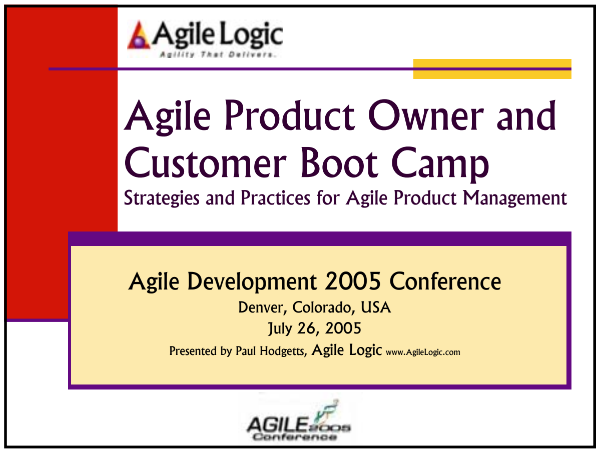


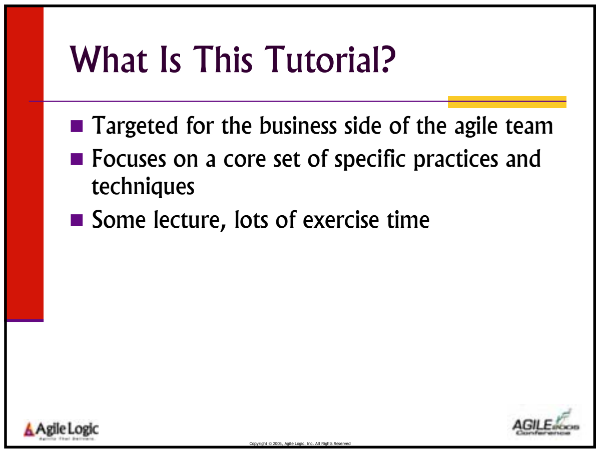
Agile Logic
Agility That Delivers.

Agile Product Owner and Customer Boot Camp

Strategies and Practices for Agile Product Management

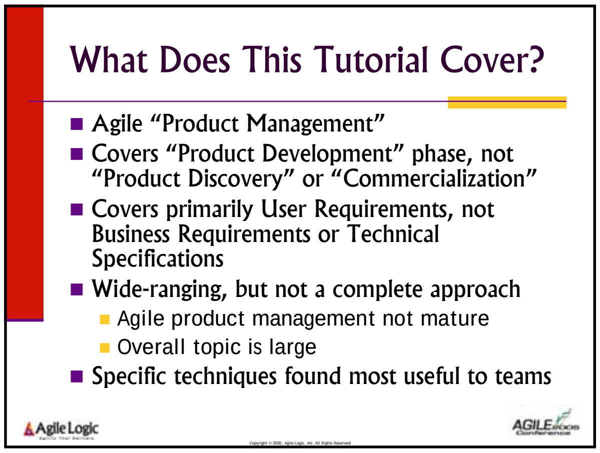
Agile Development 2005 Conference
Denver, Colorado, USA
July 26, 2005
Presented by Paul Hodgetts, Agile Logic www.AgileLogic.com

AGILE 2005 Conference



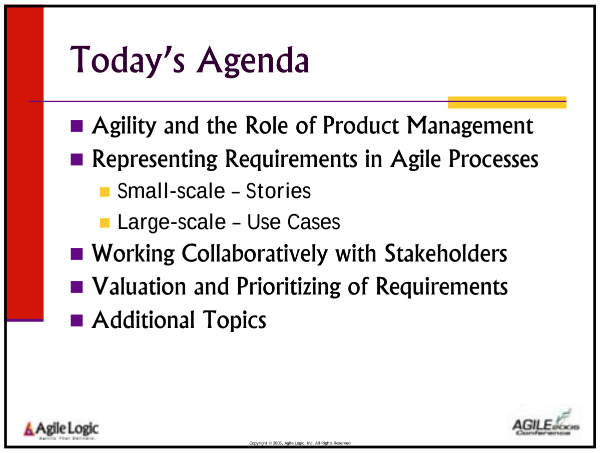
What Is This Tutorial?

- Targeted for the business side of the agile team
- Focuses on a core set of specific practices and techniques
- Some lecture, lots of exercise time



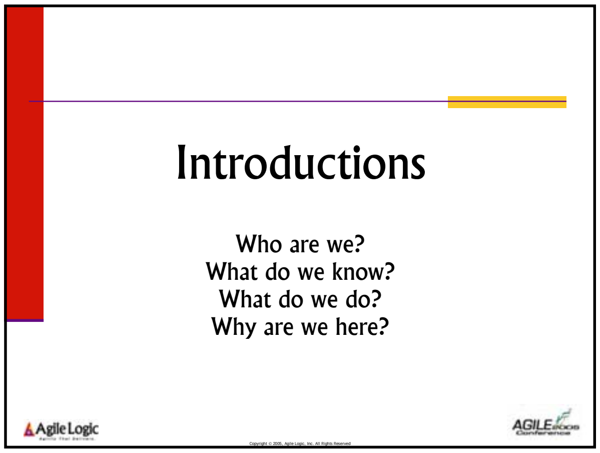
What Does This Tutorial Cover?

- Agile "Product Management"
- Covers "Product Development" phase, not "Product Discovery" or "Commercialization"
- Covers primarily User Requirements, not Business Requirements or Technical Specifications
- Wide-ranging, but not a complete approach
  - Agile product management not mature
  - Overall topic is large
- Specific techniques found most useful to teams



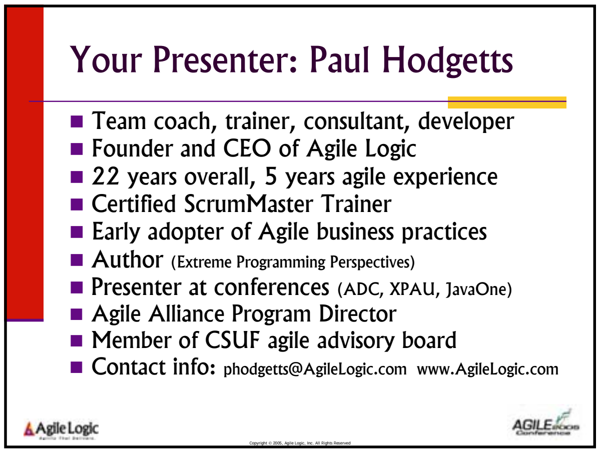
Today's Agenda

- Agility and the Role of Product Management
- Representing Requirements in Agile Processes
  - Small-scale – Stories
  - Large-scale – Use Cases
- Working Collaboratively with Stakeholders
- Valuation and Prioritizing of Requirements
- Additional Topics



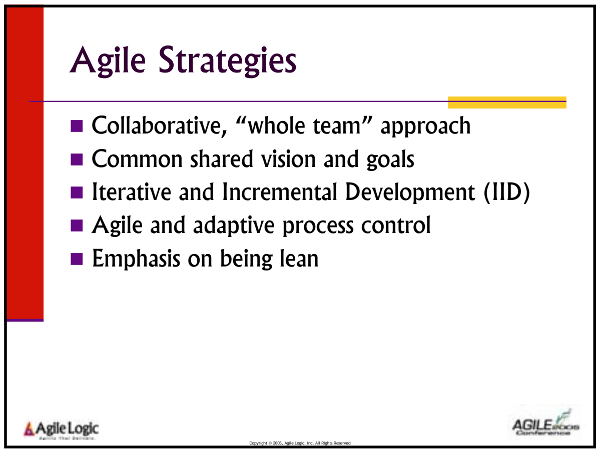
Introductions

Who are we?
What do we know?
What do we do?
Why are we here?



