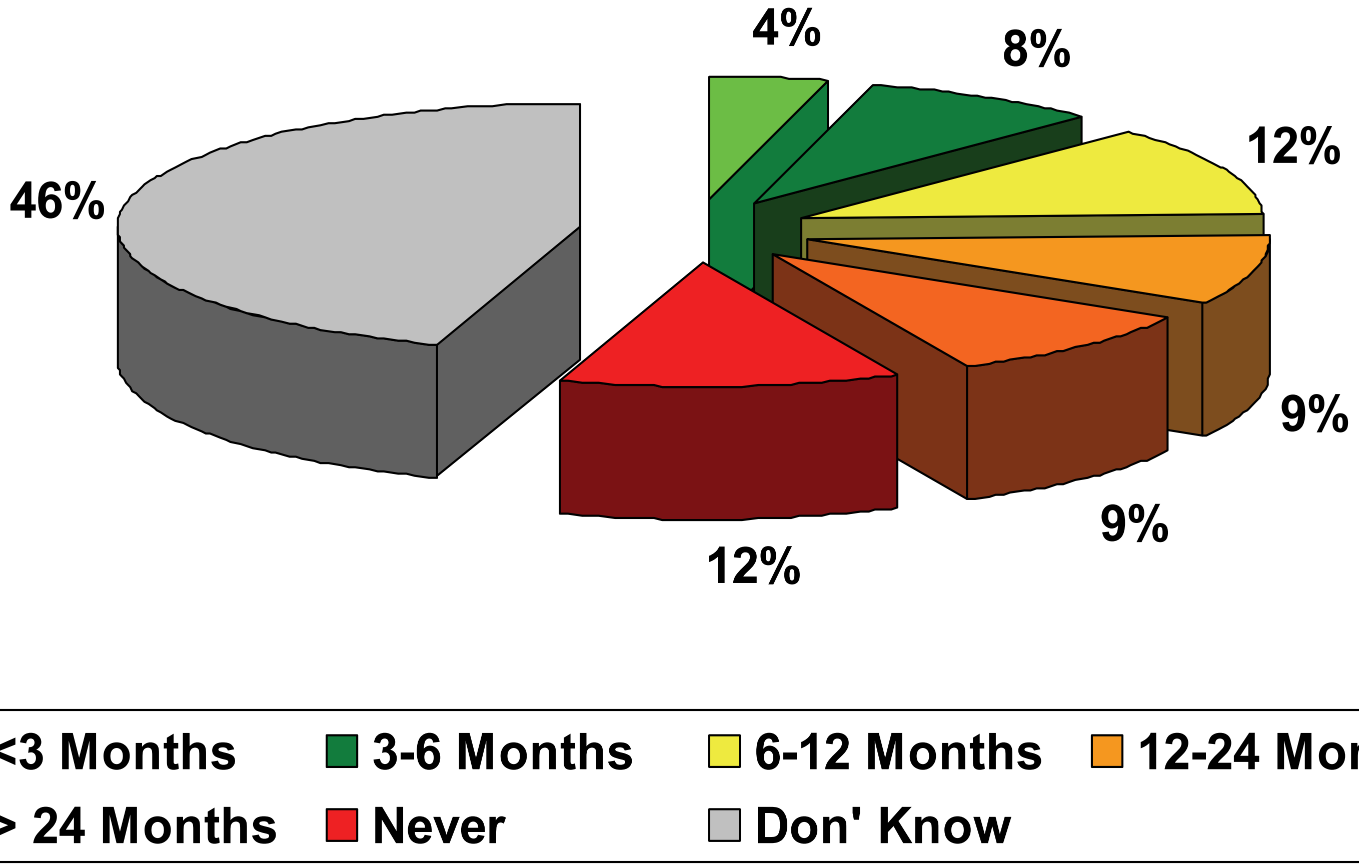
- March 2007
- Advertised in Editor’s blog on www.ddj.com
- Data, summary, and slides downloadable from www.ambysoft.com/surveys/agileMarch2007.html

- Respondents:
  - 781 in total
  - 52% were developers, 22% were in management
  - 40% had 10-20 years IT experience, 33% had 20+ years
  - 33% worked in orgs of 1000+ people
  - 85% worked in commercial firms
  - 95% North American

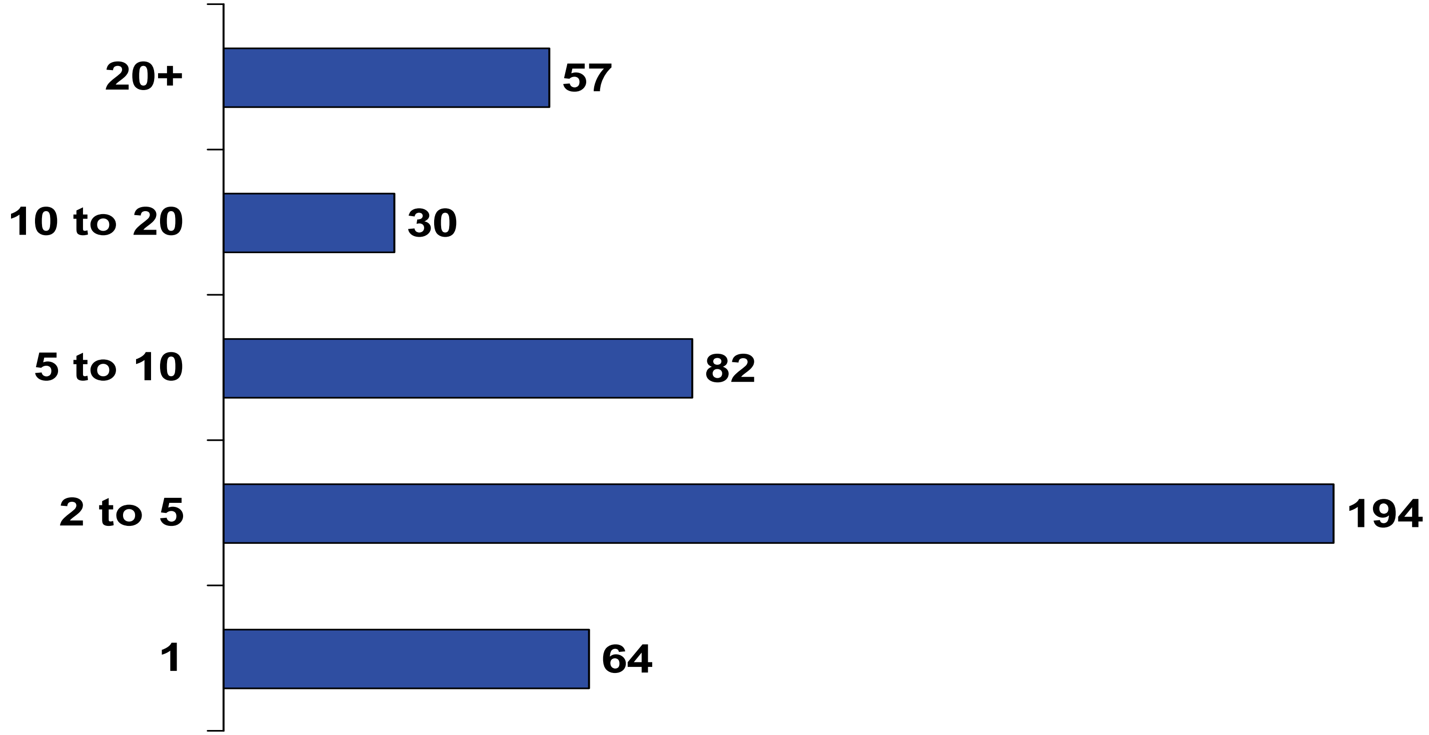
Has Your Organization Adopted One or More Agile Techniques?



