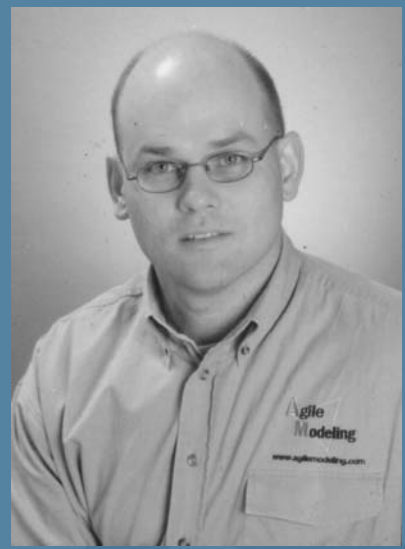
Crazy Talk: Examining Why Agile Software Development Works

Scott W. Ambler

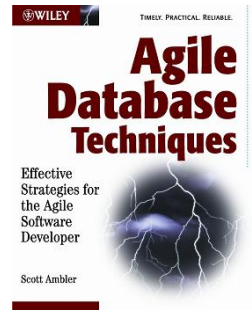
Practice Leader, Agile Modelling
www.ambysoft.com/scottAmbler.html

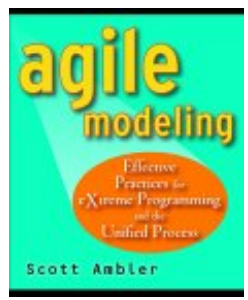


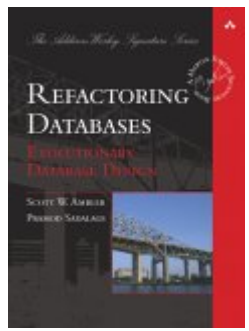
TS-3987

Scott W. Ambler

- Methodologist, Author, Consultant
- Fellow, Int. Assoc. of Software Architects
- Services:
  - Agile Model Driven Development (AMDD)
  - RUP/EUP/AgileUP mentoring
  - Agile software development coaching/mentoring
  - Training workshops
  - Management SPI workshops
  - Internal conference keynotes







Presentation Overview

- Warning!
- My Process Background
- Agile Software Development
- Adoption Rate
- Agile Techniques
- Why Agile Works
- The Eclipse Process Framework (EPF)
- Interesting Observations

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

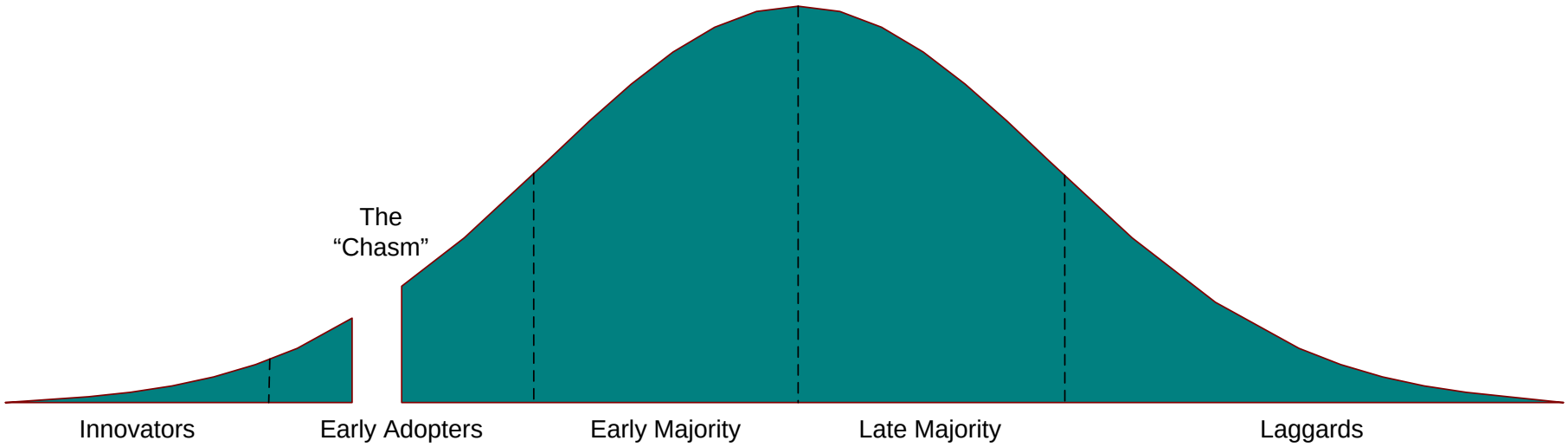
My Process Background

- Thought leader behind:
  - Pinball SDLC (1995)
  - Object-Oriented Software Process (1997–1999)
  - Enterprise Unified Process (1998+)
  - Agile Modeling (2001+)
  - Agile Data (2002+)
  - Agile Unified Process (2001+)
- Actively developed software:
  - Following processes from very agile to very traditional
  - On small to very large projects (~ $100 million a year)
  - On short to very long projects (multi-year)

Agile Software Development

- Agile software development is an approach to software development that is:
  1. People-oriented
  2. That enables teams to respond effectively to change
  3. Results in the creation of working systems that meets the changing needs of its stakeholders

Adoption Rate of Agile Techniques



Common Agile Practices

- Regular deployment of working software
- Pair programming
- Active stakeholder participation
- Model with others
- Sandboxes
- Test First Design (TFD)
- Test Driven Design (TDD)
- Continuous regression testing
- Tests as primary artifacts
- Continuous integration
- Follow guidance
- Scrum
- Agile Model Driven Development (AMDD)
- Agile requirements management