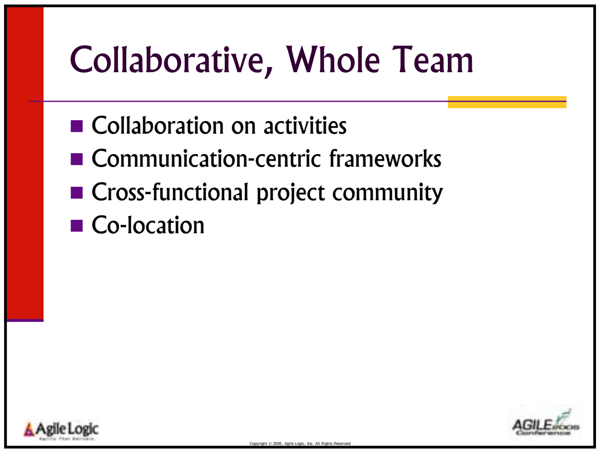
Your Presenter: Paul Hodgetts

- Team coach, trainer, consultant, developer
- Founder and CEO of Agile Logic
- 22 years overall, 5 years agile experience
- Certified ScrumMaster Trainer
- Early adopter of Agile business practices
- Author (Extreme Programming Perspectives)
- Presenter at conferences (ADC, XPAU, JavaOne)
- Agile Alliance Program Director
- Member of CSUF agile advisory board
- Contact info: phodgetts@AgileLogic.com www.AgileLogic.com



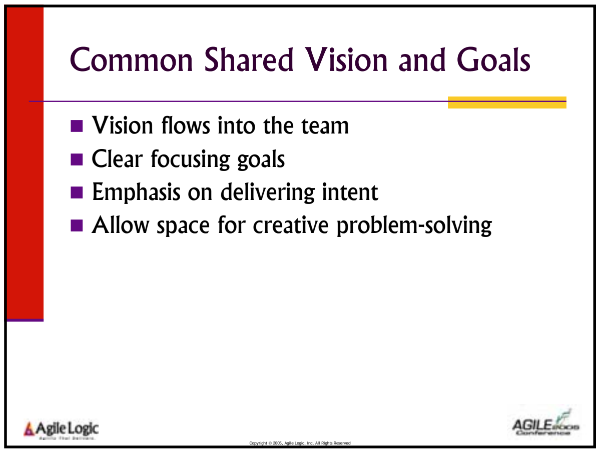
Agile Strategies

- Collaborative, “whole team” approach
- Common shared vision and goals
- Iterative and Incremental Development (IID)
- Agile and adaptive process control
- Emphasis on being lean



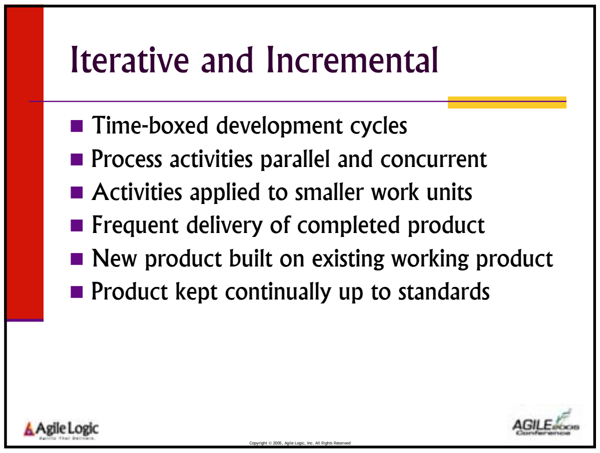
Collaborative, Whole Team

- Collaboration on activities
- Communication-centric frameworks
- Cross-functional project community
- Co-location



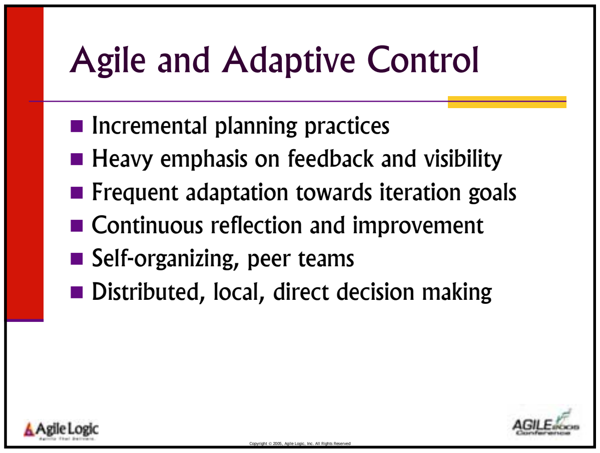
Common Shared Vision and Goals

- Vision flows into the team
- Clear focusing goals
- Emphasis on delivering intent
- Allow space for creative problem-solving



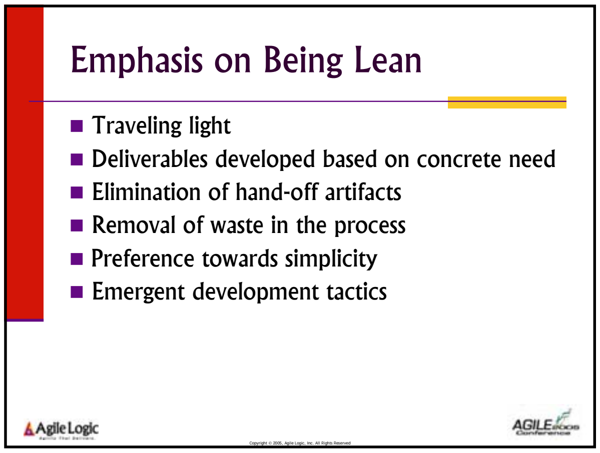
Iterative and Incremental

- Time-boxed development cycles
- Process activities parallel and concurrent
- Activities applied to smaller work units
- Frequent delivery of completed product
- New product built on existing working product
- Product kept continually up to standards