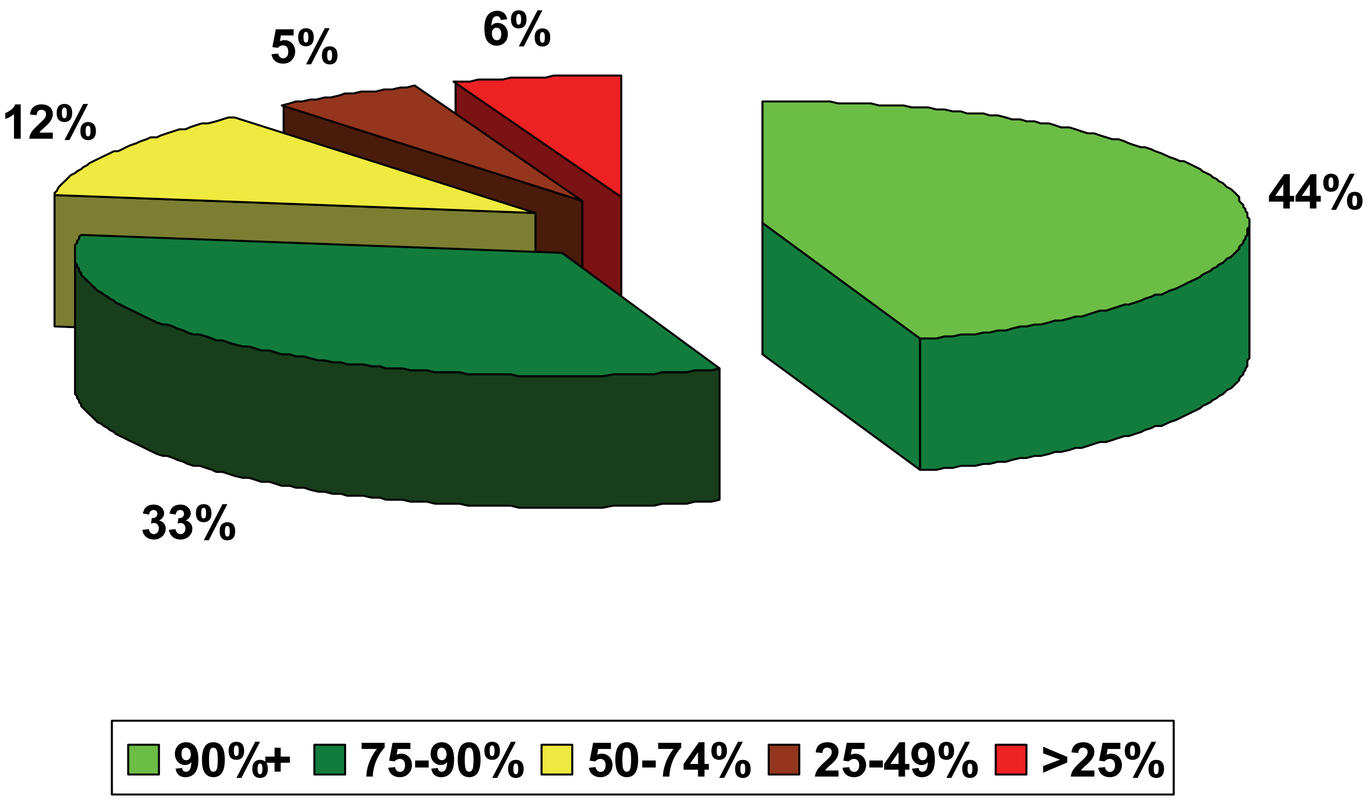
When Will You Adopt Agile? (241 said they hadn't yet)



