Agile Planning, Tracking and Project Management Boot Camp

XP Agile Universe Conference

Calgary, Alberta, Canada

August 15, 2004

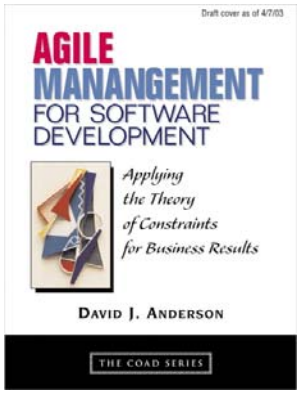
Agile Logic
Agility That Delivers.

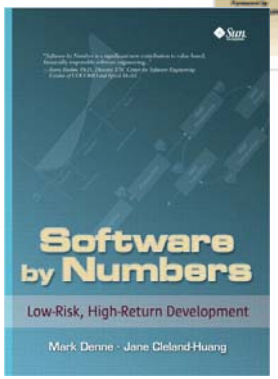
XP Agile Universe
August 15-18, 2004
Calgary, Alberta, Canada
www.xpuniverse.com

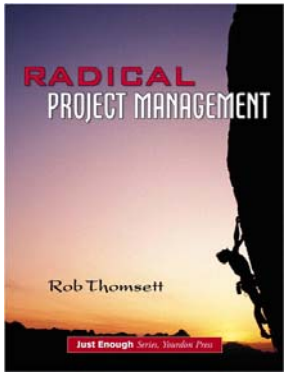
Your Instructor

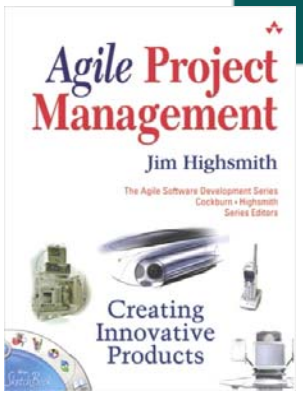
- Paul Hodgetts
  - Founder and CEO of Agile Logic
  - Team coach, trainer, consultant, developer
  - 21 years overall, 4½ years agile experience
  - Contributing author (Extreme Programming Perspectives)
  - Presenter at conferences (ADC, XPAU, JavaOne)
  - Agile Alliance Program Director
  - Member of CSUF agile advisory board
  - Contact info: www.agilelogic.com phodgetts@agilelogic.com

Why Is Agile Project Management Hot?









Agile Logic
Agility That Delivers.

XP Agile Universe
August 15-18, 2004
Calgary, Alberta, Canada
www.xpuniverse.com

This Tutorial

- Looks at agile development from a project manager's perspective
- Information useful to understanding how to do project management for an agile project
- Exercises to help gain a feel for some of the concepts and practices

The Plan

- Agile Project Management Concepts
- The Agile Project Community
- Three Views of an Agile Project
- "Managing" the Project

Agile Project Management Concepts

- What Is Agility?
- What Benefits Are We Trying to Gain from an Agile Approach?
- What Makes Agile Project Management Different?
- The Agile Process Landscape

What Is an "Agile" Process?

- According to the Merriam-Webster on-line dictionary "agile" means:
  - "1: marked by ready ability to move with quick easy grace;"
  - "2: having a quick resourceful and adaptable character."
- In agile software development, "agile" tends to mean "the ability to respond to change."

Change in Projects

- Changes in Requirements and Priorities
- Changes from Technology and Tools
- Changes from People
- Changes from the Inherent Complexity of Software

Changes in Requirements and Priorities

- Stakeholders learn from the solution
  - Learn what their true needs are
  - Learn how to better communicate their needs
- Business environment and conditions change
- Business processes are re-engineered

Changes from Technology and Tools

- We are often learning new things on the fly
- Actual capabilities may vary from expectations
- Combinations create compatibility issues
- New versions are released

Changes from People

- Team composition changes over time
- Team interactions are complex
- Individual behavior can be unpredictable

Changes from the Inherent Complexity of Software

- Network of dependencies is very large
- Solutions need recursive feedback and validation
- Difficult to predict activities and dependencies

Agile Project Management Concepts

- What Is Agility?
- What Benefits Are We Trying to Gain from an Agile Approach?
- What Makes Agile Project Management Different?
- The Agile Process Landscape

What Are We Trying to Gain With an Agile Process?

- Respond to change and leverage learning
- Deliver the highest business value (ROI)
- Decrease time-to-delivery
- Increase productivity and efficiency
- Produce better quality solutions
- Create a more fulfilling development culture

Agile Project Management Concepts

- What Is Agility?
- What Benefits Are We Trying to Gain from an Agile Approach?
- What Makes Agile Project Management Different?
- The Agile Process Landscape

What's Really Different About Managing an Agile Process?