Regular Deployment of Working Software

- How many projects have you seen that:
  - Were “90% complete” for months?
  - Delivered wonderful plans but no software?
  - Delivered wonderful models, but no software?
- The only accurate measure of software development is the delivery of software
  - Deliver something at the end of each cycle/iteration
  - Iterations should be short
  - At all points in time stakeholders can see what they’ve gotten for their investment to date

Pair Programming

- Two programmers work side-by-side, collaborating on the same design, algorithm, code, or test
- The driver has control of the keyboard/mouse and actively implements the program
- The observer continuously observes the work of the driver to identify tactical (syntactic, spelling, etc.) defects and also thinks strategically about the direction of the work
- They periodically switch roles, working together as equals
- On demand, the two programmers can brainstorm any challenging problem
- Significant evidence exists which shows that pair programming is more effective, overall, than solo programming for the vast majority of developers
- pairprogramming.com

Active Stakeholder Participation

- Project stakeholders should:
  - Provide information in a timely manner
  - Make decisions in a timely manner
  - Actively participate in business-oriented modeling
- www.agilemodeling.com/essays/activeStakeholderParticipation.htm
- www.agilemodeling.com/essays/inclusiveModels.htm

Model with Others

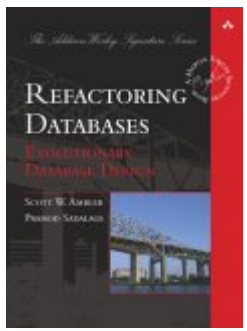
- The modeling equivalent of pair programming
- You are fundamentally at risk whenever someone works on something by themselves

Refactoring

- A code refactoring is a small change to your code to improve your design that retains the behavioral semantics of your code; examples: Rename Method, Move Method, and Remove Setting Method
- Refactoring enables you to evolve your development assets in a controlled manner, enabling your design to remain high quality
- www.refactoring.com

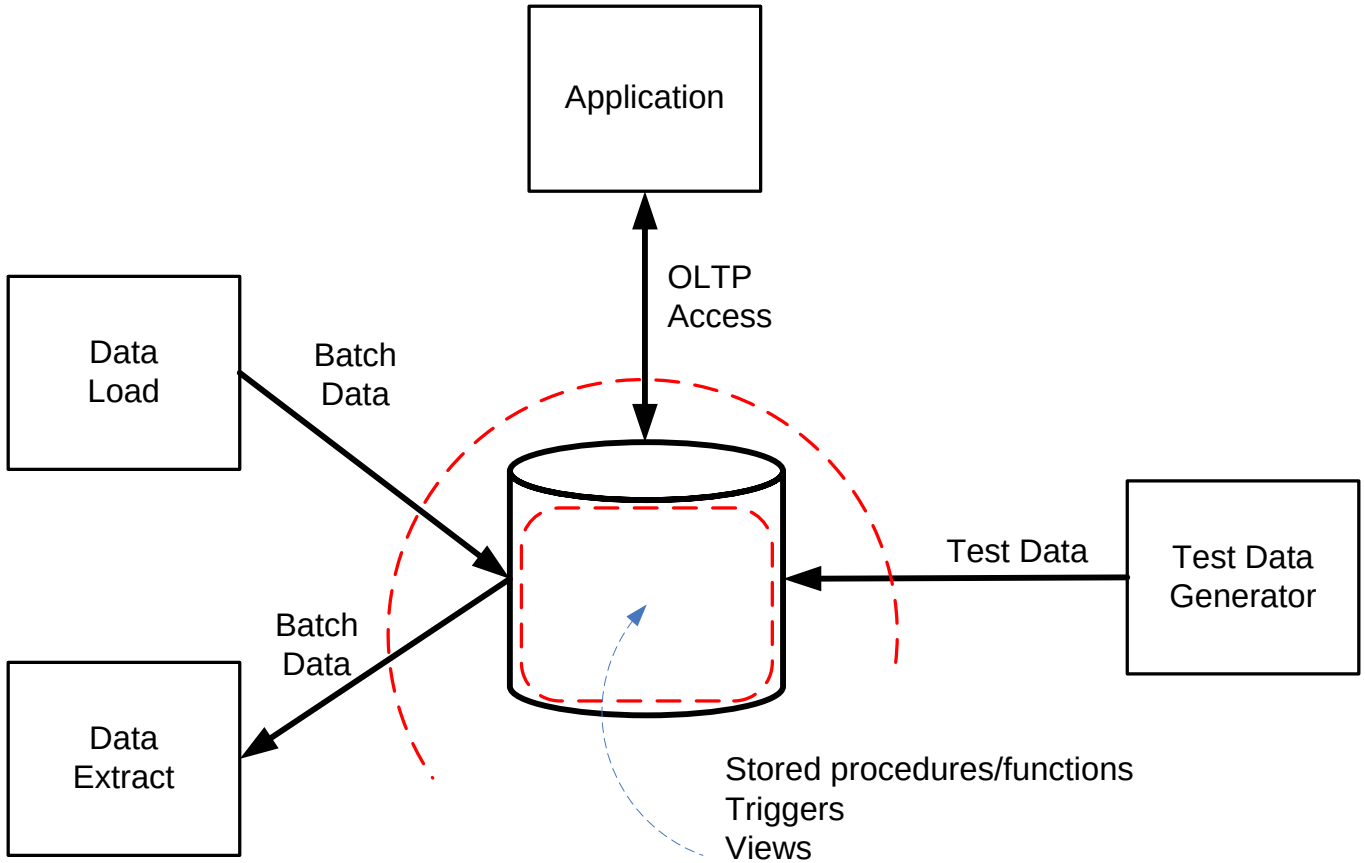
Database Refactoring

- A database refactoring is a simple change to a database schema that improves its design while retaining both its behavioral and informational semantics; examples: Move Column, Rename Table, and Replace Blob With Table
- A database schema includes both structural aspects such as table and view definitions as well as functional aspects such as stored procedures and triggers
- Important: Database refactorings are a subset of schema transformations, but they do not add functionality
- www.agiledata.org/essays/databaseRefactoring.html
- www.databaserefactoring.com