Number of Agile Projects Run
85% have run multiple agile projects

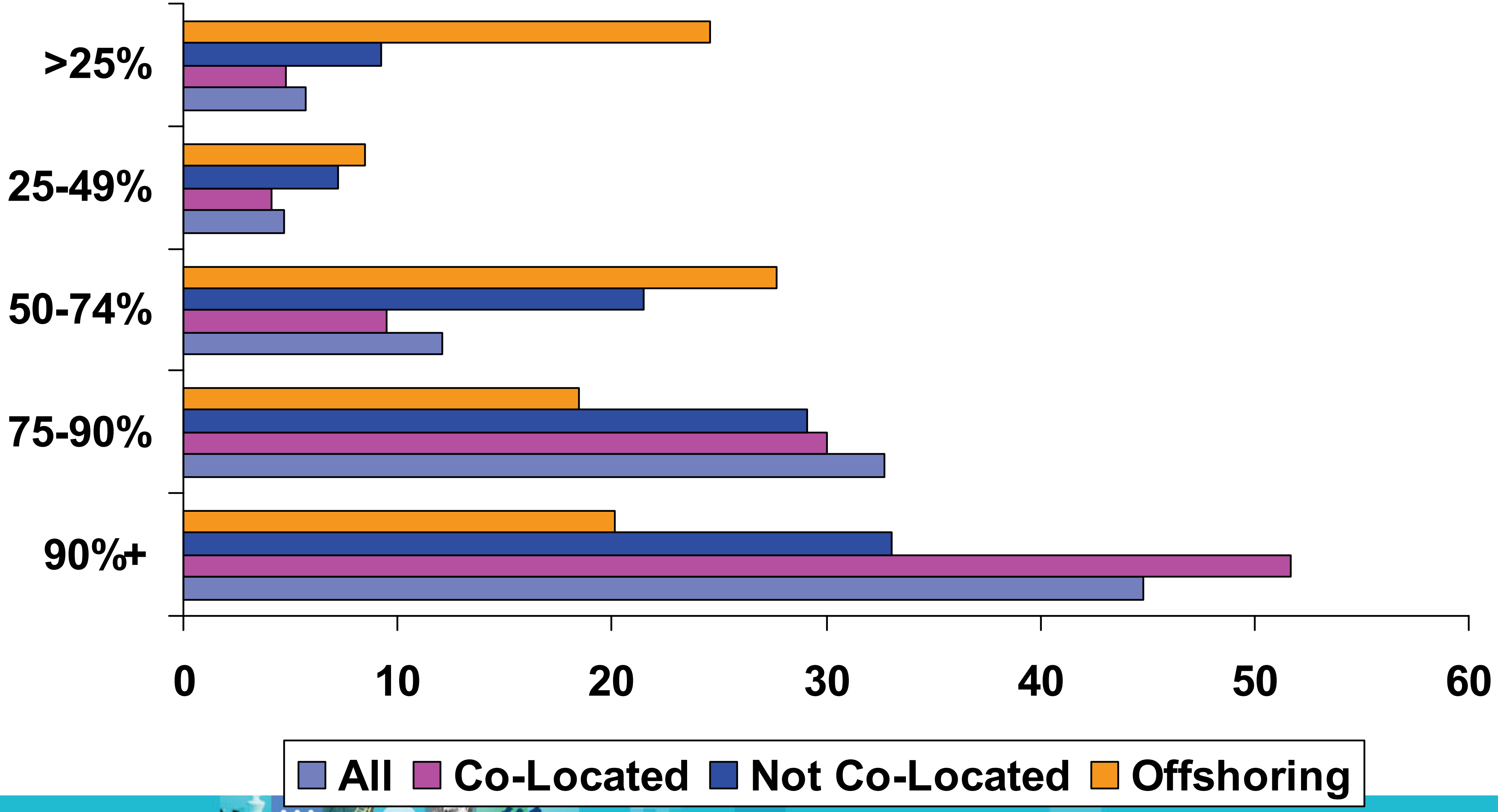


% of Successful Agile Projects

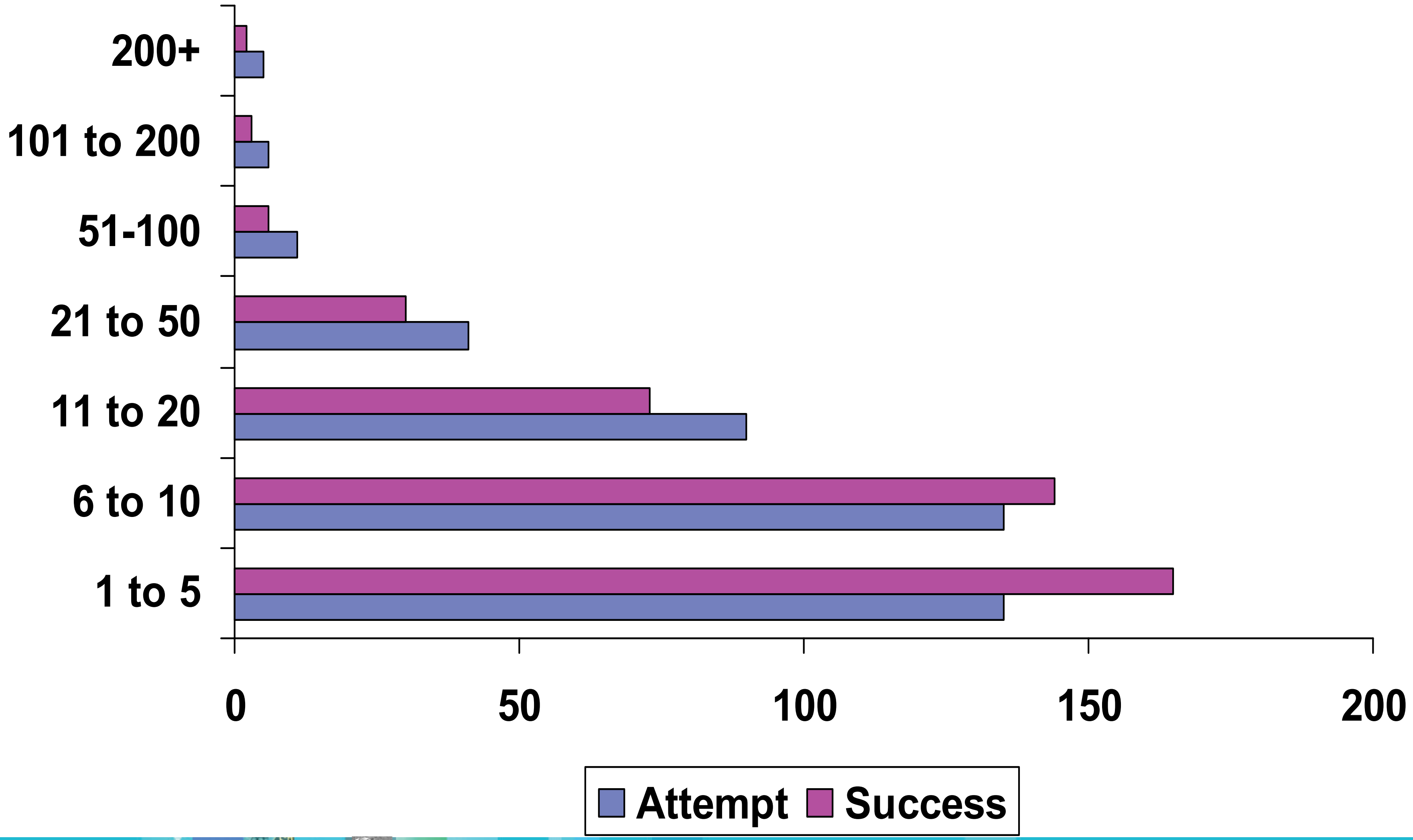


% of Successful Agile Projects

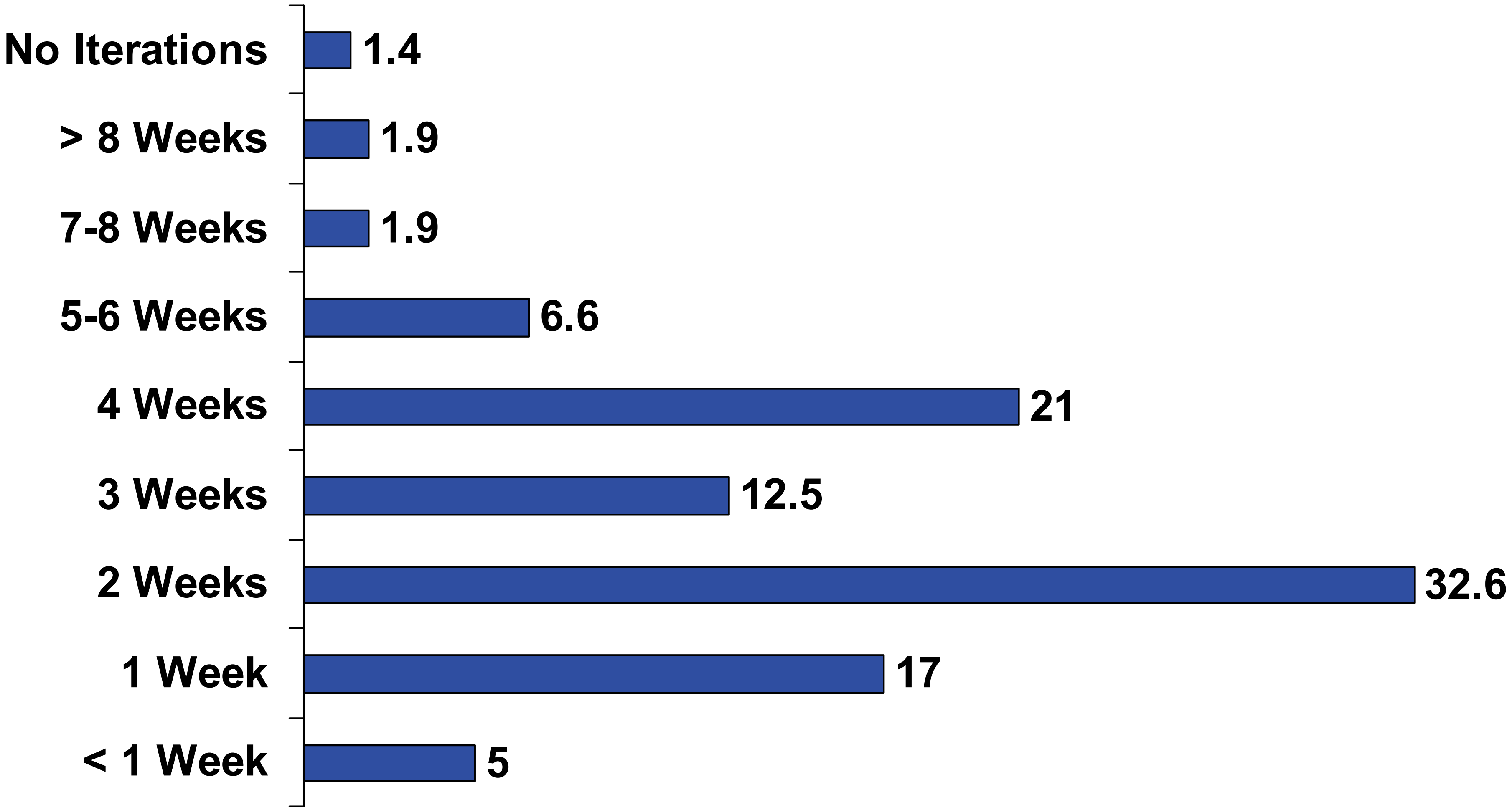
(296 co-located, 251 not co-location, 130 offshoring)

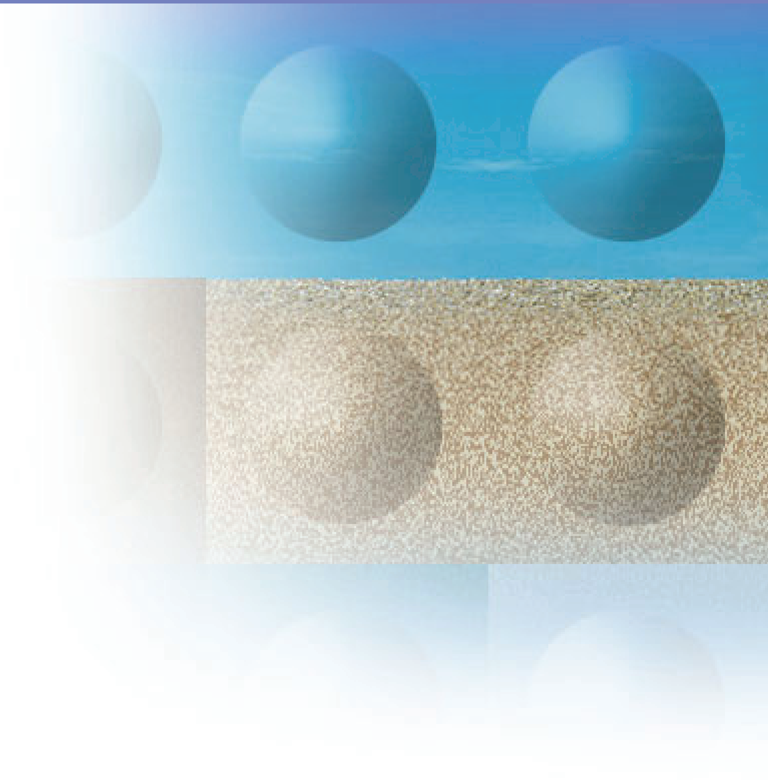


Largest Team Size Attempted vs. Successful



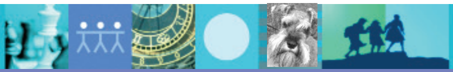
Length of Iterations (% respondents)
83% have iterations 1-4 weeks in length