- Iterative and incremental
- Parallel and concurrent, not phased
- Planned around deliverables, not activities
- Dynamic project balancing via scope adjustments
- Heavy emphasis on collaboration
- Management by facilitation

Iterative and Incremental

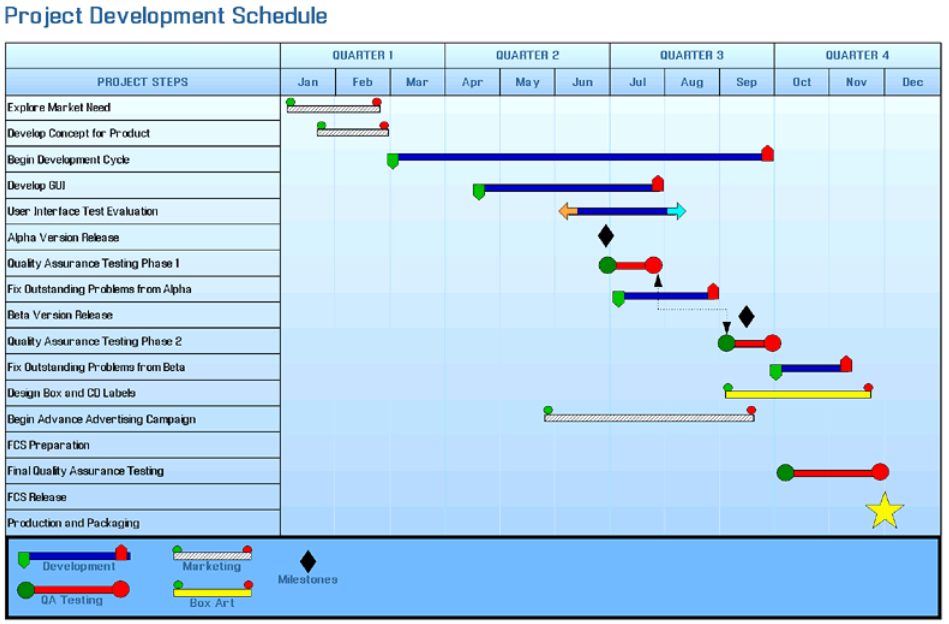
- Iterative
  - Repeatedly executing nested process cycles
  - Iterations provide synchronizing points
  - Iterations provide feedback points
- Incremental
  - System is built in progressive stages
  - Iterations add features and refinements
  - Each increment is a working system

Phased vs. Concurrent Activities

- Phased Approach
  - Gathers similar activity types together
  - Preference towards serial completion
  - Ultimate in phased approach is waterfall
- Concurrent and Parallel
  - Activities occur opportunistically
  - Activities of all types happening at same time
  - Partial completion considered the norm

“Predictive” Planning

- Creation of comprehensive activity-based plans
- Execution of defined activities to follow plan
- Management by controlling activities to conform to plan

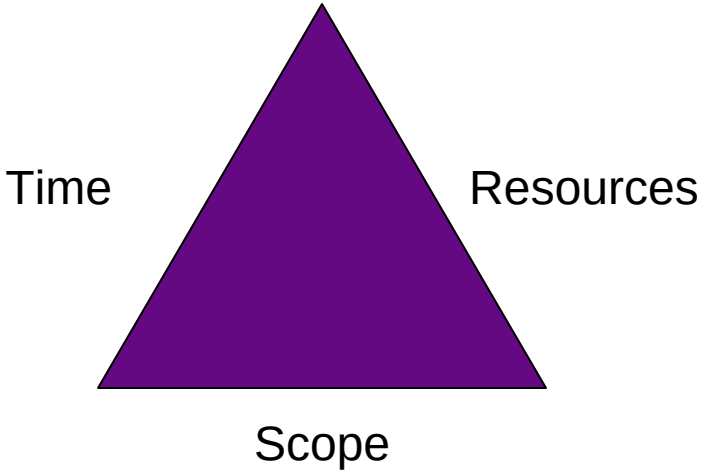


"Agile" Planning

- Creation of prioritized set of deliverables
- Opportunistic execution of activities to create deliverables
- Management via feedback and adaptation

The "Classic Trio" of Software Development

- Resources
- Time
- Scope
- Must be in balance for a healthy project



The Resource Variable

- **Staffing is usually the least effective variable to adjust.**
  - Staffing increases have long lead times.
  - Increased intensity has diminishing returns.
  - Team culture requires some degree of stability.
- **Tools and technology can provide benefits.**
  - Effective tools provide continuing benefits.
  - Front-end costs need to be carefully amortized.
  - The wrong tools and technology increase friction.

The Time Variable

- Can be the most painful variable to adjust
  - Early commitments are usually date-based.
  - Target dates are often the most important objective.
  - Within a date boundary, there's only so much time.