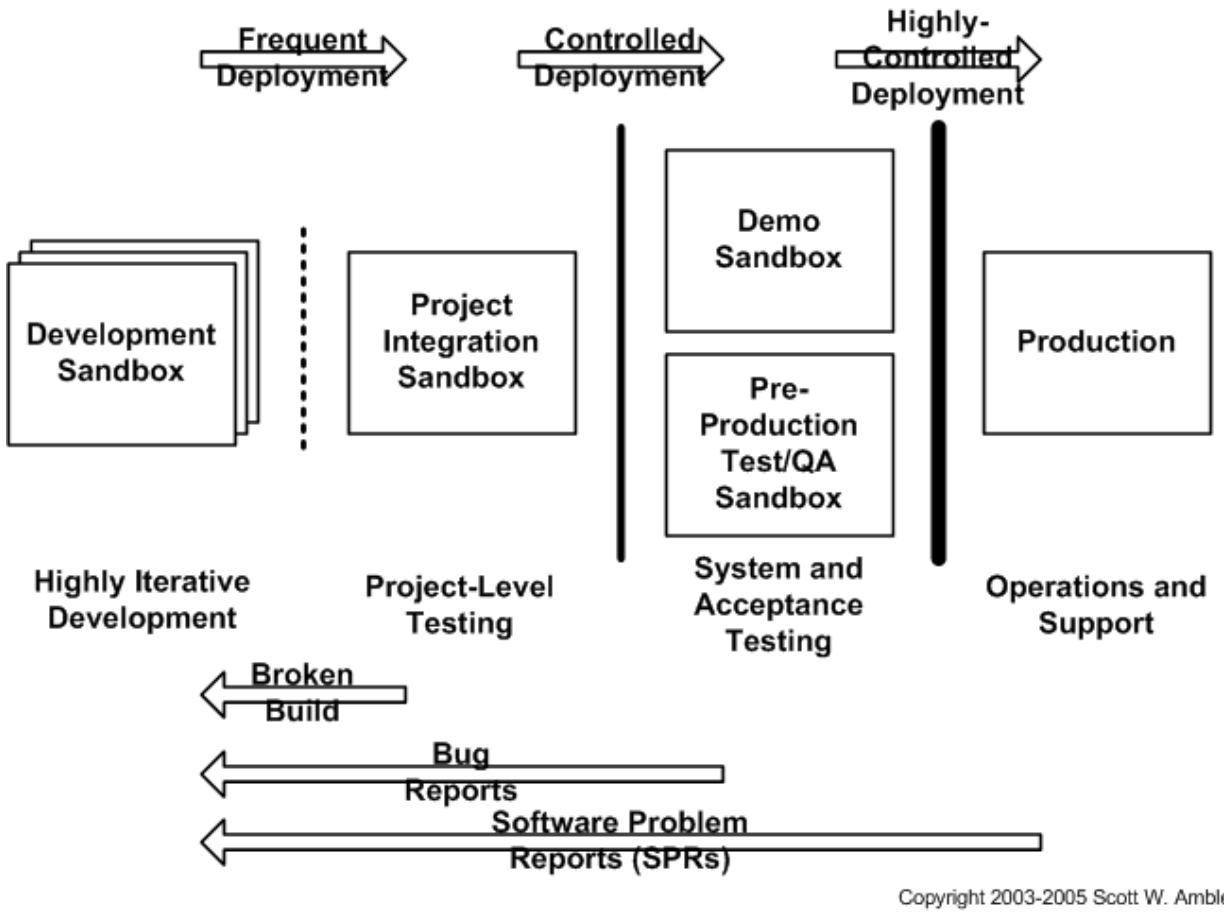
Database Testing

www.agiledata.org/essays/databaseTesting.html



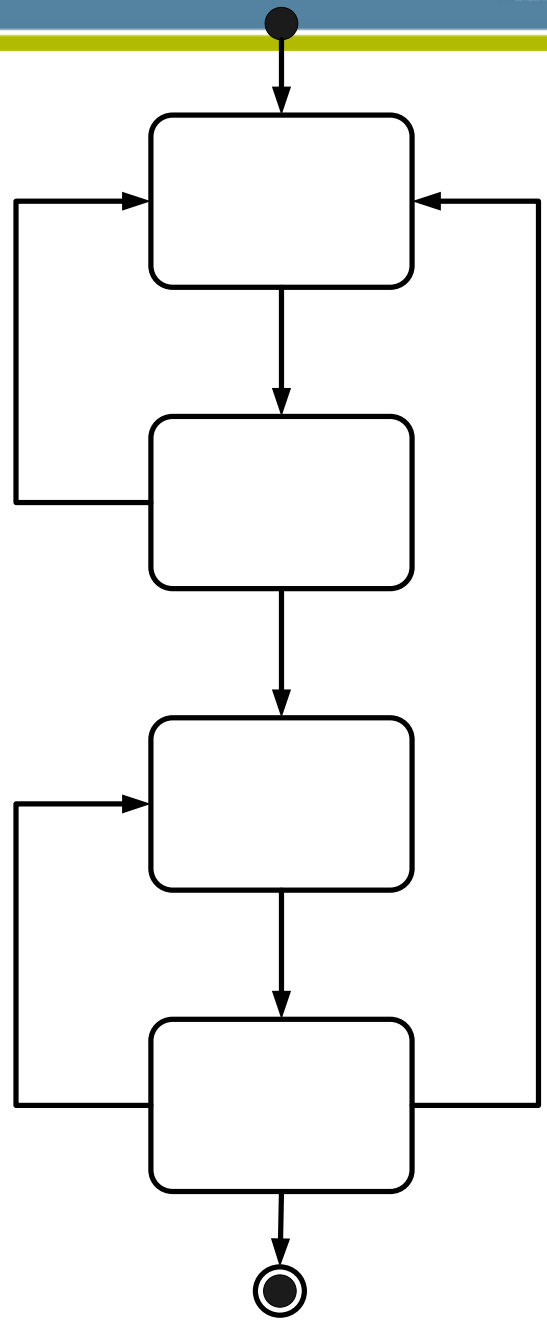
Sandboxes

www.agiledata.org/essays/sandboxes.html



Test First Design (TFD)

www.agiledata.org/essays/tdd.html



Test Driven Design (TDD)

- TDD = Refactoring + TFD
- Steps:
  1. Before implementing a requirement, ask yourself if the existing design is the simplest one to implement the requirement
  2. If not, refactor the code before continuing
  3. Take a TFD approach to implement the requirement

Continuous Regression Testing

- Regression testing is critical to the success of evolutionary (iterative and incremental) development
- It's such a good idea, agilists prefer to do it all the time

Tests as Primary Artifacts

- Acceptance tests are considered to be primary requirements artifacts
- Unit tests are considered to be detailed design artifacts

Continuous Integration

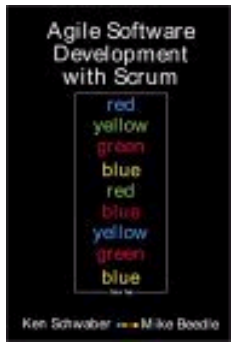
- Daily builds are a good start
- We update and test our code constantly
- Therefore we need to build the system constantly
- Some tools:
  - Cruise Control
  - Maven/Continuum
  - Ant/AntHill
  - Tinderbox
- damagecontrol.codehaus.org/Continuous+Integration+Server+Feature+Matrix?print=1