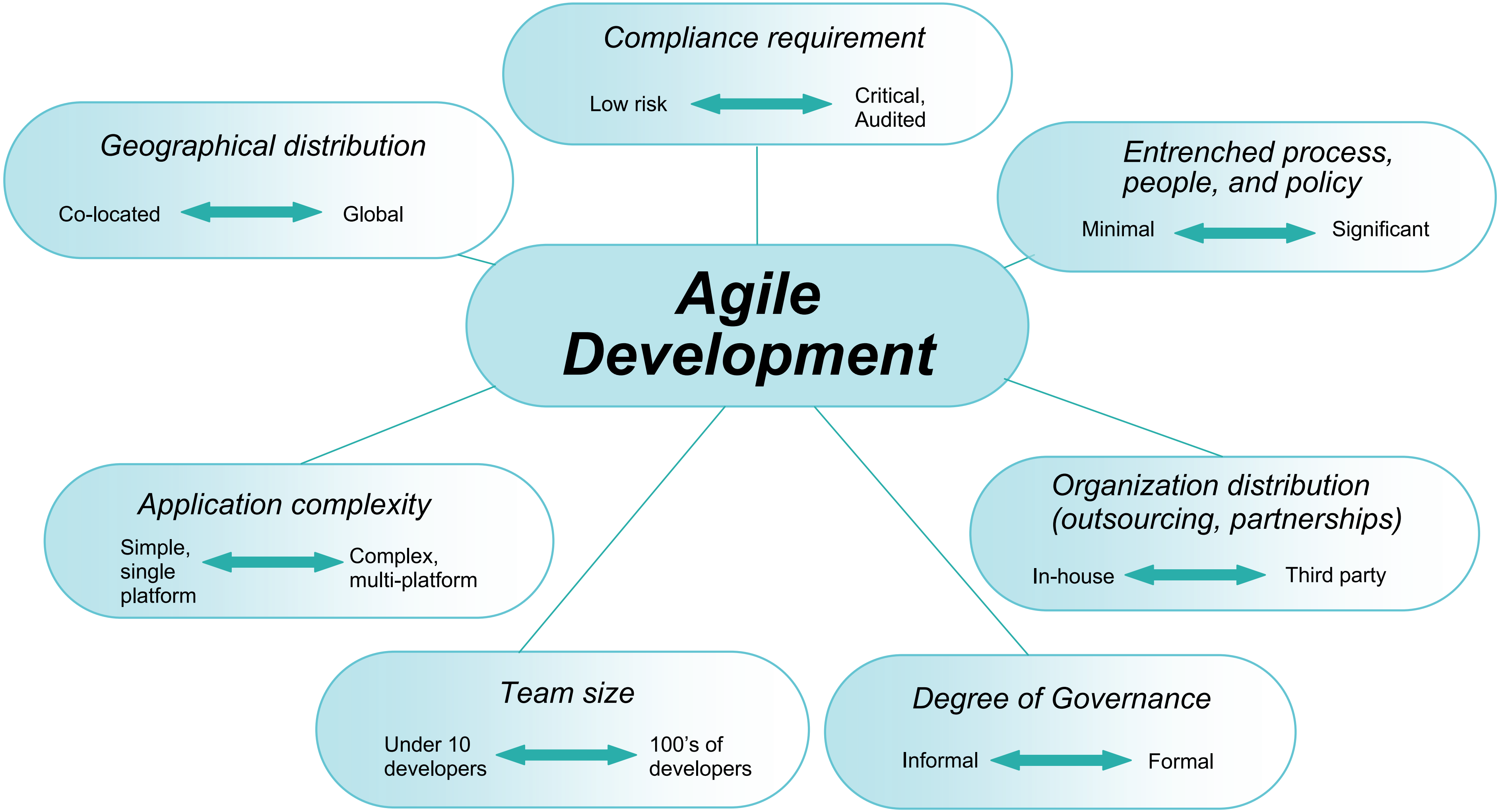


Discussion Topics

- Warning!
- Agile Software Development
- Agile Adoption Rate
- Scaling Agile
  - Challenges with Agile in the Mainstream
  - Agility is Relative
  - Process and Tooling
  - AMDD Practices
  - RUP Agile
  - Agile Data Practices
  - Lean Governance
- Parting Thoughts



Challenges with Agile in the Mainstream



Agile Development

Compliance requirement

Low risk ↔ Critical, Audited

Geographical distribution

Co-located ↔ Global

Entrenched process, people, and policy

Minimal ↔ Significant

Application complexity

Simple, single platform ↔ Complex, multi-platform

Organization distribution (outsourcing, partnerships)

In-house ↔ Third party

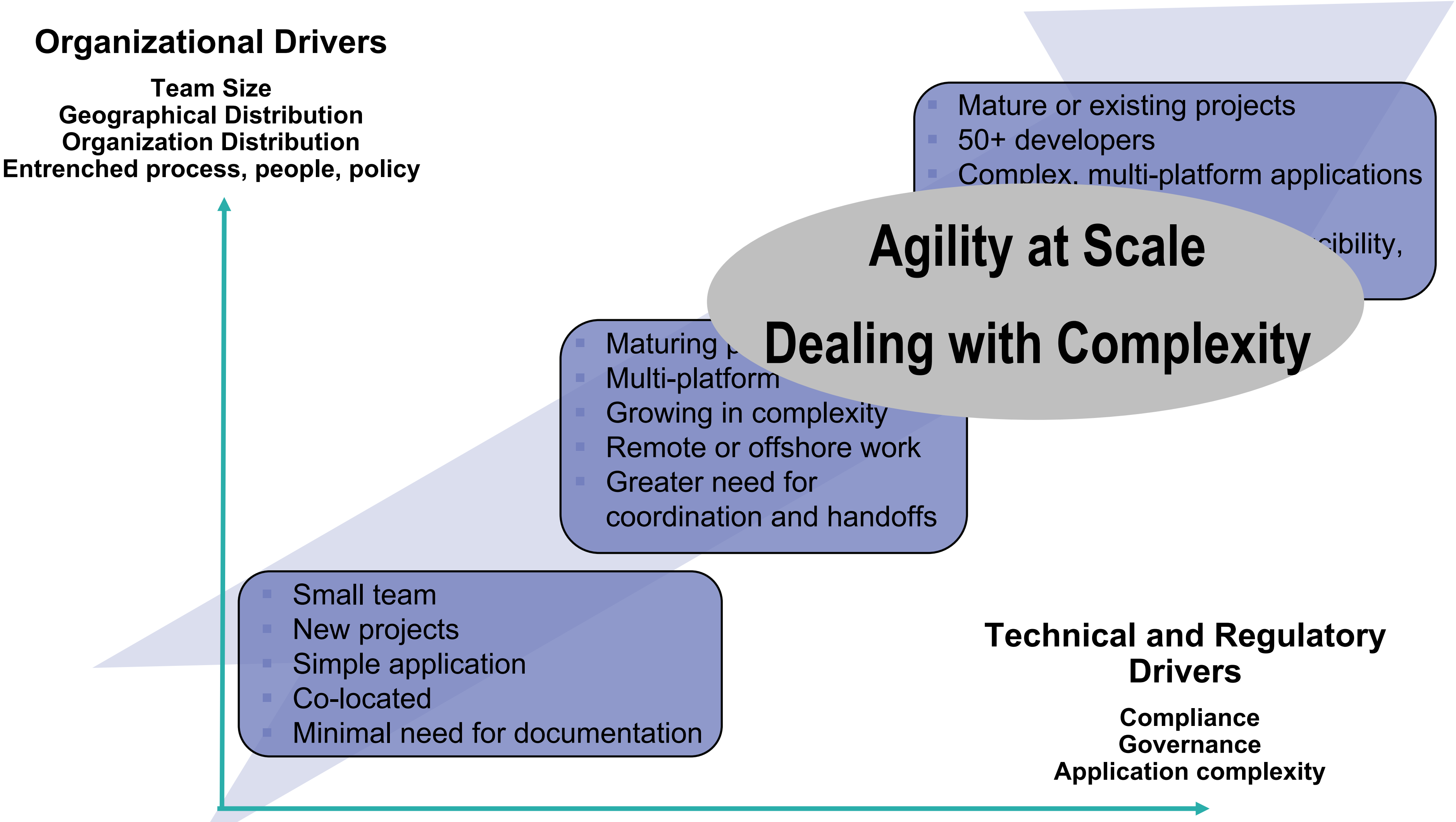
Team size

Under 10 developers ↔ 100’s of developers

Degree of Governance

Informal ↔ Formal

Agility is Relative – It Depends on Project Dynamics



Organizational Drivers

Team Size
Geographical Distribution
Organization Distribution
Entrenched process, people, policy

- Small team
- New projects
- Simple application
- Co-located
- Minimal need for documentation

- Maturing p
- Multi-platform
- Growing in complexity
- Remote or offshore work
- Greater need for coordination and handoffs