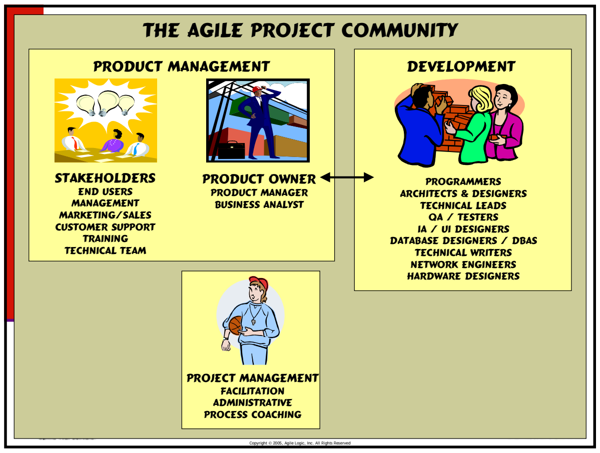
Agile and Adaptive Control

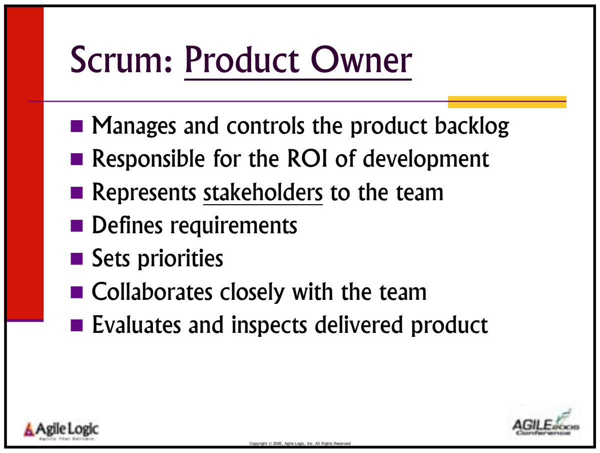
- Incremental planning practices
- Heavy emphasis on feedback and visibility
- Frequent adaptation towards iteration goals
- Continuous reflection and improvement
- Self-organizing, peer teams
- Distributed, local, direct decision making



Emphasis on Being Lean

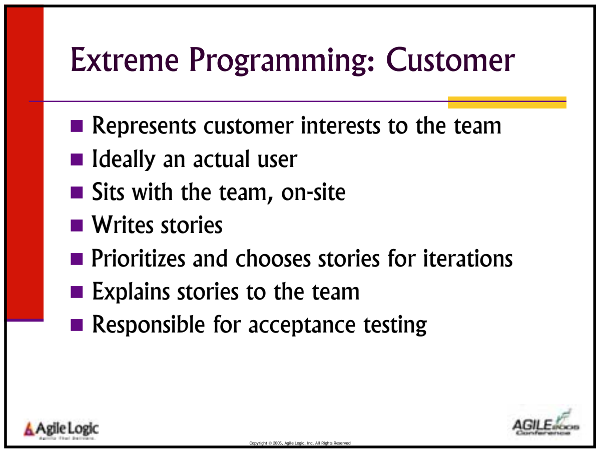
- Traveling light
- Deliverables developed based on concrete need
- Elimination of hand-off artifacts
- Removal of waste in the process
- Preference towards simplicity
- Emergent development tactics





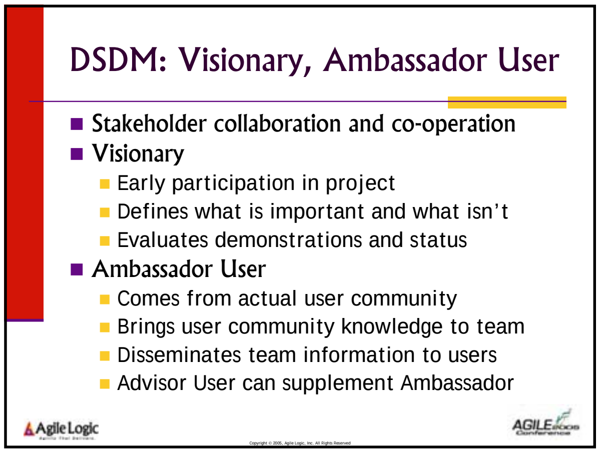
Scrum: Product Owner

- Manages and controls the product backlog
- Responsible for the ROI of development
- Represents stakeholders to the team
- Defines requirements
- Sets priorities
- Collaborates closely with the team
- Evaluates and inspects delivered product



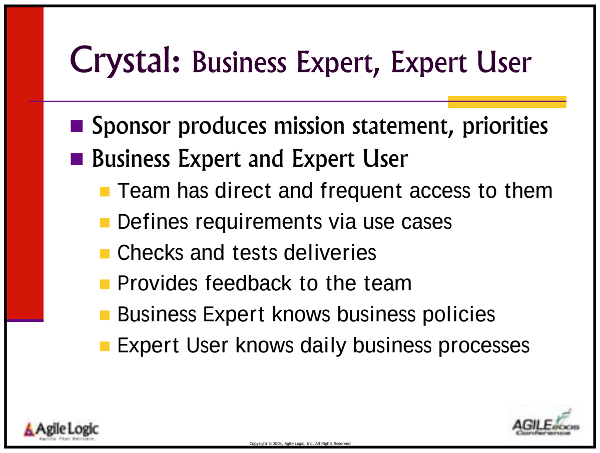
Extreme Programming: Customer

- Represents customer interests to the team
- Ideally an actual user
- Sits with the team, on-site
- Writes stories
- Prioritizes and chooses stories for iterations
- Explains stories to the team
- Responsible for acceptance testing



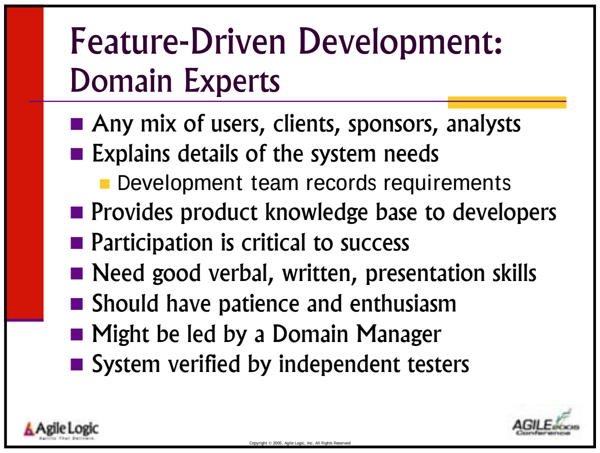
DSDM: Visionary, Ambassador User

- Stakeholder collaboration and co-operation
- Visionary
  - Early participation in project
  - Defines what is important and what isn't
  - Evaluates demonstrations and status
- Ambassador User
  - Comes from actual user community
  - Brings user community knowledge to team
  - Disseminates team information to users
  - Advisor User can supplement Ambassador