The Scope Variable

- Can be the most effective variable to adjust
  - Can adjust scope breadth – what's included.
  - Can adjust scope depth – refinement.
  - Partial scope can often generate immediate returns.
  - It is often preferable to reach a date with partial scope completely finished, rather than complete scope partially finished.

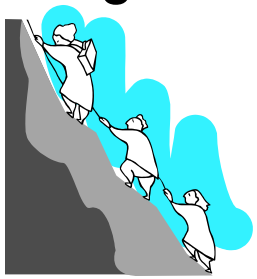
Project Balance in an Agile Process

- Sustainable resource management
  - Stable teams
  - Steady pace
  - Favor high ROI tools and technology
- Fixed time management
  - Time-boxed development cycles
- Adaptive scope management
  - Feedback-based scope adjustments

"Heroic" vs. "Collaborative"

- Heroic development emphasizes individuals
  - Activities assigned to individuals
  - Project results heavily dependent on individual performance
  - Increases "keyhole" risks

- Collaborative development emphasizes teams
  - Teams self-organize activities to meet goals
  - Teams leverage diverse skills
  - Teams mitigate keyhole risks



Management by Facilitation

- Command and Control Strategy
  - Decisions made by central authorities
  - Activities delegated
  - Manager controls activities
- Facilitation and Empowerment Strategy
  - Decisions made by those with the most info
  - Activities accepted
  - Team self-manages and adapts
  - Organization ensures supportive environment

Agile Project Management Concepts

- What Is Agility?
- What Benefits Are We Trying to Gain from an Agile Approach?
- What Makes Agile Project Management Different?
- ➲ The Agile Process Landscape

The World of Agile Processes

- Extreme Programming (XP)
- Feature-Driven Development (FDD)
- DSDM (Dynamic System Development Method)
- Scrum
- Crystal Family of Processes, e.g. Crystal Clear
- Lean Software Development
- Adaptive Software Development (ASD)
- Others: Agile UP/RUP, Evo, Win-Win Spiral

The Agile Alliance

- 2001 – representatives from agile processes meet in Snowbird, Utah.
- Agreed on a "manifesto" of values and principles:
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan
- "That is, while there value in the items on the right, we value the items on the left more."

The Plan

- Agile Project Management Concepts
- The Agile Project Community
- Three Views of an Agile Project
- "Managing" the Project

The Project Community

- The "Business" or "Customer" or "Product Owner" role
- The "Developer" role
- The "Manager" role
- Emphasis on a "Whole Team" approach
  - While each role represents certain interests and is perhaps assigned accountability for certain aspects of the project, the team as a whole maintains interest in and responsibility for the overall project.

Business / Customer Role

- Brings to the table:
  - Understanding of the business and product needs
  - Specific definitions of features (requirements, scope)
  - Priorities
  - The ability to accept the product
- Speaks as a single voice to team
- Is likely to be multiple stakeholder types
- Is likely to have a multiplicity of stakeholders
- Stakeholders may be represented by proxies

Business / Customer Role

- Potential members:
  - Product Managers
  - Marketing, Sales
  - Business Analysts
  - Quality Assurance (acceptance testing)
  - End Users, Their Managers
  - Business/System Operations
  - Others when acting as a stakeholder

Developer Role

- Brings to the table:
  - Ability to create and communicate solutions
  - Cost estimations and explaining trade-offs
  - Delivering usable functionality that meets requirements and priorities
- Ideal is a group of wide-ranging generalists
- Likely to consist of specialists to some degree
- "Generalizing specialists" preferred

Developer Role

- Potential members:
  - Programmers
  - Architects and Designers
  - Technical Leads
  - Interface Architects/UI Designers
  - Database Designers and DBAs
  - Operations and Network Designers

Manager Role

- Brings to the table:
  - Defines overall organizational goals
  - Interfaces with organizational entities (status)
  - Environmental support (facilities, equipment)
  - Cultural support (organizational values)
  - Personnel administration (reviews, hiring, etc.)
  - Business administration (budgets, etc.)