- Mature or existing projects
- 50+ developers
- Complex, multi-platform applications

cibility,

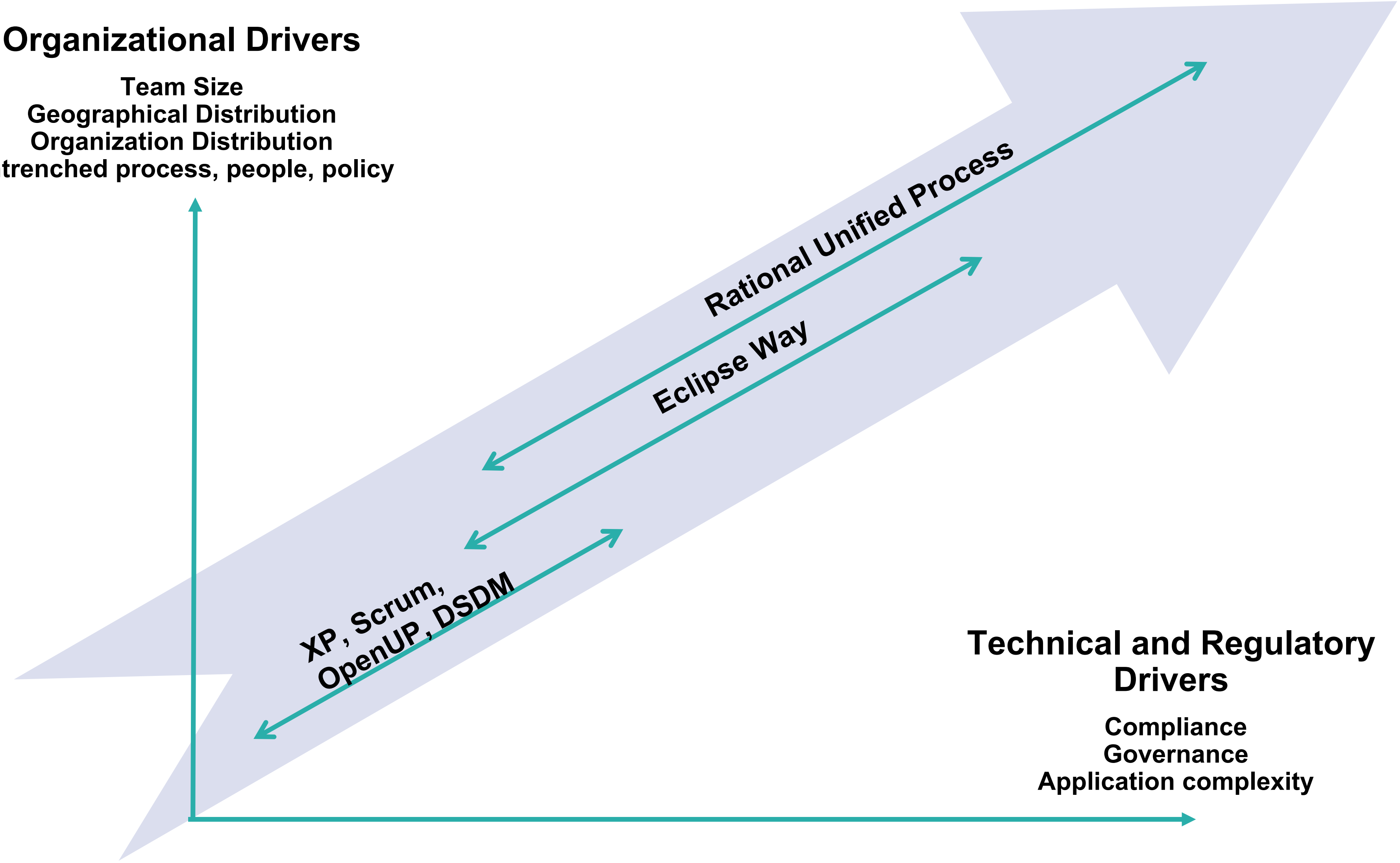
Agility at Scale

Dealing with Complexity

Technical and Regulatory Drivers

Compliance
Governance
Application complexity

The Agile Process Continuum



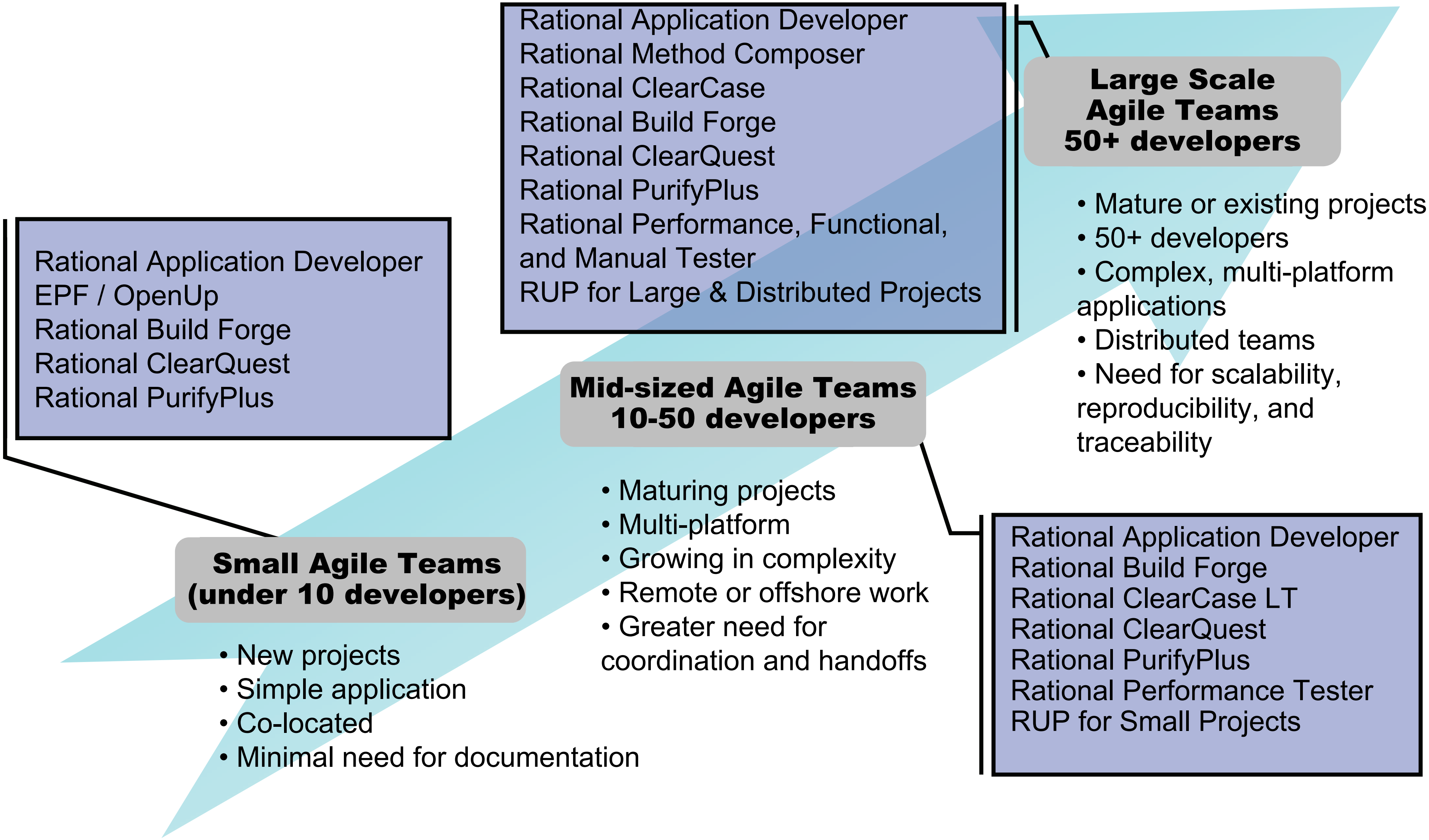
Organizational Drivers

Team Size
Geographical Distribution
Organization Distribution
Entrenched process, people, policy

Technical and Regulatory Drivers

Compliance
Governance
Application complexity

Solutions for Agile Teams of Every Size



Small Agile Teams (under 10 developers)

- New projects
- Simple application
- Co-located
- Minimal need for documentation

Rational Application Developer
EPF / OpenUp
Rational Build Forge
Rational ClearQuest
Rational PurifyPlus

Mid-sized Agile Teams 10-50 developers

- Maturing projects
- Multi-platform
- Growing in complexity
- Remote or offshore work
- Greater need for coordination and handoffs

Rational Application Developer
Rational Build Forge
Rational ClearCase LT
Rational ClearQuest
Rational PurifyPlus
Rational Performance Tester
RUP for Small Projects

