The Agile Customer’s Toolkit

XP Agile Universe 2003
Paul Hodgetts
CEO/Consultant, Agile Logic Inc.
Today’s Plan

• Introduction
• Business Planning
• Product / Project Planning
• Generating Stories
• Release & Iteration Planning
• Measuring Results
• Complexities in Customer Teams
Your Presenter

- Your presenter is Paul Hodgetts
- Founder and CEO of Agile Logic, based in Southern California
- Agile Logic provides training, coaching, consulting, and custom development services
- 20+ years experience in all aspects of software development
- Served as coach, developer, and mentor for agile teams for the past 4 years
- Recent focus on agile processes, enterprise Java, and organizational change management
- Contributing author to Extreme Programming Perspectives
- Previous presenter at XP/Agile Universe and JavaOne
- Taught at C.S.U.F, member of XP, Java, J2EE, C++ advisory boards
What Do We Mean By “Agile?”

• 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 From Requirements
  – Customers Learn from the Solution
  – Business Environment and Conditions Change
  – Business Processes are Re-engineered

• Changes From Technology
  – Tools/Platform Release New Versions
  – Actual Tool/Platform Capabilities May Vary from Plans

• Changes From People
  – Interactions are Complex
  – Individual Behavior is Unpredictable
What’s Really Different About “Agile?”

• “Ad-Hoc” Processes
  – Still the Most Common Process
  – Not Necessarily Chaotic, Just Not Consistent

• “Defined” Processes
  – Creation of Comprehensive Activity-Based Plans
  – Execution of Defined Activities
  – Management by Controlling Activities to Conform to Plan

• “Adaptive” Processes
  – Setting of Goal-Based Objectives for Deliverables
  – Opportunistic, Local Planning and Execution of Activities
  – Management by Retrospection, Learning, Adaptation
The Defined, Controlling/Optimizing/Repeatable Model Doesn’t Work So Well

• Software development is not really manufacturing
• Software development is not really engineering
An Empirical, Adaptive Model Works Better

- Software development is more of a craft
- Software development is “a cooperative game of invention and communication”

Alistair Cockburn
Agile Principles

- Principles are Guiding Ideas and Insights
- Practices Realize Principles
- Principles Transfer Between Environments
- Practices Are Context-Specific
Survey of Agile Processes

• Extreme Programming
• Scrum
• Lean Development
• DSDM
• Crystal
• Adaptive Software Development
• Feature-Driven Development
Extreme Programming

- Rapid Feedback
- Assume Simplicity
- Incremental Change
- Embracing Change
- Quality Work
- Teach Learning
- Small Initial Investment
- Play to Win

- Concrete Experiments
- Open Honest Communication
- Work With Instincts
- Accepted Responsibility
- Local Adaptation
- Travel Light
- Honest Measurement
Scrum

• “The Art of the Possible”
• Cut Through Noise with Action
• Self-Organization
• Respond Empirically
• Daily Visibility into the Project
• Incremental Product Delivery
Lean Development

• Eliminate Waste
• Amplify Learning
• Decide as Late as Possible
• Deliver as Fast as Possible
• Empower the Team
• Build Integrity In
• See the Whole
DSDM

• Active User Involvement
• Teams Empowered to Make Decisions
• Frequent Delivery of Products
• Fitness for Business Purpose is Essential Criterion for Acceptance
• Iterative and Incremental Development to Converge on Solution
• All Changes are Reversible
• Requirements Baselined at High Level
• Testing Integrated Throughout Lifecycle
• Collaborative and Cooperative Approach Between Stakeholders
Crystal

- Use Short Subprojects
- Reflect on Practices
- Work in Close Location
- Attend to Community
Adaptive Software Development

• Speculate – Collaborate – Learn
• Mission-Driven
• Component-Based
• Iterative
• Time-Boxed
• Risk-Driven
• Change-Tolerant
Characteristics of Agility

- Empowered, self-organizing teams
- Multi-discipline, cross-functional teams (whole team culture)
- Project- and product-centric focus, minimal organizational focus
- Shared responsibility, role-based accountability
- Shared vision of standards of excellence
- Close, continuous collaboration, direct communication
Characteristics of Agility

- Early, frequent, and continuous demonstration of progress through concrete deliverables
- Rapid feedback, reflection, learning, adjustment
- Small work batch sizes, minimal specialization, reduced queuing delays
- Just in time production, minimize production of artifacts not immediately (or ever) consumed
- Low friction – simplicity, minimalism, pragmatism
- Avoidance of debt, focus on forward movement
- Parallelism and opportunistic control
- Sustainable, constant, predictable pace
Potential Benefits of Agility

• Delivering the most value to the business, efficient use of resources, maximize ROI and time-to-ROI
• Faster development, higher productivity
• Flexibility to respond to change and leverage learning
• Better quality solutions, more enduring systems
• More fulfilling development culture
The Business Side Can Be Agile, Too!