Manager Role

- Potential members:
  - Owners, Shareholders
  - Board of Directors
  - Executive Management
  - Project Management
  - Technical Management, Administrative Management
  - Process (Quality and Process Engineering)

The Plan

- Agile Project Management Concepts
- The Agile Project Community
- Three Views of an Agile Project
- "Managing" the Project

Three Views of an Agile Project

- Work Products
- Cycles
- Timeline of Events

The Work Product Structure of Agile Development

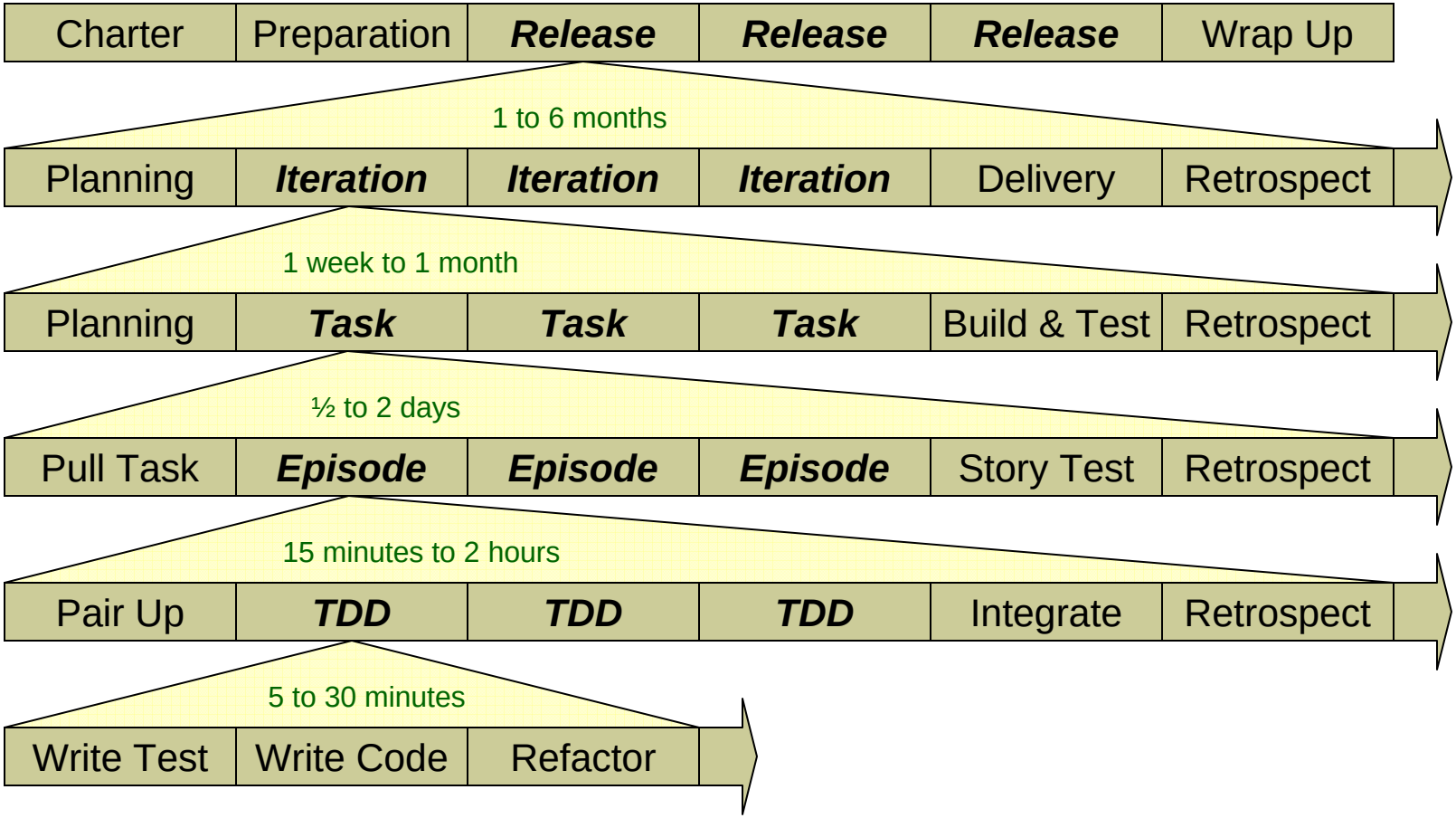
- Incremental Building Blocks
  - Business Objectives
  - Projects and Products
  - Feature Sets
  - Sagas and Stories
  - Technical Work Units (Tasks)
  - Technical Integrations
- Traceability through the Deliverables

The Cycles of Agile Development

- Project Cycle
- Release Cycle
- Iteration Cycle
- Task Cycle
- Episode Cycle

- Forward-Driving Activities
- Resolution Increases
- Feedback through the Cycles

The Cycles of Agile Development



Agile Logic
Agility That Delivers.

XP Agile Universe
August 15-18, 2004
Calgary, Alberta, Canada
www.xpuniverse.com

The Project Cycle

- Chartering
- Preparation
- Release Delivery…

Chartering the Project

- Elements of a Charter
  - Defines the overall mission
  - Defines specific, testable goals – "business stories"
  - Defines schedule constraints
  - Defines available resources and limits
  - Defines the project community, roles and accountability

Developing Product Strategies

- Not a formulaic operation – requires some leg work
- Based on product development best practices
  - Market research
  - Competitive analysis
  - Customer feedback

Exercise

- Congo.com is a new on-line retailer of technical books. They need to establish themselves against the other river.
- Write a charter for Congo's new project.
  - What strategies could they use to break into the market?
  - How can we express those as business stories?
  - Constraints, resources, team at your discretion.