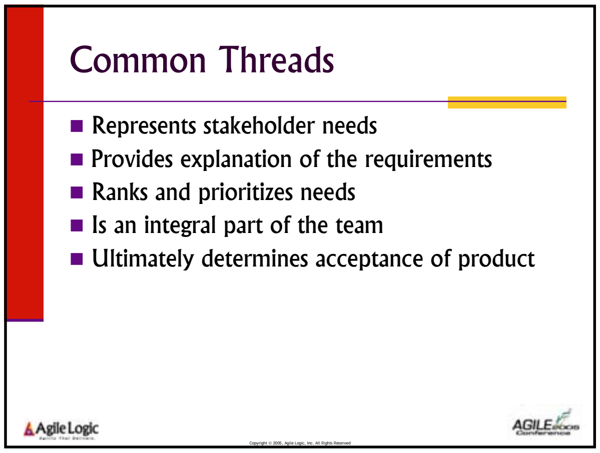
Crystal: Business Expert, Expert User

- Sponsor produces mission statement, priorities
- Business Expert and Expert User
  - Team has direct and frequent access to them
  - Defines requirements via use cases
  - Checks and tests deliveries
  - Provides feedback to the team
  - Business Expert knows business policies
  - Expert User knows daily business processes



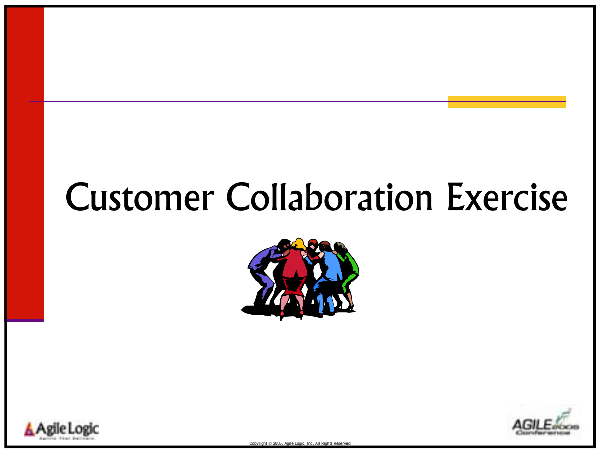
Feature-Driven Development: Domain Experts

- Any mix of users, clients, sponsors, analysts
- Explains details of the system needs
  - Development team records requirements
- Provides product knowledge base to developers
- Participation is critical to success
- Need good verbal, written, presentation skills
- Should have patience and enthusiasm
- Might be led by a Domain Manager
- System verified by independent testers

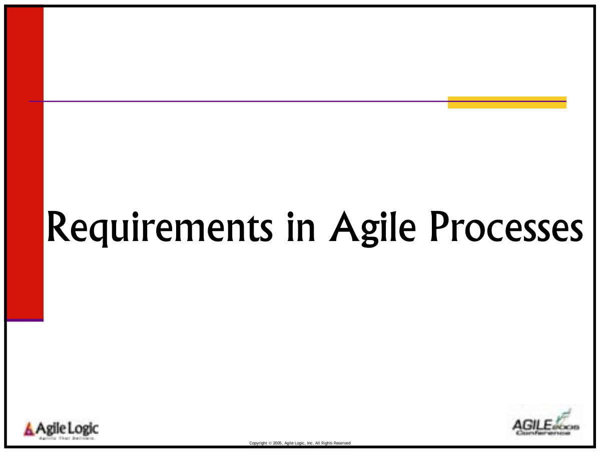


Common Threads

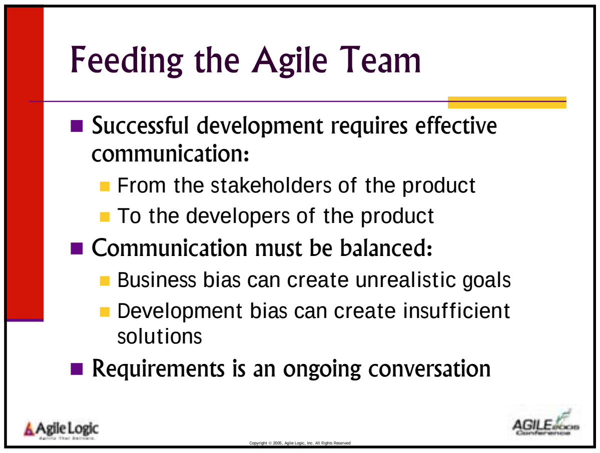
- Represents stakeholder needs
- Provides explanation of the requirements
- Ranks and prioritizes needs
- Is an integral part of the team
- Ultimately determines acceptance of product



Customer Collaboration Exercise

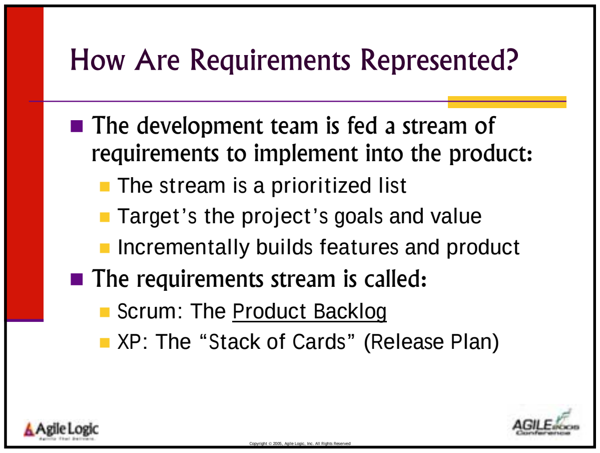


Requirements in Agile Processes



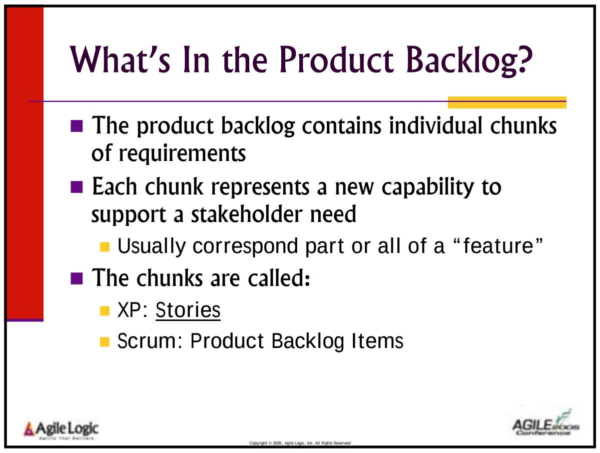
Feeding the Agile Team

- Successful development requires effective communication:
  - From the stakeholders of the product
  - To the developers of the product
- Communication must be balanced:
  - Business bias can create unrealistic goals
  - Development bias can create insufficient solutions
- Requirements is an ongoing conversation



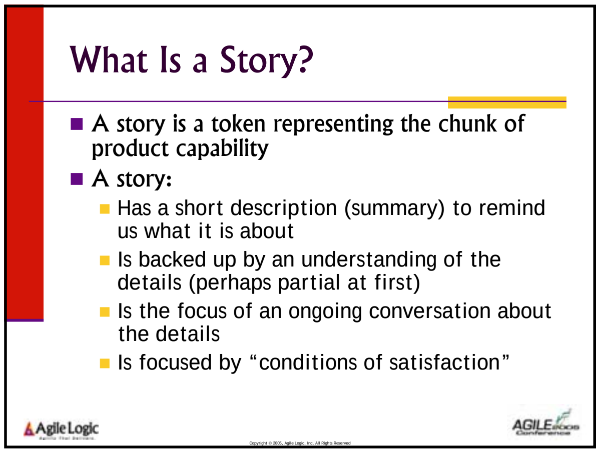
How Are Requirements Represented?