- Incorporate all involved folks into a collaborative Project Community (Whole Team)
- Organize business and product strategy-setting into agile cycles
- The business and development sides need to spin at compatible rates
- Base business and product strategies on concrete, testable goals
- Be incremental – avoid large-artifact, up-front activities
Project Community

- Owners, Shareholders
- Board of Directors
- Executive Management
- Product Management, Marketing, Sales
- Project Management
- Technical Management, Administrative Management
- IA/UI Design
- Developers, Architects, DBAs
- Quality Assurance
- Operations
- End Users, Their Managers, etc.
Product Development Pipeline

1. Business Strategy
2. Product Initiatives
3. Product Features
4. Stories & Plans
5. Development
Overall Business Objectives

• What is the business trying to accomplish?
• Defined in terms of revenue, growth, market share…
• Based on satisfying the needs of the business stakeholders
• Owned by board of directors and/or executive management
Product Initiatives

• What kind of product development supports business objectives?
• Defined in terms of overall products
• Products targeted at market segments, general types of users, contracts…
• Based on industry and market research
• Owned by marketing/product development, perhaps with executive management
Product Feature Sets

• What does the product have to deliver to realize the initiatives?
• Defined in terms of features and functionality
• Features targeted at the needs of the identified users
• Based on market research, user workshops, competitive analysis…
• Owned by marketing and/or product development
Stories and Plans

• How will we go about implementing the features?
• Defined in terms of stories and release plans
• Targeted at the user needs, balanced by business needs and schedules
• Based on market windows, development capabilities (velocity)
• Owned by product management, project management, development
Development

• Make it so using development practices

• Results in (XP):
  – Tasks
  – Unit tests
  – Code
  – Deliverable Product
Product Development Cycles

- Business Cycles
- Product Life Cycles
- Releases
- Iterations
- Tasks & Episodes
Business Cycles and Development Cycles

- Wrapping the development cycles with the business cycles
- Business cycles provide raw materials for development cycles
Development Cycles in XP

- Programming episode
- Story/task cycle
- Iteration cycle
- Release cycle
Business Cycle Practices

- Cycle Charter
- Product Strategy Planning
- Product Feature Set Planning
- Release Planning and Strategic Feedback
- Strategic Investment and Resource Commitments
- Release Development
- Business Retrospectives
Cycle Charter

• Elements of a Charter
  – Defines the overall mission
  – Defines specific, testable goals – business stories
  – Defines desired or required schedule requirements
  – Defines available resources and limits
  – Defines the project community, roles and accountability
Cycle Charter

- Defines overall business objectives
- Business objectives defined with business stories
- Business story states a testable business goal
- Business stories may not be always be “completed,” but nevertheless serve to align the team
- Example: Increase revenue by 15%
- Business stories can also define overall constraints
- Example: Development costs can not exceed $1.2M per month
Product Strategy Planning

• Defines products and capabilities
• Products defined with product stories
• Product stories can be tested, but are generally an intermediate artifact
• Example: Sell a shopping cart system for retail book sellers
Product Feature Set Planning

• Defines product features, requirements, priorities
• Product features defined with informal, light-weight use cases
• Use case incrementally developed and tested with stories
• Use cases can be an intermediate artifact, but useful for communication
• Example: Use case for “User adds book to shopping cart”
Release Planning and Strategic Feedback

- Defines stories for implementing use cases
- Produces prioritized and estimated story queue
- Standard XP/Scrum release planning practices
- Results of planning used as feedback for investment planning
Strategic Investment and Resource Commitments

- Defines resource constraints (team size, equipment, budget) for cycle
- May loop back to adjust plans based on resource constraints
- Resource adjustments may take time (hiring, purchasing)
Release Development

- Standard XP iteration development cycles
- Iteration results may force adjustments to business cycle
- Release retrospective done to provide business feedback
Business Retrospectives

- Release results used to learn and adjust business cycle
- Test the business stories
- Business retrospective done to provide feedback for next cycle
Business Cycle Duration

- Business is a bigger ship and turns more slowly
- Feedback takes longer to measure revenue, market share changes
- Minimize cycle duration – quarterly, semi-annually at most
- Schedule multiple releases per cycle where possible
Business Cycle Intervals

• Business cycles may be offset from XP cycles
• Business planning has lead time and needs to be ready for releases
• Business retrospective needs delay to measure slow-moving feedback
• No easy answer, but a rhythm will emerge
Concrete Business Feedback

• Avoiding guesswork and pet projects
• Measurement with real metrics
• This is business, not science – keep it simple!
• Metric resolution should be realistic with acceptable tolerances
Example of Concrete Feedback

• Business Story: Increase revenue by 15%
• Metric: Measure revenue, calculate increase
Example of Concrete Feedback

• Product Story: Generate 10% more revenue from shopping cart product
• Product Feature Set Planning: Reduce number of cancelled transactions by 20%
• Use Case: Change shopping cart check out to ask for just-in-time registration info
• Metric: Usability testing to validate new use case is less threatening
• Metric: Measure number of cancelled transactions
• Metric: Measure revenue from shopping cart product, calculate increase
Traceability in the Overall Process
Why Bother with Traceability?