Developing Feature Sets

- Canonical XP is incremental and just-in-time
  - Need a car… A convertible… A red one… With under 15,000 miles…
  - We tend to ask for the highest values parts first
- But some forces favor more up-front work
  - Highest value nuggets not obvious
  - Larger plans required for funding
  - Contracts specify "fixed" scope

Exercise

- Given Congo's charter and strategy, develop a list of candidate feature sets and features that would implement the charter and strategy.

Preparing Requirements

- Use best practices – use cases
- But don't over-do it
  - Cockburn's lighter-weight format
  - Constantine's essential use cases
- Overall use cases useful for generating a shared mental model
- Fill in detail as incrementally as possible

Use Cases vs. User Stories

- A story is not a use case, a use case is not a story
- A use case defines functional requirements in total
- Defines breadth and depth of system behavior
- Additional non-functional requirements often needed
- A story defines a piece of system capability to be implemented
- Stories as change requests – add, modify or remove capability

Generating Stories from Use Cases

- The overall set of use cases is the quilt
- The stories are the patches that are incrementally sewn together to fill it in
- Story scope – what use case(s) are being asked for?
- Story breadth – what use case scenarios are being asked for?
- Story depth – what level of completion is being asked for?

Example of Uses Cases and Stories

- Overall set of use cases for shopping cart system:
  - Customer Views Catalog
  - Customer Adds Book to Shopping Cart
  - Customer Checks Out Order
  - Warehouse Clerk Enters New Shipment Received
  - Customer Service Rep Checks Shipping Status

Example of Uses Cases and Stories

- Single Use Case in More Detail:
  - Customer Checks Out Order, Card Accepted, Card Type X
    - Step 1
    - Step 2…
    - Calculate Discount
    - Calculate Tax
    - Authorize Payment
    - More Steps…
  - Customer Checks Out Order, Card Denied, Card Type X
  - Customer Checks Out Order, Card Accepted, Card Type Y

Example of Uses Cases and Stories

- Example of a story that maps to use case:
  - "Implement the Customer Checkout use case. Handle only the card accepted, card type X scenario. Don't worry about sales tax or discount calculations at this point."

Example of Uses Cases and Stories

- Example of a cross-cutting story
  - "Implement sales tax calculations. Sales tax calculations are needed in these use cases: Review Shopping Cart, User Checkout, Admin Review Order. Handle only a single fixed sales tax rate at this point – don't worry about state tax tables."

Why Would We Go Though All This?

- Good stories are the essential building blocks of release plans
- Increase the value of the software
  - Scope the highest priority stories to the core business value
    - "Minimal Marketable Feature" – MMF
  - Early delivery of MMFs
    - Accelerates break-even on the project
    - Increases the Net Present Value (NPV) of the project

The Release Cycle

- Release Planning
- Delivering Iterations…
- Release Delivery
- Release Retrospective

Release Planning

- Presenting Stories
- Estimating Stories
- Estimating Velocity
- Story Selection and Prioritization
- Create the Plan

Presenting Stories

- Stories are discussed and understood (analysis)
- Developers determine overall technical approach (architecture and design)

Estimating Stories

- For a release plan, goal is to relatively quickly sort the stories into groupings of coarser-grained sizes.
- We're not trying to calculate effort to any degree of precision.

Story Estimate Sizes

- Story estimates are probabilities
- Choose estimate "bucket" sizes, e.g. 1, 2, 3, 5, 8
- Small stories implement the "small batch size" principal
- Smaller stories allow more of the team capacity to be used
- Smaller stories have less chance of catastrophic overruns
- Suggest limiting max size to one-half an iteration

Estimating Stories using Relative Estimates

- Choose an "average" sized story
- Assign a mid-range value to it
- Compare each remaining story against it
- Assign a relative value to each
- Can break team away from being too precise

Estimating Stories using Ideal Days

- Developers determine uninterrupted time to complete
- Can lead to issues due to perceived precision
- Whose ideal day is it?
- If team systemically over- or under-estimates, ideal day estimates can turn into relative estimates
- Can give the team a reference point

Estimating Issues

- Estimates require:
  - Sufficient level of definition
  - Ability to understand technical issues
- Encourage team to talk through the issues
- Not enough level of definition
  - Table the story
  - Have the customer bring it back when ready
- Insufficient understanding
  - Table the story
  - Spin off an investigative task to address (spike)