- The development team is fed a stream of requirements to implement into the product:
  - The stream is a prioritized list
  - Target's the project's goals and value
  - Incrementally builds features and product
- The requirements stream is called:
  - Scrum: The Product Backlog
  - XP: The "Stack of Cards" (Release Plan)



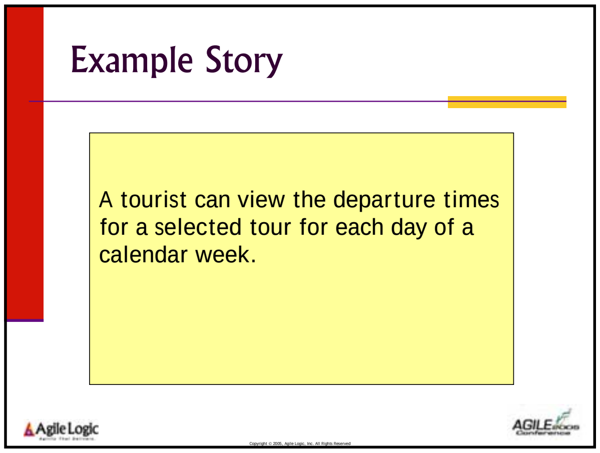
What's In the Product Backlog?

- The product backlog contains individual chunks of requirements
- Each chunk represents a new capability to support a stakeholder need
  - Usually correspond part or all of a "feature"
- The chunks are called:
  - XP: Stories
  - Scrum: Product Backlog Items



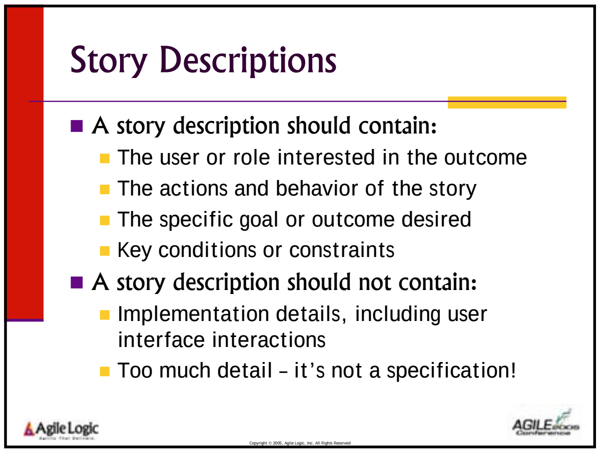
What Is a Story?

- A story is a token representing the chunk of product capability
- A story:
  - Has a short description (summary) to remind us what it is about
  - Is backed up by an understanding of the details (perhaps partial at first)
  - Is the focus of an ongoing conversation about the details
  - Is focused by "conditions of satisfaction"



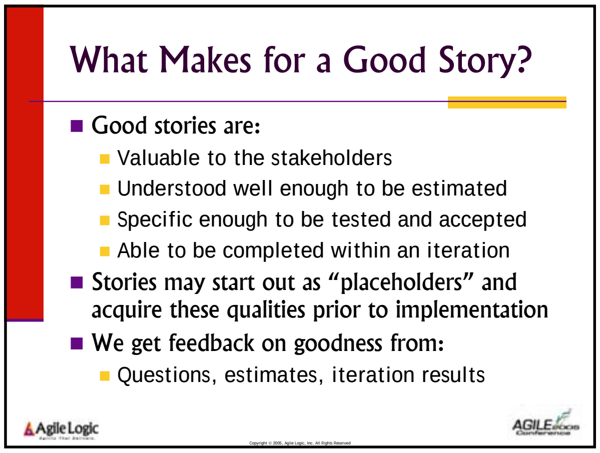
Example Story

A tourist can view the departure times for a selected tour for each day of a calendar week.



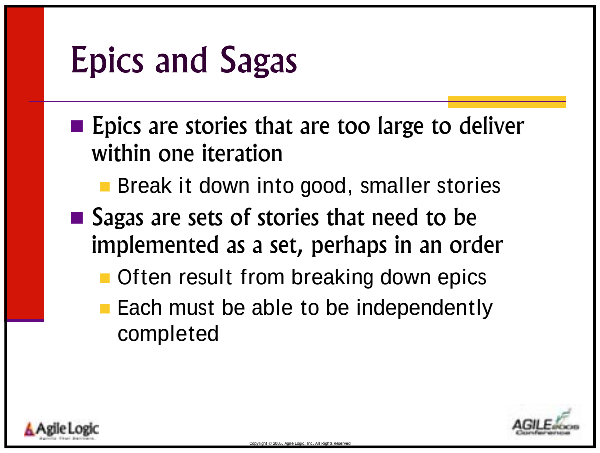
Story Descriptions

- A story description should contain:
  - The user or role interested in the outcome
  - The actions and behavior of the story
  - The specific goal or outcome desired
  - Key conditions or constraints
- A story description should not contain:
  - Implementation details, including user interface interactions
  - Too much detail – it's not a specification!



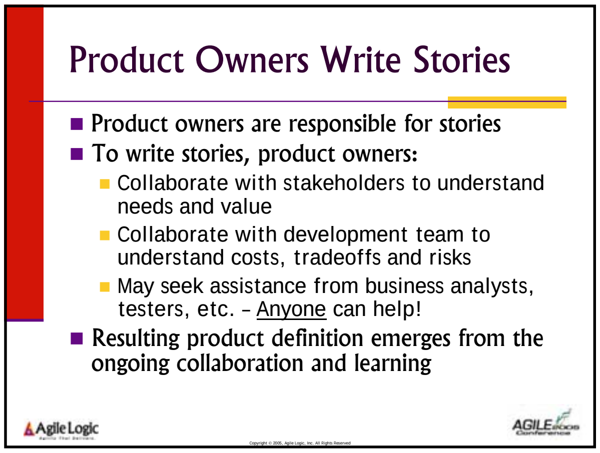
What Makes for a Good Story?