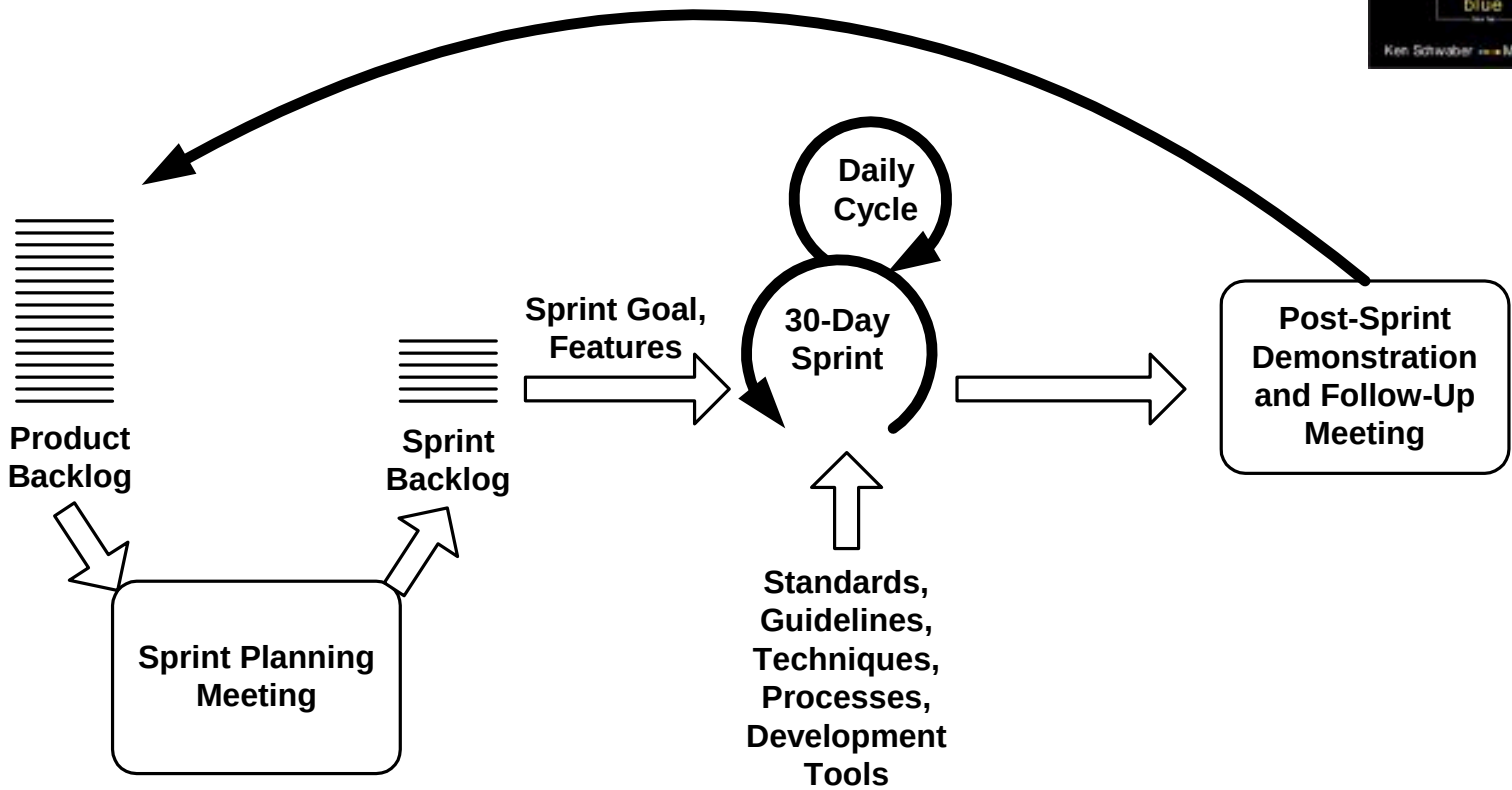
Follow Guidance

- Guidance = Standards and guidelines
- Agile developers prefer to develop high-quality artifacts, and that includes ensuring that they are developed in a consistent manner