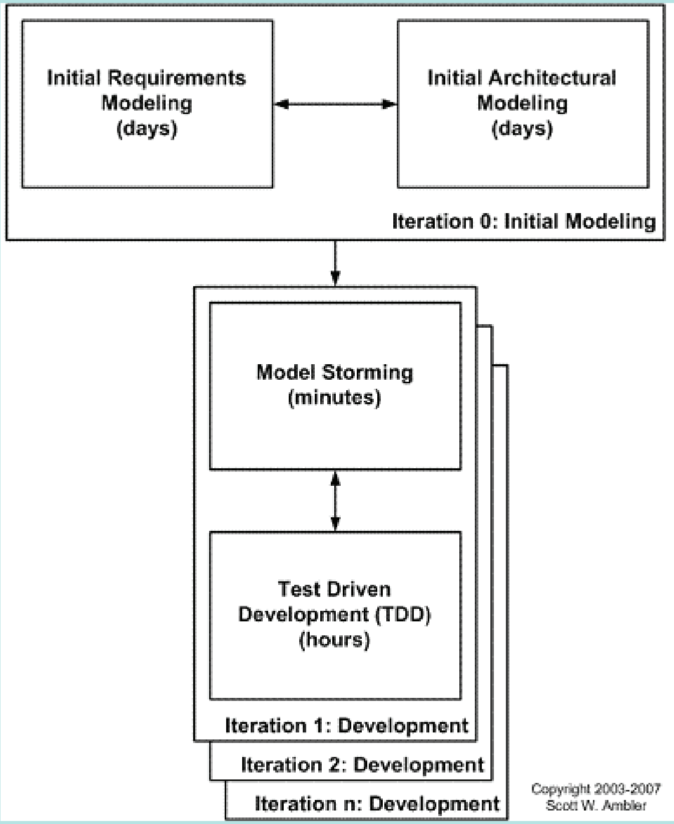
Large Scale Agile Teams 50+ developers

- Mature or existing projects
- 50+ developers
- Complex, multi-platform applications
- Distributed teams
- Need for scalability, reproducibility, and traceability

Rational Application Developer
Rational Method Composer
Rational ClearCase
Rational Build Forge
Rational ClearQuest
Rational PurifyPlus
Rational Performance, Functional, and Manual Tester
RUP for Large & Distributed Projects

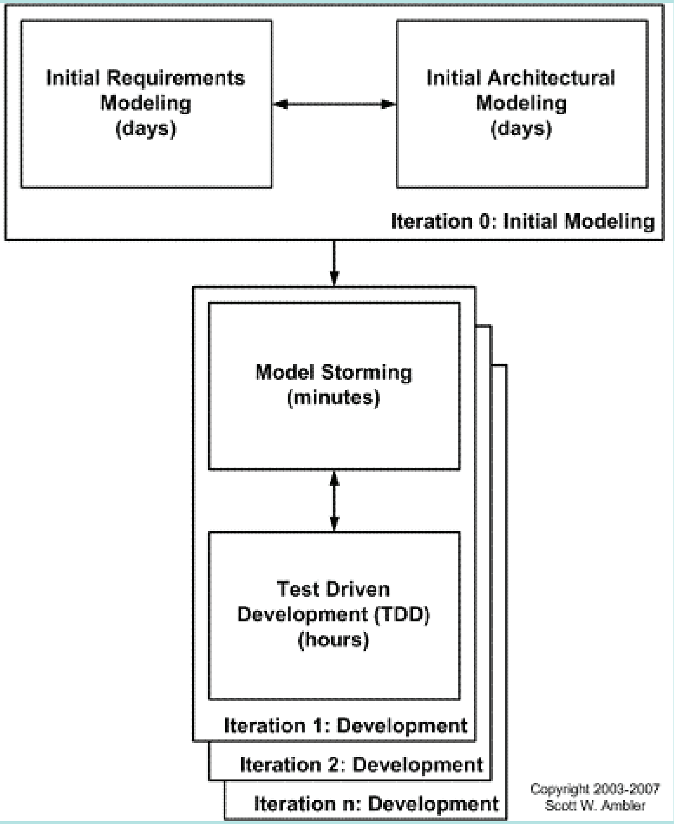
Initial Requirements Modeling

- Your goals are to:
  1. Identify and agree to the initial scope of your project
  2. Develop the initial stack of requirements
  3. Gather enough information to address initial scheduling and estimating concerns
- Critical models for business application development:
  - Some sort of usage model (use cases, user stories, …)
  - Conceptual/domain model
  - Some UI sketches


Initial Requirements Modeling (days)
Initial Architectural Modeling (days)
Iteration 0: Initial Modeling
Model Storming (minutes)
Test Driven Development (TDD) (hours)
Iteration 1: Development
Iteration 2: Development
Iteration n: Development
Copyright 2003-2007 Scott W. Ambler

Initial Architectural Modeling

- Your goals are to:
  1. Identify and agree to a potential initial architecture of your system
  2. Provide sufficient technical vision for estimating and scheduling concerns
- Critical models for business application development:
  - Some form of deployment diagram
  - A free-form "technology stack" diagram
  - A UI navigation diagram


Initial Requirements Modeling (days)
Initial Architectural Modeling (days)
Iteration 0: Initial Modeling
Model Storming (minutes)
Test Driven Development (TDD) (hours)
Iteration 1: Development
Iteration 2: Development
Iteration n: Development

Copyright 2003-2007 Scott W. Ambler

Model Storming

- Just-in-time (JIT) modeling
- Model when you run into “difficulty”
  - You have a new feature to implement, and need to “analyze” the details
  - You need to think through a technical issue
- Most modeling on agile projects is done in 10-15 minute bursts followed by hours or days of programming
- Waiting to model is more effective:
  - You understand the domain better
  - Stakeholders understand the current solution better
  - You can focus only on what you need to build

Tooling strategies:

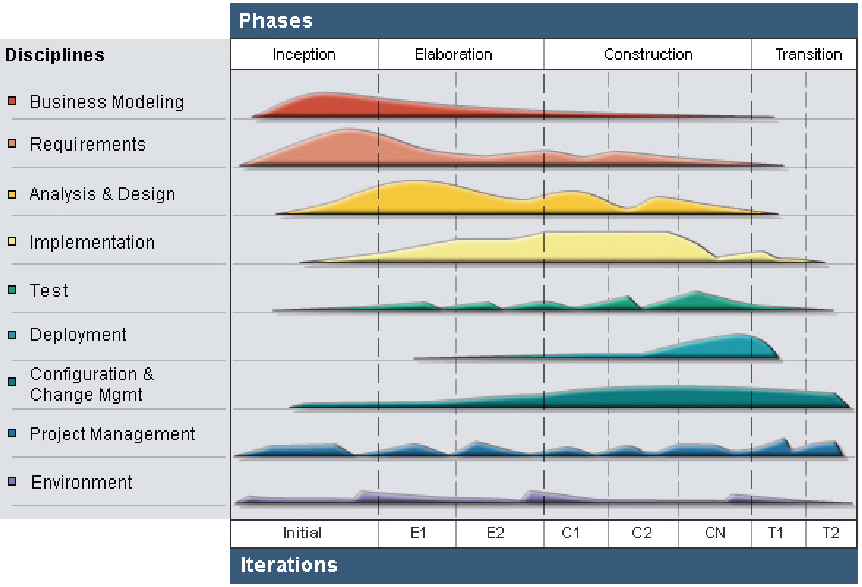
- Choose tools based on skill and need
- Sometimes the “simplest” tool is a complex one
- Whiteboards and paper are great for thinking things though, not so good for capturing information permanently
- Different teams will have different tooling needs