- Good stories are:
  - Valuable to the stakeholders
  - Understood well enough to be estimated
  - Specific enough to be tested and accepted
  - Able to be completed within an iteration
- Stories may start out as "placeholders" and acquire these qualities prior to implementation
- We get feedback on goodness from:
  - Questions, estimates, iteration results



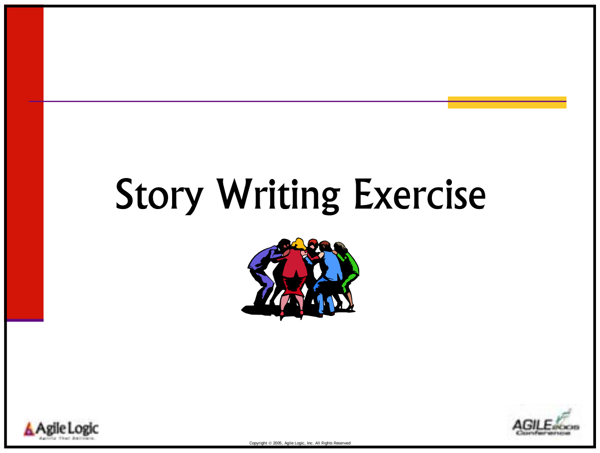
Epics and Sagas

- Epics are stories that are too large to deliver within one iteration
  - Break it down into good, smaller stories
- Sagas are sets of stories that need to be implemented as a set, perhaps in an order
  - Often result from breaking down epics
  - Each must be able to be independently completed

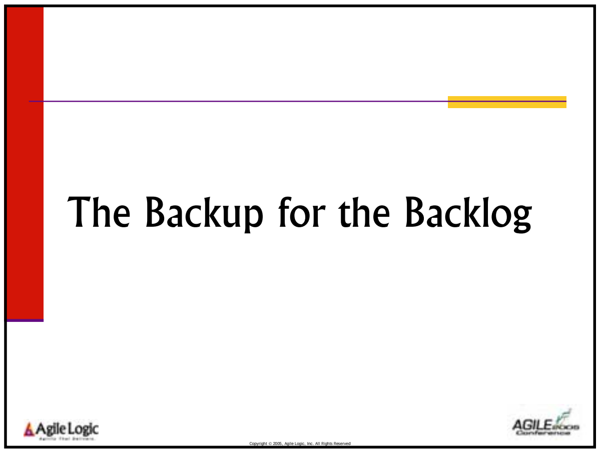


Product Owners Write Stories

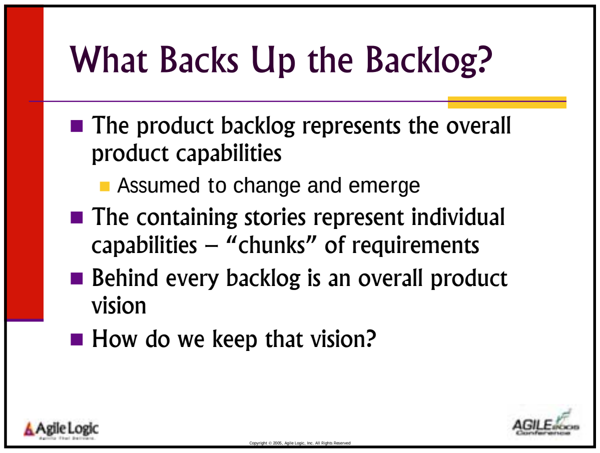
- Product owners are responsible for stories
- To write stories, product owners:
  - Collaborate with stakeholders to understand needs and value
  - Collaborate with development team to understand costs, tradeoffs and risks
  - May seek assistance from business analysts, testers, etc. – Anyone can help!
- Resulting product definition emerges from the ongoing collaboration and learning



Story Writing Exercise

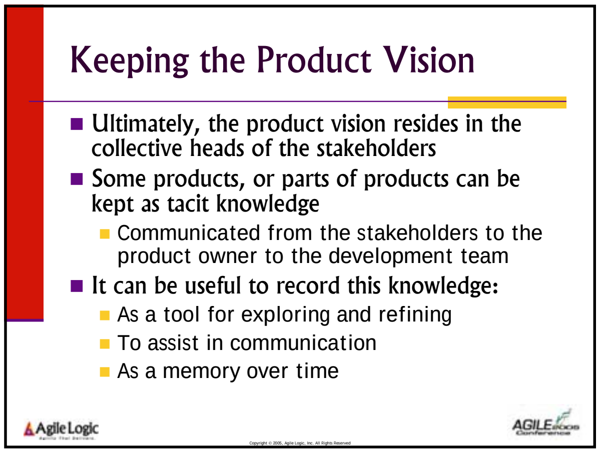


The Backup for the Backlog



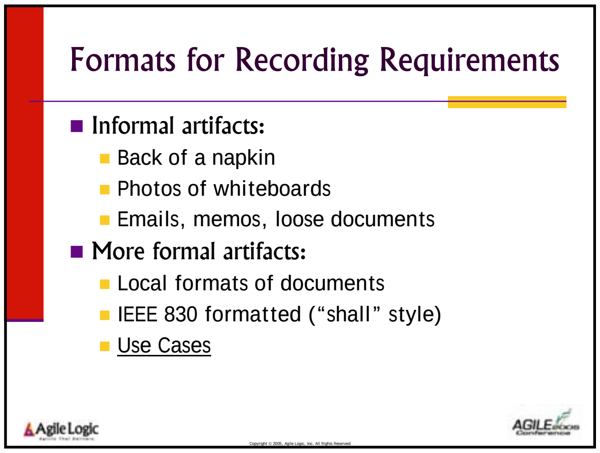
What Backs Up the Backlog?

- The product backlog represents the overall product capabilities
  - Assumed to change and emerge
- The containing stories represent individual capabilities – "chunks" of requirements
- Behind every backlog is an overall product vision
- How do we keep that vision?



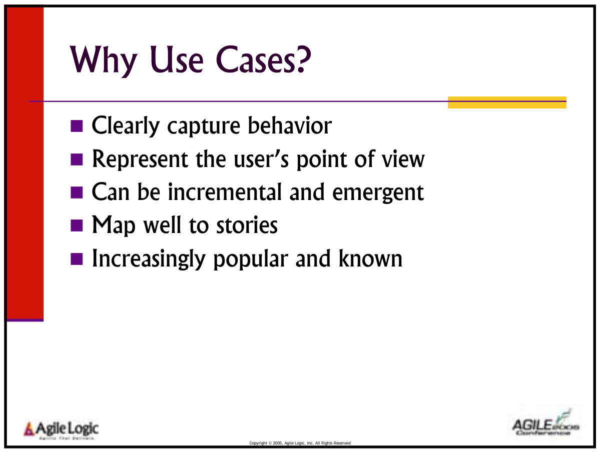
Keeping the Product Vision

- Ultimately, the product vision resides in the collective heads of the stakeholders
- Some products, or parts of products can be kept as tacit knowledge
  - Communicated from the stakeholders to the product owner to the development team
- It can be useful to record this knowledge:
  - As a tool for exploring and refining
  - To assist in communication
  - As a memory over time