Estimating Velocity

- When using relative estimates, just have to guess on the first one and then employ feedback
- When using ideal days, take number of developers X iteration length X an overhead factor (usually .4 to .8)
- Optionally, take number of work streams (pairs) X iteration length X a lower overhead factor

Create the Plan

- Story selection and prioritization
  - Highest value stories favored
  - Release objectives and themes considered
- Choose a target release date
  - May be contingent on a minimal scope delivery
- Arrange stories into probable iterations
  - Based on prioritization

Sample (and Simple) Release Plan

Story	Priority	Estimate
User browses product category list	M	2.0
User selects & browses product category	M	2.0
User views product details	M	1.0
User adds product to shopping cart	M	1.0
User views shopping cart	M	1.0
User starts check out, reviews order	M	1.0
User enters shipping info	M	1.0
User pays by credit card - simple case	M	2.0
User enters separate billing address	S	1.0
System validates payment with service	M	3.0
System save user profile for later order	S	2.0
User enters CV number	S	1.0
System displays order summary	S	1.0
User searches for product key words	S	3.0

Release Re-estimation

- Customer refines the product strategy
- Business needs may change
- Stories added or modified
- Team learns about non-systemic over- or under-estimation, e.g. on all rules engine work
- Try to schedule on an iteration boundary

The Iteration Cycle

- Iteration Planning
- Delivering Tasks…
- Iteration Delivery
- Iteration Retrospective

Iteration Planning

- Estimating velocity
- Presenting Stories
- Task Breakdowns
- Task Estimating, or Not
- Task Sign-Ups, or Not

Estimating velocity

- The first iteration uses the release planning velocity
- Subsequent iterations use a "yesterday's" weather velocity
  - Could be exactly the same as last iteration
  - Could be a moving average (3 or 5)
  - Could be modified based on known factors (vacations, etc.)
- Always make sure the team can commit to the velocity

Presenting Stories

- Stories selected may deviate from original release plan
  - Actual progress may be different
  - May need to consider specialized resources
- Stories are discussed in greater detail (analysis)
- Developers determine mid-level technical approach (design)

Task Breakdowns

- Developers create a task list for each story
- Tasks are technical tasks
- Vertical slices preferred over horizontal slices
- Sometimes technologies and specialties drive tasks

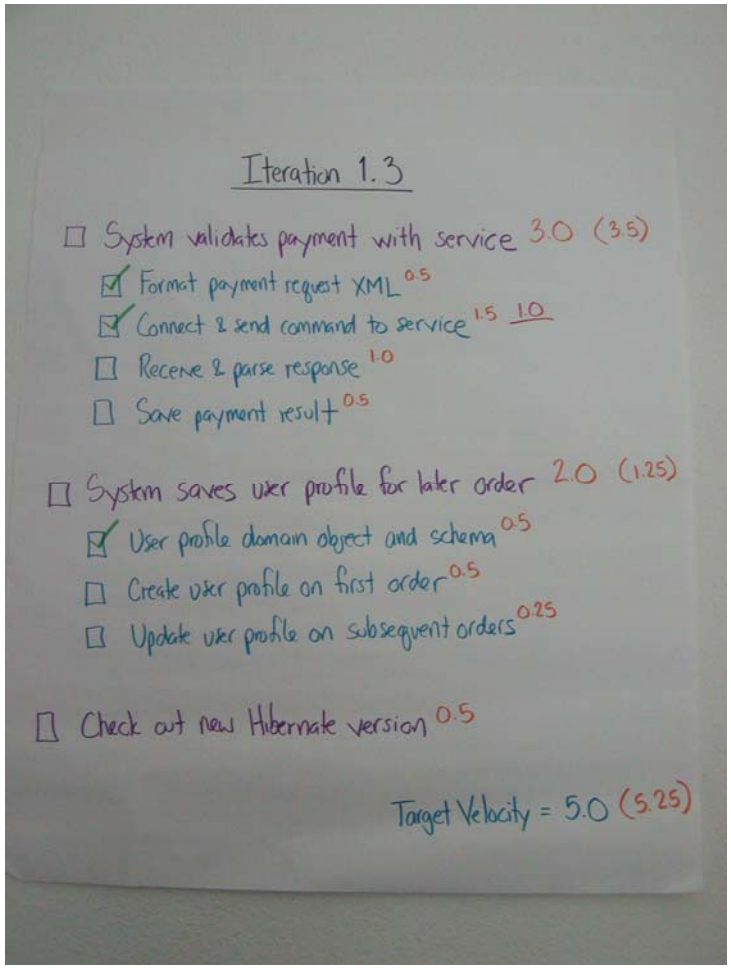
Task Estimating, or Not

- Some teams also estimate tasks in ideal hours
- Can provide feedback on accuracy of story estimates

