The Agile Unified Process (AUP)

Scott W. Ambler
Senior Consultant, Ambysoft Inc.
www.ambysoft.com/scottAmbler.html
Scott W. Ambler

- Methodologist, Author, Consultant
- Services:
  - Agile Model Driven Development (AMDD)
  - RUP/EUP/AgileUP mentoring
  - Agile Software Development Coaching/Mentoring
  - Training Workshops
  - Management SPI Workshops
  - Internal Conference Keynotes
Overview

- Warning!
- The Unified Process
- Agile Software Development
- The AUP Disciplines
- Secrets of Success
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
Observation: It’s the Same Cost Curve (JIT)
Rational Unified Process (RUP)
Agile UP

www.ambysoft.com/unifiedprocess/agileUP.html
## Agile UP Phases and Milestones

**Inception**
- Define project scope
- Estimate cost and schedule
- Define risks
- Develop business case
- Prepare project environment

**Elaboration**
- Specify requirements in greater detail
- Identify architecture
- Validate architecture
- Evolve project environment
- Staff project team

**Construction**
- Model, build, and test system
- Develop supporting documentation

**Transition**
- System testing
- User testing
- System rework
- System deployment

**Lifecycle Objectives (LCO)**
- Scope concurrence
- Initial requirements definition
- Plan concurrence
- Risk acceptance
- Process acceptance
- Business case
- Project plan

**Lifecycle Architecture (LCA)**
- Vision stability
- Requirements stability
- Architecture stability
- Risk acceptance
- Cost and estimate acceptance
- Realistic chance to succeed
- Project plan

**Initial Operating Capacity (IOC)**
- System stability
- Requirements stability
- Prepared stakeholders
- Risk acceptance
- Cost and estimate acceptance
- Project plan

**Product Release (PR)**
- Business acceptance
- Operations acceptance
- Support acceptance
- Cost and estimate acceptance

Copyright 2001-2005 Scott W. Ambler
The Disciplines of the AUP

- Modeling
- Implementation
- Test
- Deployment
- Configuration Management
- Project Management
- Environment
The Modeling Discipline
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](http://www.agilemodeling.com/essays/activeStakeholderParticipation.htm)
Agile Model Driven Development (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.

---

Copyright 2001-2005 Scott W. Ambler
Agile Software Requirements Management

Changing Requirements Are a Competitive Advantage if You Can Act on Them: [link](http://www.agilemodeling.com/essays/agileRequirements.htm)
There is More to Modeling than UML
The Agile Data (AD) method is a collection of philosophies that will enable IT professionals within your organization to work together effectively when it comes to the data aspects of software-based systems.

Six philosophies:

- **Data.** Data is one of several important aspects of software-based systems.
- **Enterprise issues.** Development teams must consider and act appropriately regarding enterprise issues.
- **Enterprise Groups.** Enterprise groups exist to nurture enterprise assets and to support other groups, such as development teams, within your organization.
- **Unique situation.** Each development project is unique, requiring a flexible approach tailored to its needs. One software process does not fit all.
- **Work together.** IT professionals must work together effectively, actively striving to overcome the challenges that make it difficult to do so.
- **Sweet spot.** Avoid the black and white extremes to find the gray that works best for your overall situation.
The Implementation Discipline
Pair Programming

- Two programmers working 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
Refactoring

A refactoring is a small change to your code to improve your design that retains the behavioral semantics of your code.

Two types:

- Code refactoring
- Database refactoring

www.refactoring.com
www.databaserefactoring.com
Continuous Integration

- Daily builds are a good start
- We update and test our code constantly
- Therefore we need to build the system constantly
A database refactoring is a simple change to a database schema that improves its design while retaining both its behavioral and informational semantics.

A database schema includes both structural aspects such as table and view definitions as well as functional aspects such as stored procedures and triggers.

Database refactorings are a subset of schema transformations, but they do not add functionality.
Test Driven Design (TDD)

www.agiledata.org/essays/tdd.html
Full Lifecycle Object-Oriented Testing (FLOOT)


Requirements Testing
- Model reviews
- Prototype walkthroughs
- Prove it with code
- Usage scenario testing

Analysis Testing
- Model reviews
- Prototype walkthroughs
- Prove it with code
- Usage scenario testing

Architecture/Design Testing
- Model reviews
- Model walkthroughs
- Prototype walkthroughs
- Prove it with code

Code Testing
- Black-box testing
- Boundary value testing
- Class-integration testing
- Class testing
- Code reviews
- Coverage testing
- Inheritance-regression testing
- Method testing
- Path testing
- White-box testing

System Testing
- Function testing
- Installation testing
- Operations testing
- Stress testing
- Support testing

User Testing
- Alpha testing
- Beta testing
- Pilot testing
- User acceptance testing (UAT)

Copyright 2001-2005 Scott W. Ambler
The Deployment Discipline
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
Deployment Strategy

Development Sandbox → Frequent Deployment → Project Integration Sandbox → Controlled Deployment → Demo Sandbox → Highly Controlled Deployment → Test/QA Sandbox → System and Acceptance Testing → Operations and Support → Production

Highly Iterative Development → Project-Level Testing → System and Acceptance Testing → Operations and Support
The Configuration Management Discipline

Plan

Setup Configuration Environment

Configuration Environment

Project Overview

Configuration Manager

Create Workspace

Check Out Artifact

Check In Artifact

Resolve Check-In Conflicts

Create Baseline

Manage Configurations

Artifact

Anyone

Copyright 2005 Scott W. Ambler
The Environment Discipline
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 *Modeling Standards*
- [www.agilemodeling.com/style/](http://www.agilemodeling.com/style/)
Why Agile UP?

Pair Programming
Model With Others
Test Driven Design (TDD)
Active Stakeholder Participation

Full Lifecycle Testing
Code Inspections
Model and Document Reviews

Big Requirements Up Front (BRUF)
Big Design Up Front (BDUF)

Agile Model Driven Development (AMDD)
Secrets of Success

- Focus on collaborative approaches, not processes and tools
- Recognize that people:
  - Won’t read detailed process descriptions
  - Want templates and examples
- Keep it simple
- www.ambysoft.com/unifiedprocess/agileUP.html
Keep in Touch

Scott W. Ambler

www.ambysoft.com/scottAmbler.html