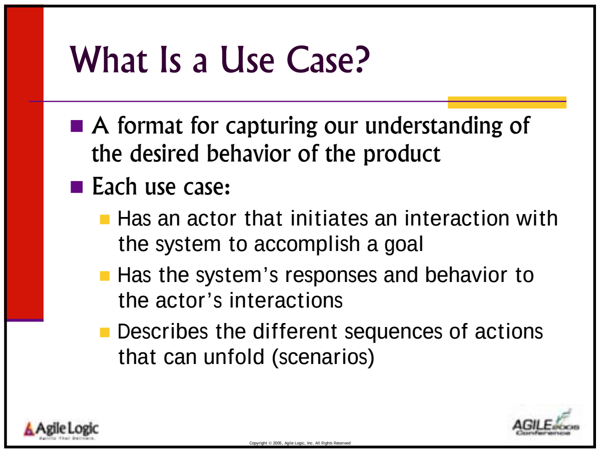
Formats for Recording Requirements

- Informal artifacts:
  - Back of a napkin
  - Photos of whiteboards
  - Emails, memos, loose documents
- More formal artifacts:
  - Local formats of documents
  - IEEE 830 formatted ("shall" style)
  - Use Cases



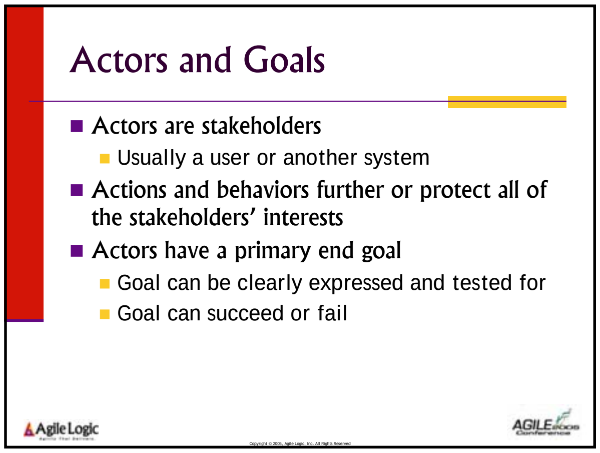
Why Use Cases?

- Clearly capture behavior
- Represent the user's point of view
- Can be incremental and emergent
- Map well to stories
- Increasingly popular and known



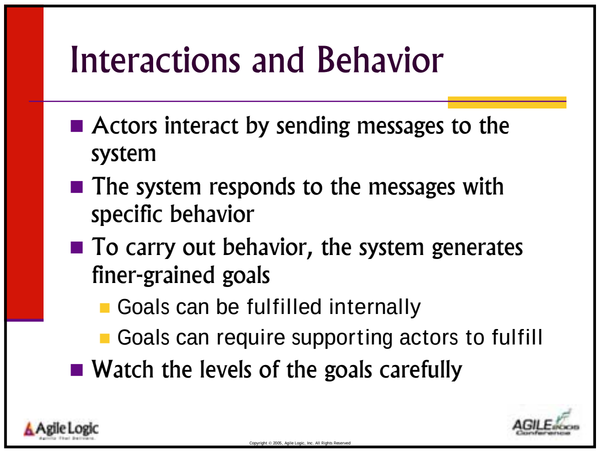
What Is a Use Case?

- A format for capturing our understanding of the desired behavior of the product
- Each use case:
  - Has an actor that initiates an interaction with the system to accomplish a goal
  - Has the system's responses and behavior to the actor's interactions
  - Describes the different sequences of actions that can unfold (scenarios)



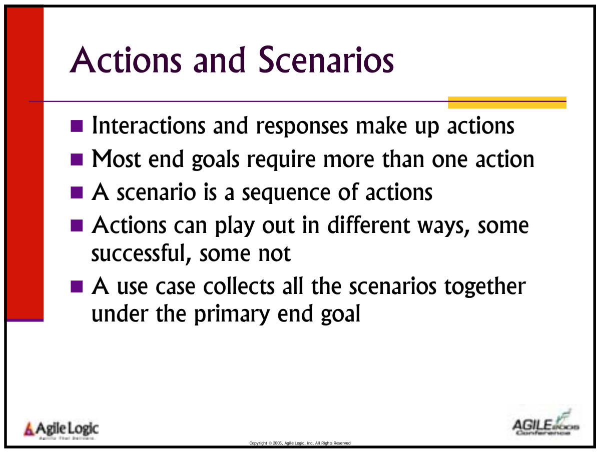
Actors and Goals

- Actors are stakeholders
  - Usually a user or another system
- Actions and behaviors further or protect all of the stakeholders' interests
- Actors have a primary end goal
  - Goal can be clearly expressed and tested for
  - Goal can succeed or fail



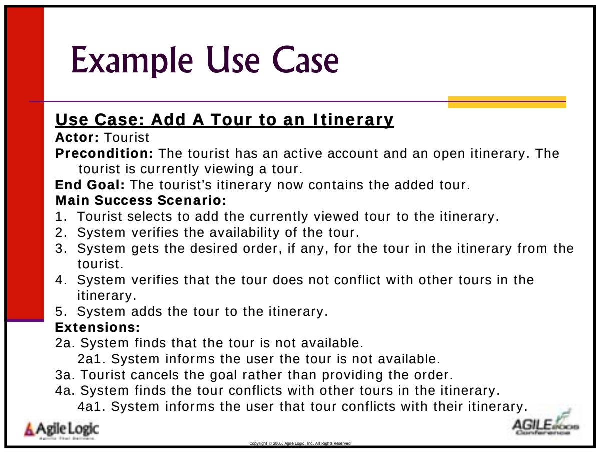
Interactions and Behavior

- Actors interact by sending messages to the system
- The system responds to the messages with specific behavior
- To carry out behavior, the system generates finer-grained goals
  - Goals can be fulfilled internally
  - Goals can require supporting actors to fulfill
- Watch the levels of the goals carefully



Actions and Scenarios

- Interactions and responses make up actions
- Most end goals require more than one action
- A scenario is a sequence of actions
- Actions can play out in different ways, some successful, some not
- A use case collects all the scenarios together under the primary end goal



Example Use Case

Use Case: Add A Tour to an Itinerary

Actor: Tourist

Precondition: The tourist has an active account and an open itinerary. The tourist is currently viewing a tour.