Task Sign-Ups, or Not

- Some teams sign up for tasks at planning time
  - Maintains developer and estimate relationship
  - Go around the team, each developer signs up for a task and places their estimate on it
- Others pull tasks on an as-needed basis
  - Allows resources to more opportunistically work on tasks
  - Estimates can be collective or at pull time

Sample Iteration Plan



Iteration Recovery

- Iteration stability buffers the team against too much change
- Short iterations enable the buffer to remain intact
- Sometimes the set of stories for the iteration must change
- Gather the team for a short planning discussion
- If possible, adjust the iteration plan and continue
- If necessary, abort the iteration and start a new one

The Task Cycle

- Pulling a Task
  - A free developer chooses next available task
  - Consideration for priorities
  - Task-level dependencies likely
- Delivering Episodes…
- Task Delivery
  - Tested and integrated
- Story Delivery
  - Story passes customer acceptance tests
- Task Retrospective

The Episode Cycle

- Getting Ready
  - Pair Programming
  - Verify understanding (analysis)
  - Determine detailed technical approach (design)
- Fine-Grained Development
  - Test-Driven Development
- Integration
  - Changes must pass tests
  - Changes added to the shared artifacts
- Episode Retrospective

Exercise

- The provided table of stories represents the raw materials for release and iteration plans.
- Consider only priority and size. How would you arrange them to produce the "best" overall plan?
- Now consider feature sets. Any changes?
- Now consider dependencies. What changed?
- Now consider specializations. What changed?

The Timeline of Agile Development

- Release Events
- Iteration Events
- Releases and Iterations synchronized to time
- Daily Events
- Tasks and Episodes not synchronized to time

Release Events

- Project chartered and approved
- Stories ready for release planning
- Ready to start iterations
- All acceptance tests pass for the release
- Release deployed to end users

Iteration Events

- Time box begins
- Ready to start task development
- System is built and new stories pass acceptance tests
- Iteration time box ends

Daily Events

- Basic Stand-Up Protocol – Status, Plans, Needs
- Dynamic Daily Task Planning
- Tips for Running Effective Stand-Up Meetings

The Plan

- Agile Project Management Concepts
- The Agile Project Community
- Three Views of an Agile Project
- "Managing" the Project

"Managing" the Project

- Project management shifts to:
  - Gathering information
  - Facilitating communication
  - Pointing things out
- Feedback and Metrics
- Tracking and Reporting
- Diagnosing

Feedback and Metrics

- Objective feedback
  - Data gathered
  - Current state of artifacts
- Subjective feedback
  - Retrospective comments
  - What's not said

Some Essential Metrics

- Amount of work for the release/iteration
  - Measured in count of story estimates
- Amount of work completed
  - Measured in count of story estimates
  - Make sure the story is really complete
- Velocity
  - Measure in story points per iteration

Other Useful Metrics

- Estimate accuracy
  - Comparison of actuals to estimates
  - Correlations to feature type, technology, etc.
- Actual cost of features
  - Developer time to produce story points
- Quality
  - Defects found at various testing points
- Progress towards business objectives
  - Portion of feature sets and strategies completed

Other Useful Metrics

- Metrics to encourage specific practices
  - Measuring number of unit tests over time
- Code size and quality
  - Measuring number of things over time
  - Measuring complexity/dependencies over time
- Value stream for features sets and features
  - How long before value is realized?
  - What happens to it along the way?