- XP practice Coding Standards
- AM practice Apply Modeling Standards
- www.agilemodeling.com/style/

Scrum

www.controlchaos.com

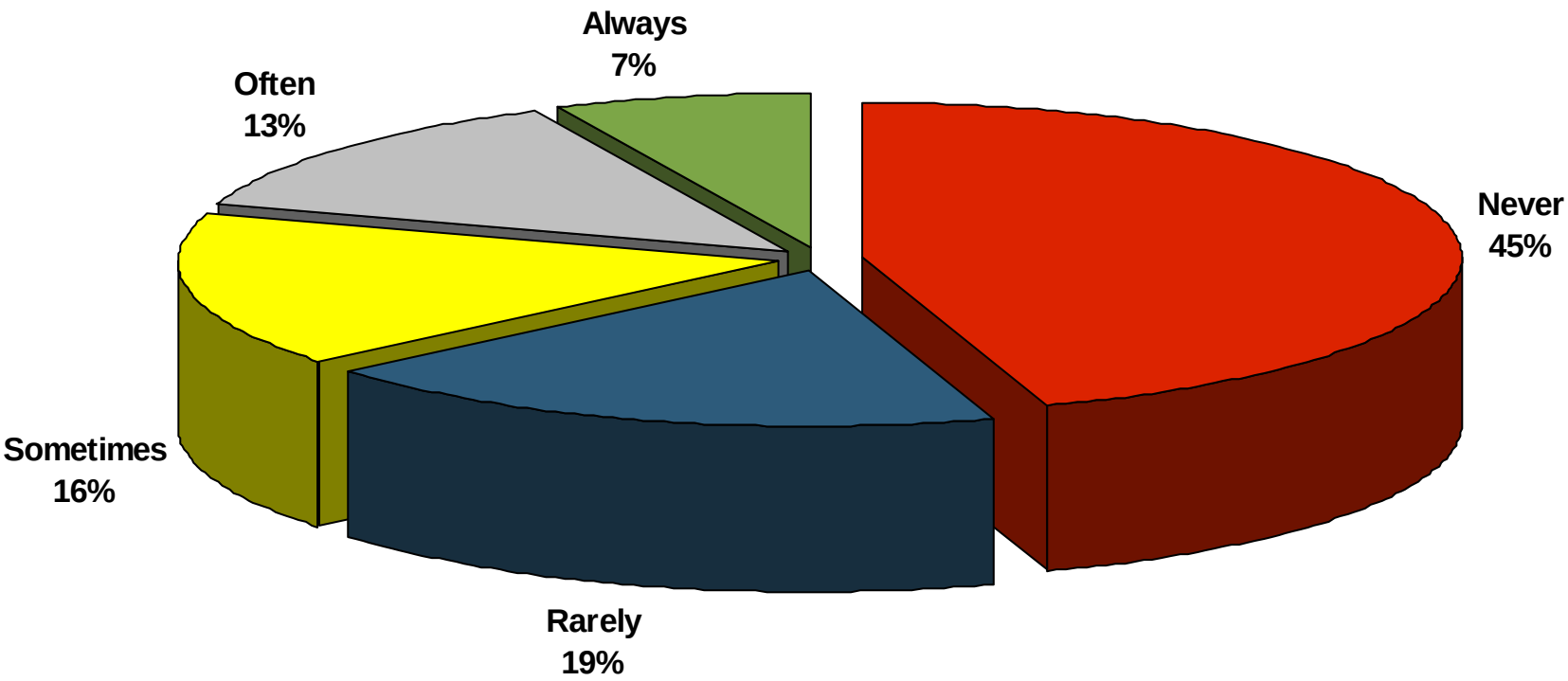




The Cost of BRUF:

Feature Usage Within Deployed Applications

www.agilemodeling.com/essays/examiningBRUF.html



Source: Jim Johnson of the Standish Group, Keynote Speech XP 2002

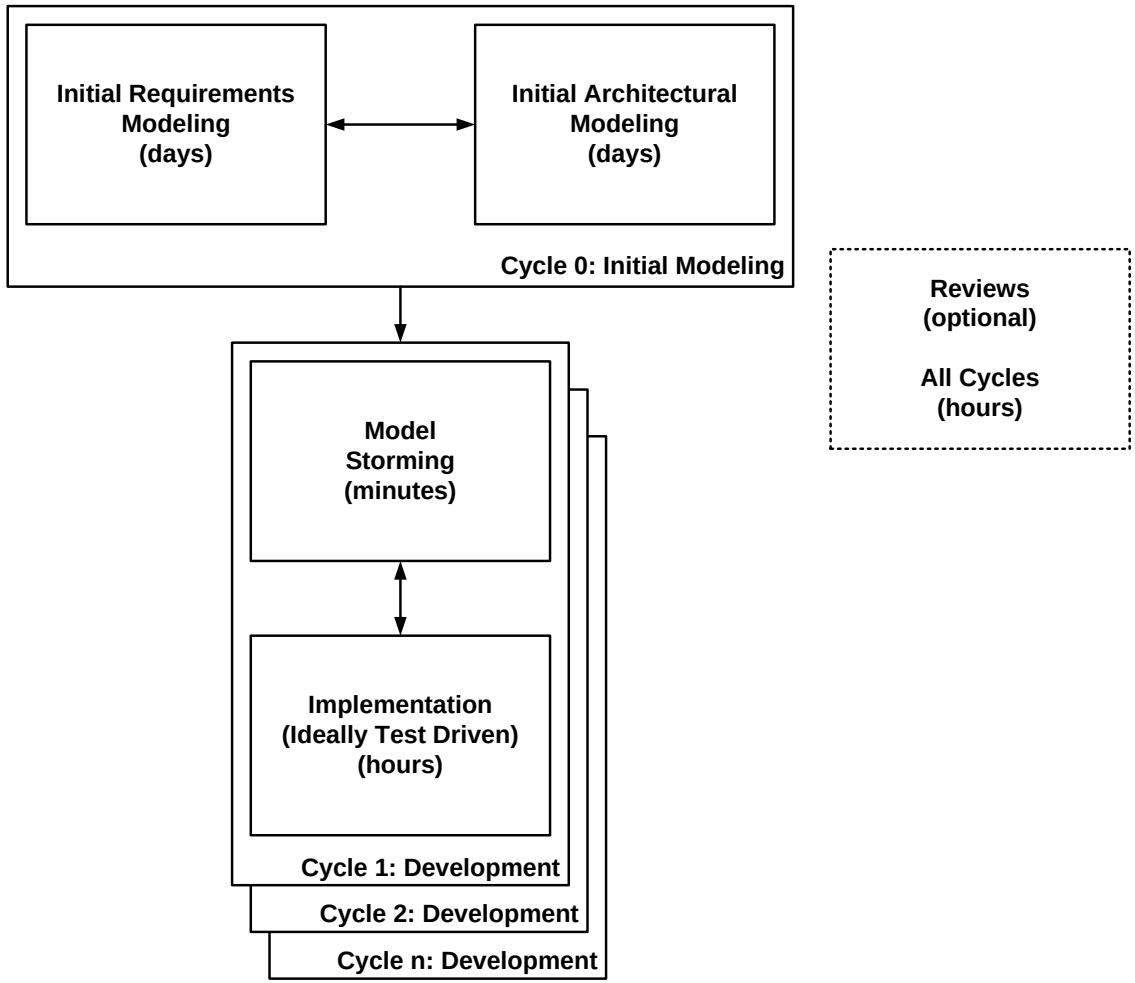
Agile Model Driven Development (AMDD)

www.agilemodeling.com/essays/amdd.htm

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

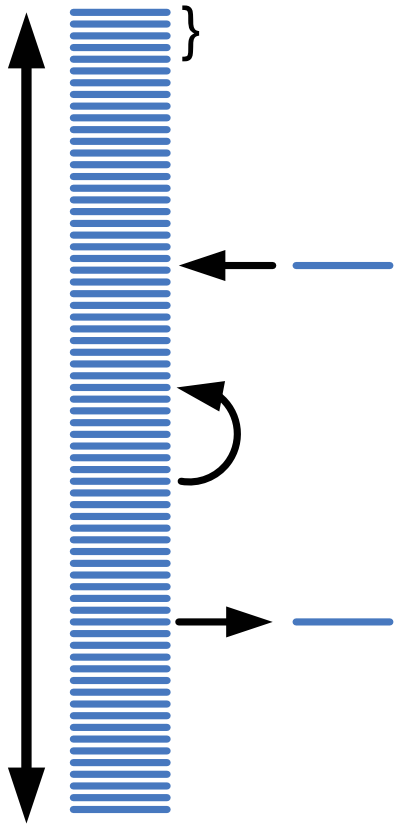
Goal: Quickly explore in detail a specific issue before you implement it.

Goal: Develop working software in an evolutionary manner.



Agile Change Management

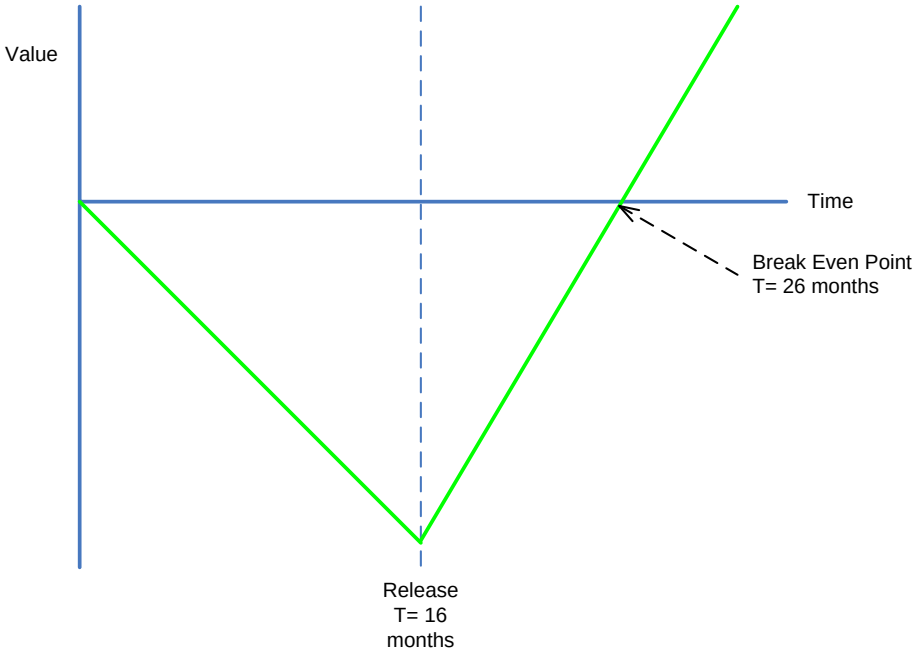
www.agilemodeling.com/essays/changeManagement.htm