Agile Database Practices

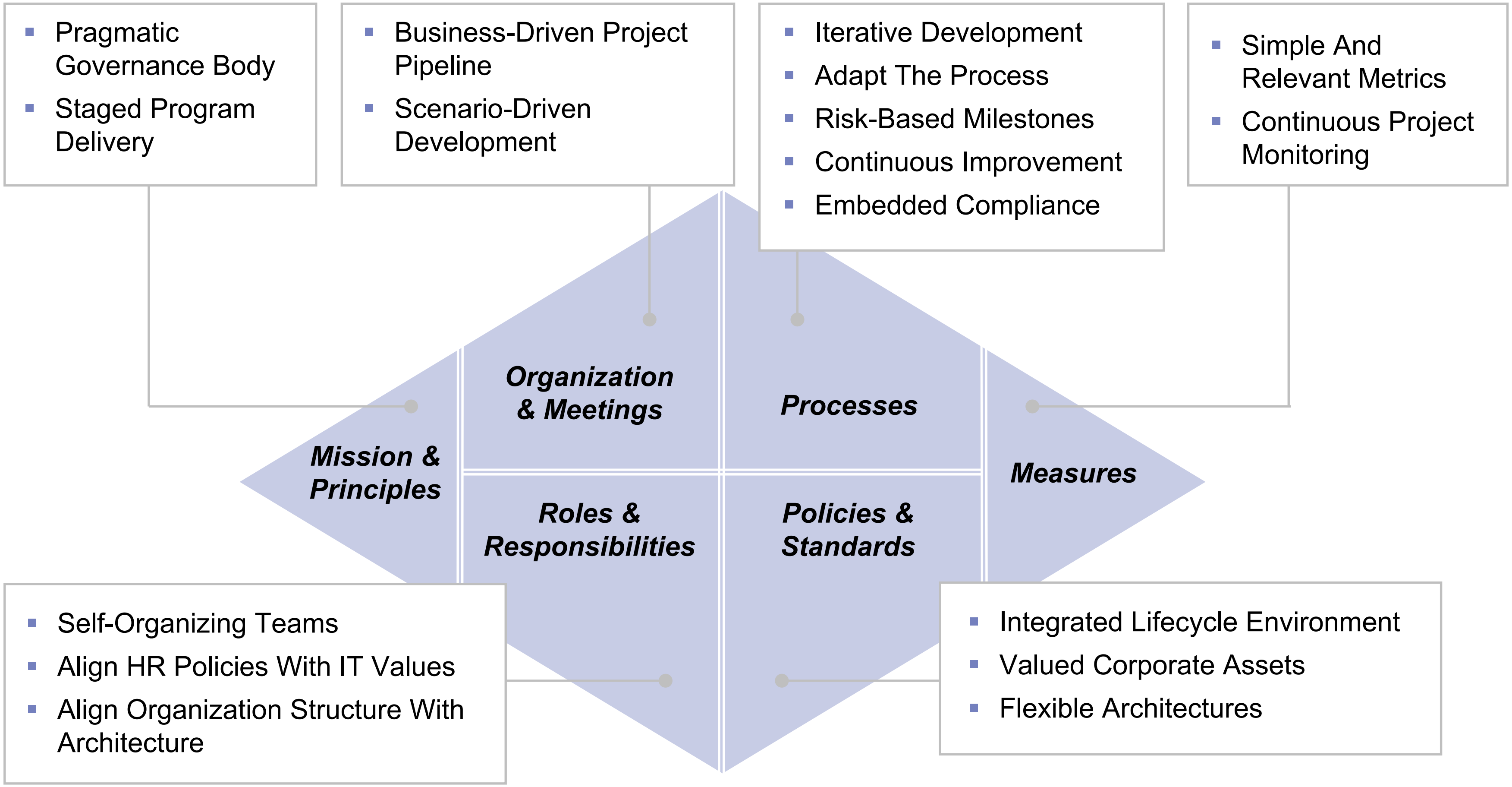
- Database Refactoring
  - www.agiledata.org/essays/databaseRefactoring.html
- Database Regression Testing
  - www.agiledata.org/essays/databaseTesting.html
- Continuous Database Integration
  - www.martinfowler.com/articles/evodb.html

Scale Agile via Rational Unified Process (RUP)

- Organizations have instantiated RUP to be very agile
- Adopt:
  - Regular delivery of working software
  - Active stakeholder participation
  - Daily stand up meetings
  - Test-Driven Development (TDD)
  - Continuous builds
  - Agile documentation techniques
- Scaling strengths:
  - Business risk addressed in Inception
  - Technical risk addressed in Elaboration



Agility at Scale: “Right-Sizing” Governance



- Pragmatic Governance Body
- Staged Program Delivery

- Business-Driven Project Pipeline
- Scenario-Driven Development

- Iterative Development
- Adapt The Process
- Risk-Based Milestones
- Continuous Improvement
- Embedded Compliance

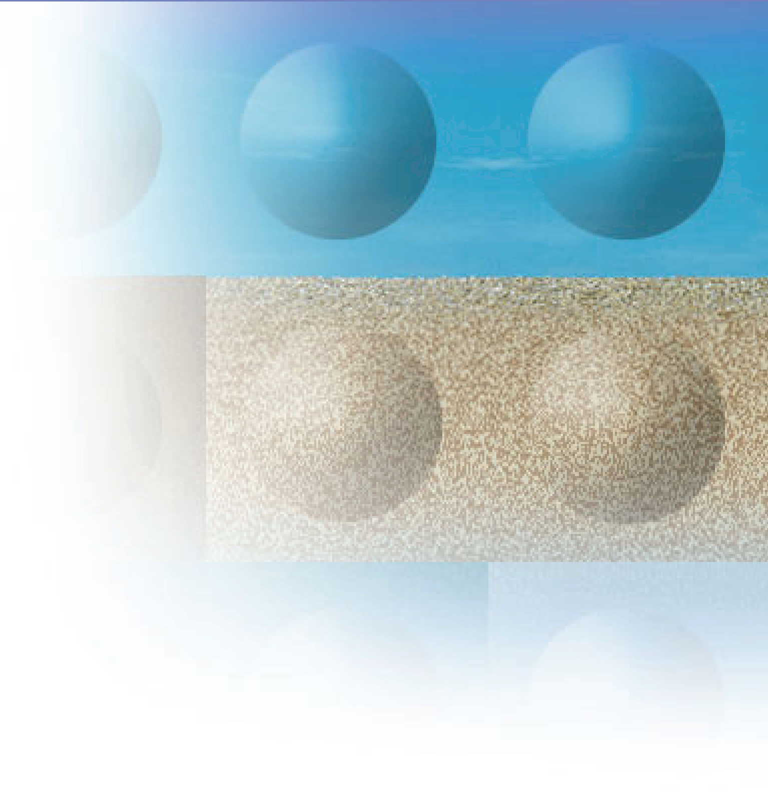
- Simple And Relevant Metrics
- Continuous Project Monitoring

- Self-Organizing Teams
- Align HR Policies With IT Values
- Align Organization Structure With Architecture

- Integrated Lifecycle Environment
- Valued Corporate Assets
- Flexible Architectures

Discussion Topics

- Warning!
- Agile Software Development
- Agile Adoption Rate
- Scaling Agile
- Parting Thoughts
  - Succeeding with Agile
  - Critical IBM Agile Resources
  - A Call to Action



Succeeding with Agile

- **Agile Development is transforming how development is done**
  - Placing greater demand on processes and tools
- **Agile can scale to accommodate technical and organizational complexity**
  - Agility at Scale has unique needs
  - IBM Rational brings a wealth of knowledge to help companies succeed in complex environments
- **IBM Rational is forging new ground in Agile Development**
  - Bringing technology, services and best practices to help customers reduce the risk of Agile projects
  - Investing significant resources to improve Agile success

Critical IBM Agile Resources

www.ibm.com/rational/agile/

www.ibm.com/developerworks/

A Call To Action

- **Consider an Agile Pilot Project**
  - See it work for yourself
  - Get mentoring help
- **Get some Agile training**
  - Project management training is critical
  - Training modelers, developers, … is also critical
- **Adopt an Agile Form of RUP**
  - RUP done right is agile
  - RUP provides the control mechanisms and risk-mitigation strategies that other agile methods lack

➔ Go to IBM

The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way.