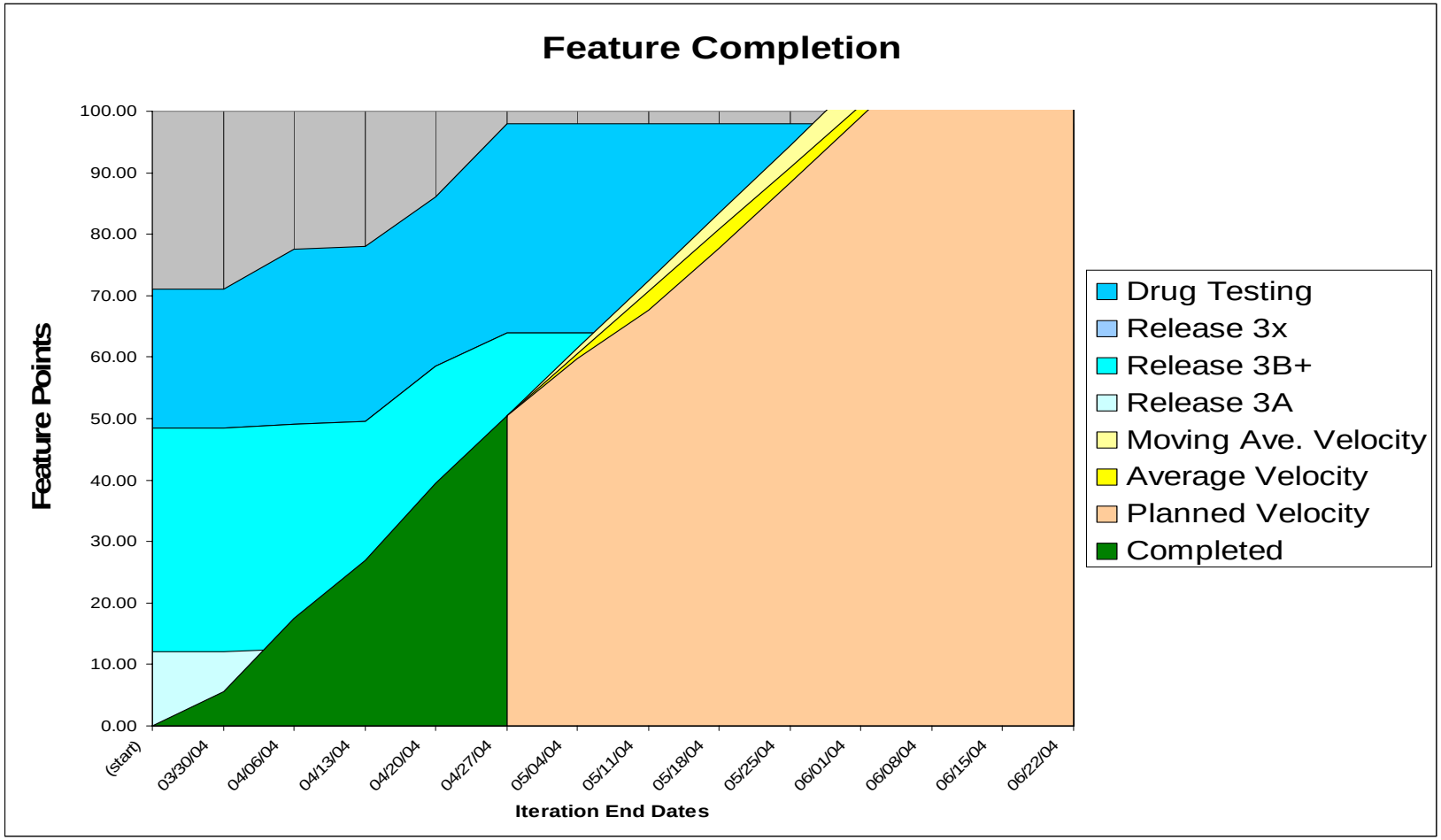
Tracking

- Leverage the process flow for tracking points
  - Completion of cycles, key events
- Collect data in small increments
  - Avoid recreating history
- Use lightweight mechanisms
  - Collect raw data
  - Offload compiling data from the team
- Use retrospectives for subjective information

Reporting

- Leverage the natural project artifacts for reporting
  - Plans and tracking data
  - Reformat for outside consumption if needed
- Keep reports simple and direct
  - Visual reporting – charts and graphs
- Make reporting accessible and unavoidable
  - Big Visible Charts
  - Project Dashboard

Sample Burn-Up Chart



Diagnosing a Project

- Leverage retrospectives
- Focus on meaningful issues
  - Look for unexpected or large variances
  - Correlate improvements to ability to deliver
- Strive for continuous improvement
  - Target something, however small, each iteration
  - Kind of like refactoring the process

Retrospectives

- Retrospectives ask:
  - What went well, what didn't?
  - What things do we want to keep doing?
  - What things do we want to change?
- Capture retrospective results
  - Summarize and keep visible to the team
- Create targets for action
- Each retrospective, follow up on prior targets

Completion Criteria

- We want to ensure we are *really* complete
  - Acceptance tests must pass
  - Resulting artifacts meet "goodness" criteria
  - The system is fully built and integrated
- Partial completion produces "debt"
  - Often hidden work that still needs to be done
  - Payoff of principal will add stories/tasks
  - Often debt carries interest that affects velocity
- Watch for signs of debt in your project

Other "Process Smells"

- Chronically missed estimates
- Deferred activities, not ready to move forward
- Artifacts that no one consumes
- Artifacts that sit around too long
- Extras in feature implementations
- Inability to focus on task at hand
- Idle people looking for work or waiting
- Friction, extra effort

More "Official" Things

- Agile processes can be sufficiently "disciplined" and "defined"
- May need additional formality and ceremony
- Agile cycles can be wrapped in Six Sigma
- Agile teams could be at least CMM level 3
- PMI/PMBOK does not seem agile-compatible
  - PMI emphasizes activity plan conformance
  - Agile emphasizes work state management
  - PMBOK adopting iterative/incremental?

Retrospective

Thank You for Attending!
Enjoy the Conference!