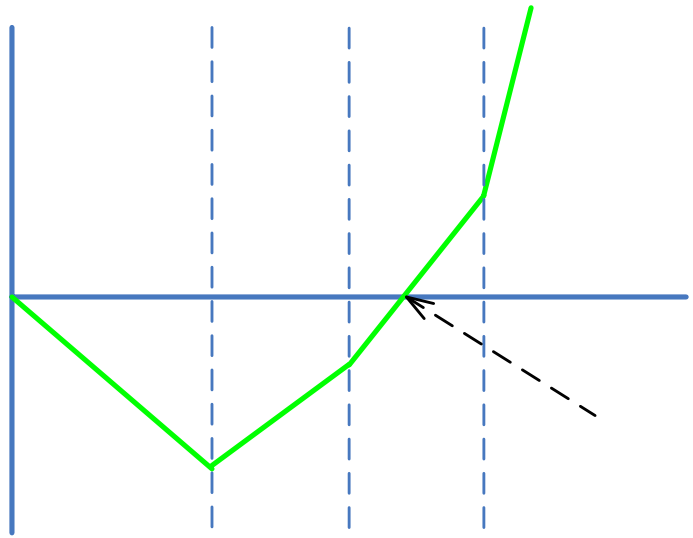


Why Agile Software Development Works

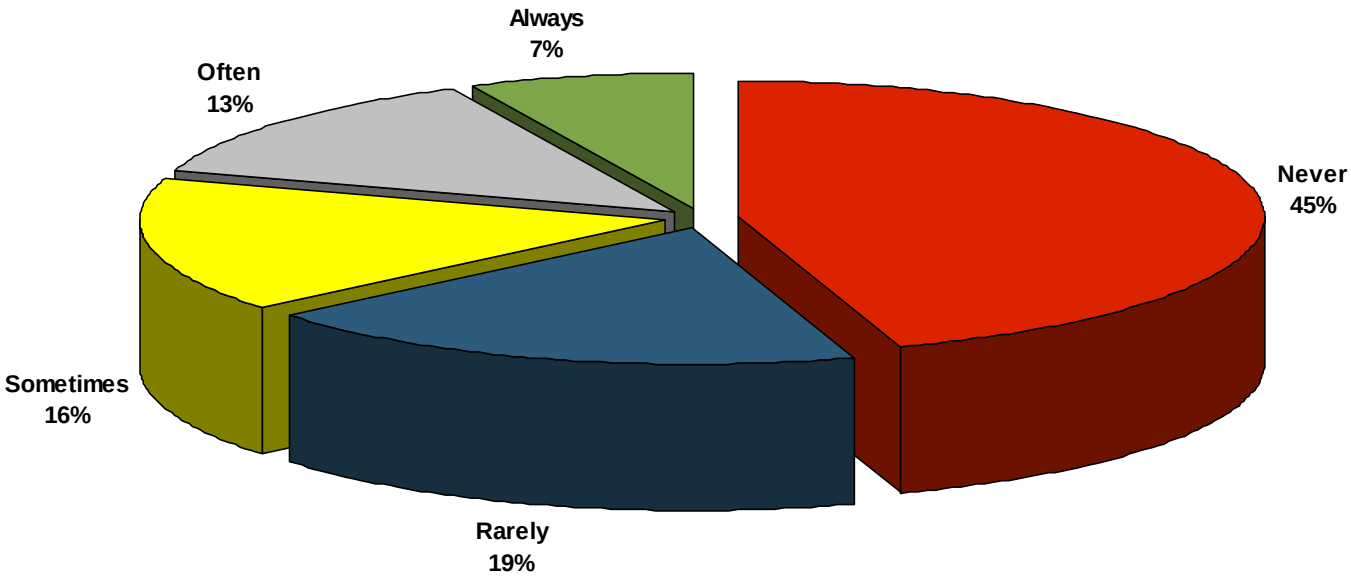
Comparing the Cost Benefit Curves
BRUF vs. Agile Change Mgmt
Cost of Change Curves