References and Recommended Reading

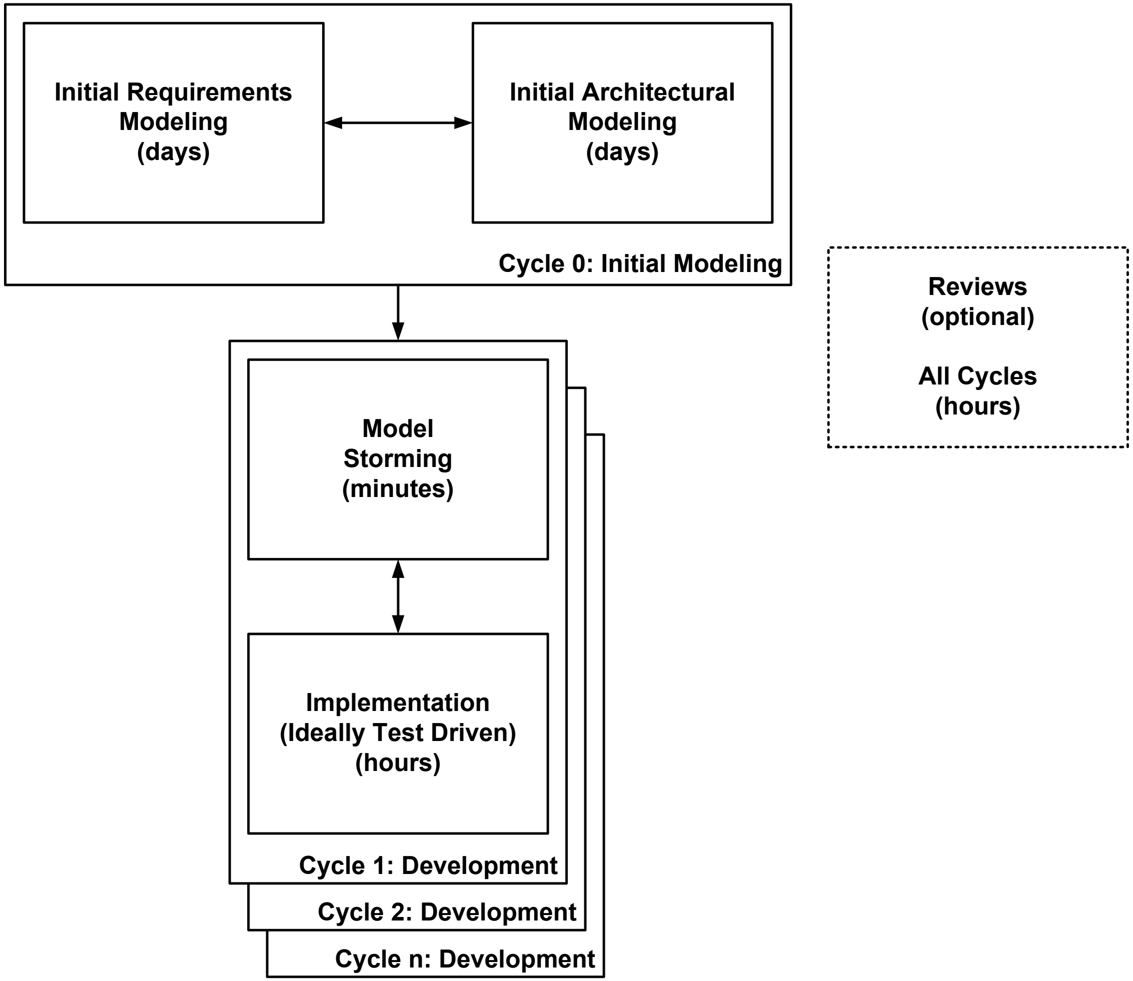
- www.agilealliance.com
- www.agilemodeling.com
- www.agiledata.org
- www.enterpriseunifiedprocess.com
- Ambler, S.W. (2002). *Agile Modeling: Effective Practices for XP and the UP*. New York: John Wiley & Sons.
- Ambler, S.W. (2003). *Agile Database Techniques*. New York: John Wiley & Sons.
- Ambler, S.W. (2004). *The Object Primer 3rd Edition: AMDD with UML 2*. New York: Cambridge University Press.
- Ambler, S.W. and Sadalage, P.J. (2006). *Refactoring Databases: Evolutionary Database Design*. Reading, MA: Addison Wesley Longman, Inc.
- Larman, C. (2004). *Agile and Iterative Development: A Manager's Guide*. Reading, MA: Addison Wesley
- McGovern, J., Ambler, S.W., Stevens, M., Linn, J., Sharan, V., & Jo, E. (2003). *The Practical Guide to Enterprise Architecture*. Prentice Hall PTR.

AMDD Project Level www.agilemodeling.com/essays/amdd.htm

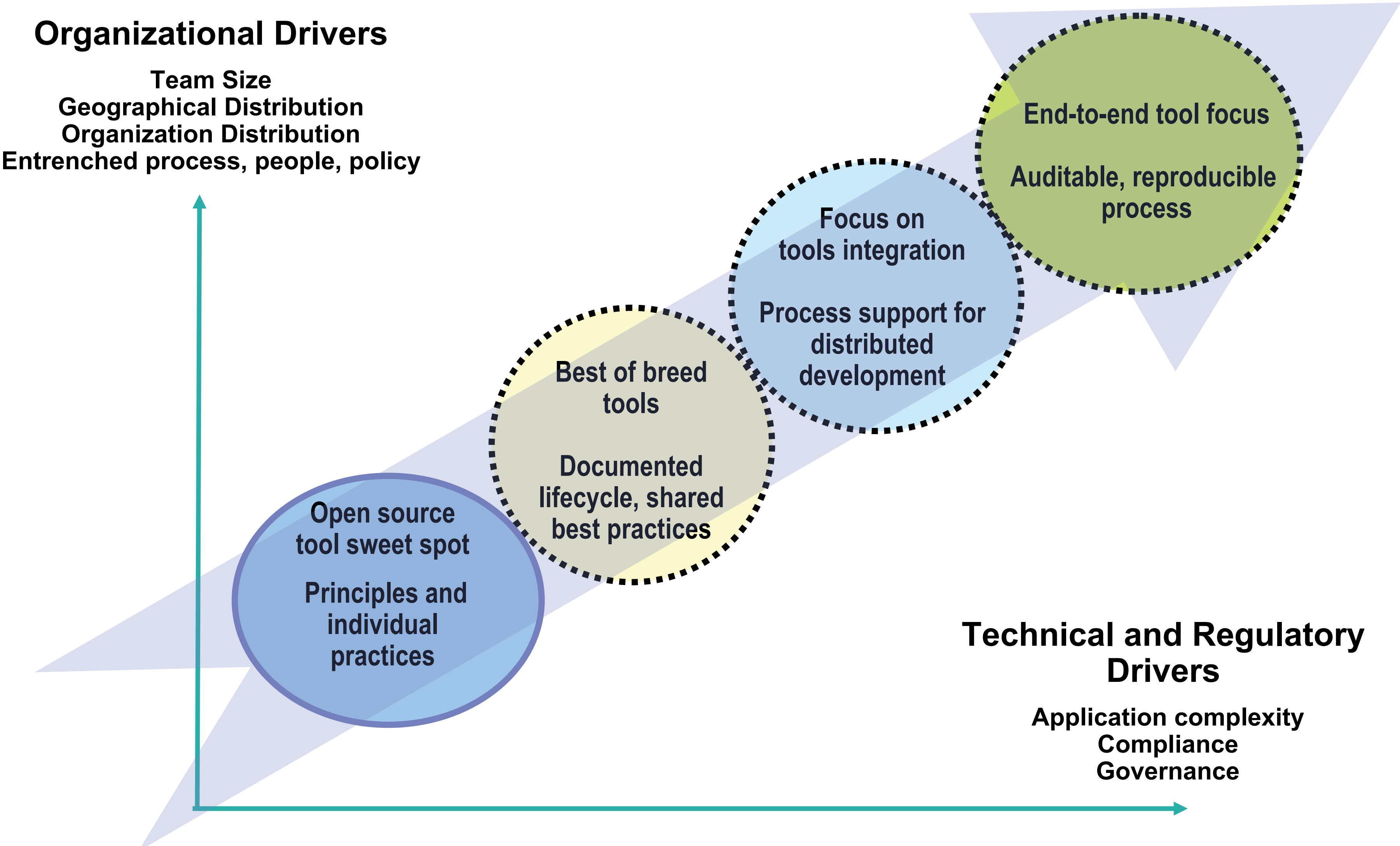
Goals: Gain an initial understanding of the scope, the business domain, and your overall approach.

Goal: Quickly explore in detail a specific issue before you implement it.

Goal: Develop working software in an evolutionary manner.


Initial Requirements Modeling (days)
Initial Architectural Modeling (days)
Cycle 0: Initial Modeling
Model Storming (minutes)
Implementation (Ideally Test Driven) (hours)
Cycle 1: Development
Cycle 2: Development
Cycle n: Development
Reviews (optional)
All Cycles (hours)

Complexity Changes the Approach for Tools & Process



Organizational Drivers

Team Size
Geographical Distribution
Organization Distribution
Entrenched process, people, policy

Open source tool sweet spot

Principles and individual practices

Best of breed tools

Documented lifecycle, shared best practices

Focus on tools integration

Process support for distributed development

End-to-end tool focus

Auditable, reproducible process

Technical and Regulatory Drivers

Application complexity
Compliance
Governance

Top 5 Most Useful Work Products on Agile Projects (Score out of 5)

Working Software (4.23)

Source Code (4.21)

Developer Tests (3.96)

Whiteboard Sketches (3.90)

Iteration Task List (3.87)

Top 5 Least Useful Work Products on Agile Projects (Score out of 5)

Detailed Gantt Chart (2.21)

High-Level Gantt Chart (2.70)

Detailed Architecture Spec (2.88)

Detailed Requirements Spec (2.90)

Detailed Use Cases (2.96)

Top 5 Most Useful Practices on Agile Projects (Score out of 5)

Iterative Development (4.38)

Regular Software Delivery (4.19)

Configuration Management (4.06)

Whiteboard Modeling (4.01)

Customer Testing (3.94)

Top 5 Least Useful Practices on Agile Projects (Score out of 5)

CASE Modeling (3.03)

Model Reviews (3.14)

Database Refactoring (3.31)

Database Testing (3.37)

Pair Programming (3.40)