- Businesses waste effort with inefficiencies and fuzzy strategies
- Better-focused development can provide a competitive advantage
- Better defined and communicated strategies creates purposed teams
- Teams with a collective purpose are more motivated
Misusing Traceability

• Traceability is not an accountability or blame mechanism
• Traceability is not an enforcement mechanism for command and control
• Traceability is not an after-the-fact way to validate strategies
• Traceability should be obvious and not need verification practices
Forward-Backwards Traceability

- Traceability flows naturally forward when using generative practices
- Traceability is best demonstrated by looking backwards
- “Jane is working on a task to help increase our quarterly revenue by 15%”
The Traceability Chain

• Development episodes trace back to tasks
• Tasks trace back to release/iteration plans and stories
• Stories trace back to product feature plans (use cases)
• Features trace back to product strategy stories
• Product strategies trace back to business stories
Multi-Customer Strategies and Practices
The Customer

- In XP, the Customer works directly with the Developers
- In Scrum, the Customer is called the Product Owner
- Canonical XP has the End User fulfilling the Customer role
- The Customer defines requirements and priorities
The Distant Customer

- What if there is space and/or time distance between the End User and Developers?
- Employ a User Proxy as End User representative to the Developers
- The User Proxy fulfills the XP Customer role
Many Customers

• What if there are multiple End Users?
• Form a Customer Board whose constituents represent the End Users
• The Customer Board appoints a single User Proxy as representative to the Developers
• The User Proxy has the accountability as Customer
• Multiple members of the Customer Board may work with the Developers
Diversity of Customers

- What if there are multiple types of End Users and/or multiple Stakeholder interests?
- Stakeholders are not End Users, but have a say in requirements and priorities
- The Customer Board pattern still applies
- Each member’s voting interest may be weighted
Customer Practices for Customer Boards

- Establish the Customer Board
- Story Telling
- Mapping and Merging Stories
- Organizing Stories to Strategic Priorities
- Establish Normative Priorities

Thanks and credit to Brad Appleton for his XP list posting!
Establish the Customer Board

• Choose members based on their ability to represent their interests
• Determine the number and composition of the board
• Number of members based on covering the varying types of interests
• Weighting of members based on importance of their interests to the overall product
• Board composition should be subject to retrospectives and adjustments
Story Telling

- Each board member has their requirements and priorities
- Allow each member to complete their release stories one by one
- All other board members are expected to actively listen
- Use different color cards for each board member
Mapping and Merging Stories

• Lay out all stories on the table
• Map duplicate stories to a single story
• Merge similar stories into a single story
• Keep placeholders for mapped and merged stories
Organizing Stories to Strategic Priorities

- Identify stories that fulfill strategic business and product strategies
- Assign a priority to each story based on this mapping
- Story priority is usually the higher of the applicable strategic priorities
- Group stories into strategic priority categories
Establish Normative Priorities

- Go through each strategic priority category
- Each story in category is assigned a normative priority
- Normative priority is based on the board member’s weighting and story priority
Forming a Single Release Plan

• Review the overall layout of the stories by strategic category and normative priority
• Decide on allocation of release plan bandwidth to strategic categories
• Fill single release plan stream with stories
• Adjust as necessary based on group consensus
• The Customer is accountable for representing the results
Generating Stories from Product Strategies and Features
Generating Product Strategies and Features

• Not a formulaic operation – requires some leg work
• Based on product development best practices
  – Market research
  – Competitive analysis
  – Customer feedback
  – Usability testing
Product Feature Set Generation

• Canonical XP is incremental and just-in-time
  – Need a car… A convertible… A red one… With under 15,000 miles…

• But many cultures are more up-front and carry inventory
  – Larger plans required for funding
  – Contracts specify “fixed” scope
Documenting Feature 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. 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
Extracting 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 Use 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 Use Cases and Stories

- Example of story that maps to use case
- “Implement the User Checkout use case. Handle only the positive path, credit card payment scenario. Don’t worry about sales tax or discount calculations.”
Example of Use 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 – don’t worry about state tax tables.”
Issues in Adopting Agile Business Practices
Issues in Adopting Agile Business Practices

• The Value of Leaders and Mentors
• A Bit of Preparation and Education Never Hurts
• Respect Everyone’s Turf – This Isn’t Just a Programmer Thing
• Countering Fear and Resistance One Small Step at a Time
Issues in Adopting Agile Business Practices

• Pilot Projects and Adoption on a Shoestring
• The Challenge of Building and Sustaining Momentum
• The Fall of Empires and the Rise of the Collective
• Adapting and the Learning Organization
Want More?

- Presentation summary will be posted on resources page
- Probable whitepapers, perhaps a book to follow
- Agile Logic can help you implement what you’ve learned in this tutorial
- Paul can be contacted at:
  - Paul Hodgetts
  - Agile Logic
  - 1519 East Chapman Ave. #254
  - Fullerton, CA, USA 92831
  - Phone: (866) 64-AGILE
  - Email: phodgetts@agilelogic.com
  - Website: www.agilelogic.com
Thank You for Attending!

Enjoy the rest of the Conference!