Comparing the Cost-Benefit Curves

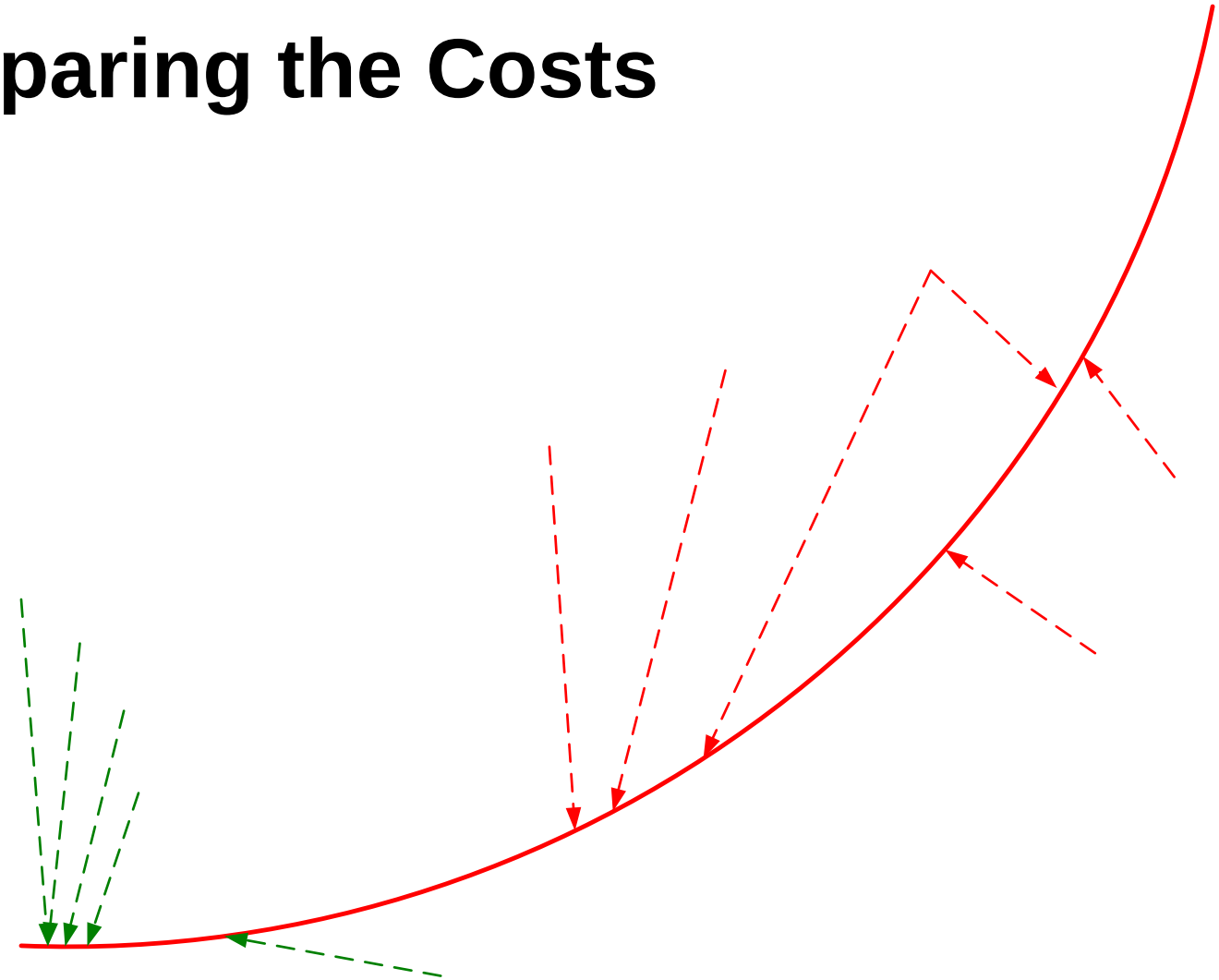




BRUF vs. Agile Change Mgmt



Comparing the Costs



The Eclipse Process Framework (EPF)

- EPF is a knowledge management system for project-centric development processes
  - Potential for several base processes (e.g. Open UP, XP, …)
  - Many process plug ins (e.g., DB Refactoring, Scrum, …)
- Support a variety of different processes and process models
  - Agile, yet still compatible to senior management
  - Endorsed by many organizations
- The tool and process material is open source
  - If you don't like it, change it
- www.eclipse.org/epf
- www.openup.org (coming soon)

Interesting Observations

- You need to become a generalizing specialist:
  - www.agilemodeling.com/essays/generalizingSpecialist.htm
- Agile software development is real and not a fad
- Agile software development is supported by a wide range of industry luminaries
- Research evidence support agile techniques is beginning to emerge
- Significant evidence exists showing that traditional techniques suffer from significant challenges
- Why is that the people saying agile doesn't work rarely seem to have tried it or even read a book about it?

Q&A

Scott W. Ambler

www.ambysoft.com/scottAmbler.html

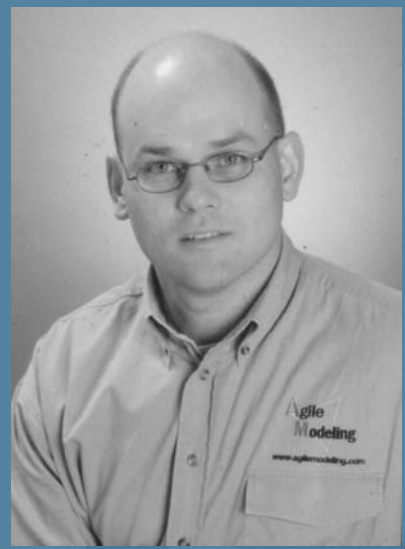
References and Recommended Reading

- www.agilealliance.com
- www.agilemodeling.com
- www.agiledata.org
- www.ambysoft.com
- www.databaserefactoring.com
- 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.

Crazy Talk: Examining Why Agile Software Development Works

Scott W. Ambler

Practice Leader, Agile Modelling
www.ambysoft.com/scottAmbler.html



TS-3987