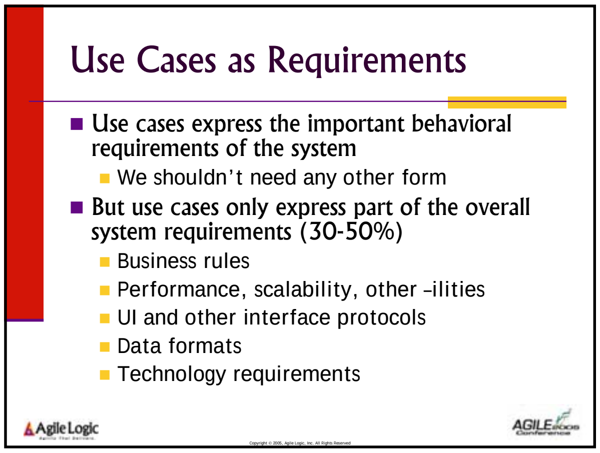
End Goal: The tourist's itinerary now contains the added tour.

Main Success Scenario:

1. Tourist selects to add the currently viewed tour to the itinerary.
2. System verifies the availability of the tour.
3. System gets the desired order, if any, for the tour in the itinerary from the tourist.
4. System verifies that the tour does not conflict with other tours in the itinerary.
5. System adds the tour to the itinerary.

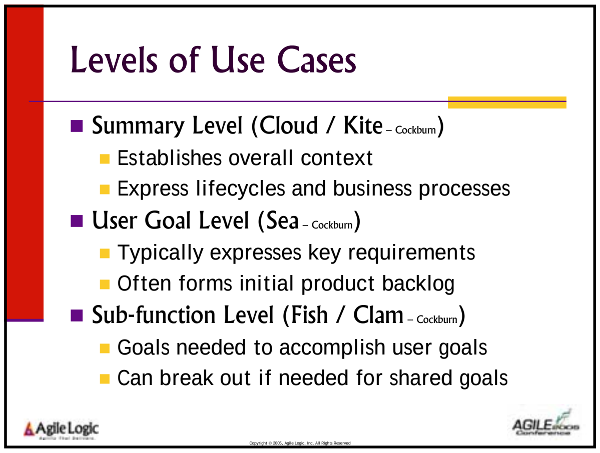
Extensions:

2a. System finds that the tour is not available.
    2a1. System informs the user the tour is not available.
3a. Tourist cancels the goal rather than providing the order.
4a. System finds the tour conflicts with other tours in the itinerary.
    4a1. System informs the user that tour conflicts with their itinerary.



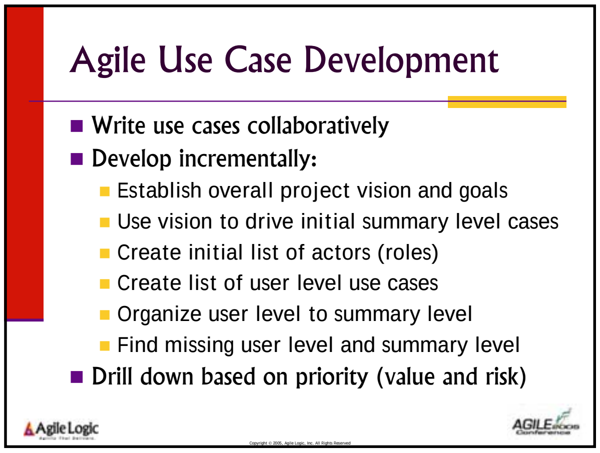
Use Cases as Requirements

- Use cases express the important behavioral requirements of the system
  - We shouldn't need any other form
- But use cases only express part of the overall system requirements (30-50%)
  - Business rules
  - Performance, scalability, other –ilities
  - UI and other interface protocols
  - Data formats
  - Technology requirements



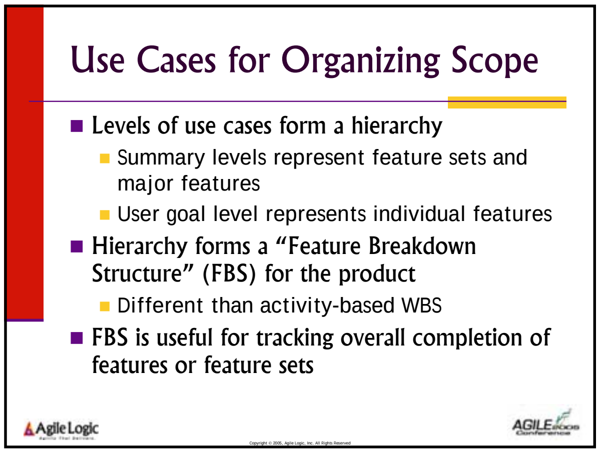
Levels of Use Cases

- Summary Level (Cloud / Kite – Cockburn)
  - Establishes overall context
  - Express lifecycles and business processes
- User Goal Level (Sea – Cockburn)
  - Typically expresses key requirements
  - Often forms initial product backlog
- Sub-function Level (Fish / Clam – Cockburn)
  - Goals needed to accomplish user goals
  - Can break out if needed for shared goals



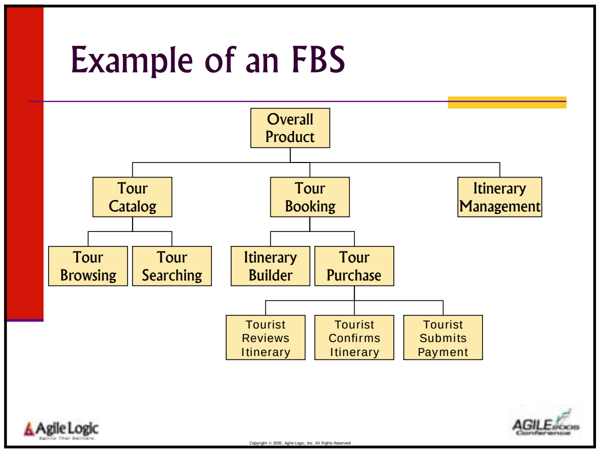
Agile Use Case Development

- Write use cases collaboratively
- Develop incrementally:
  - Establish overall project vision and goals
  - Use vision to drive initial summary level cases
  - Create initial list of actors (roles)
  - Create list of user level use cases
  - Organize user level to summary level
  - Find missing user level and summary level
- Drill down based on priority (value and risk)

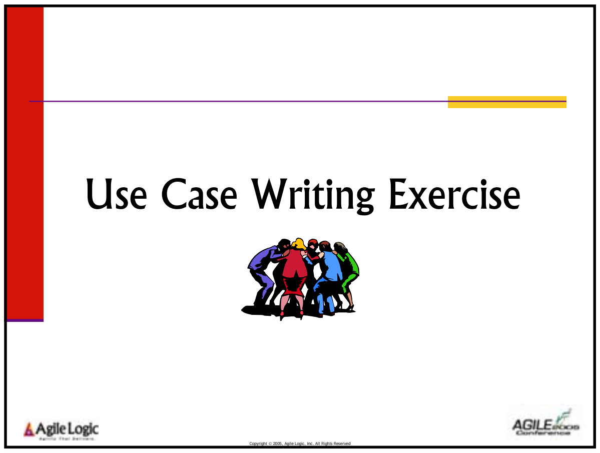


Use Cases for Organizing Scope

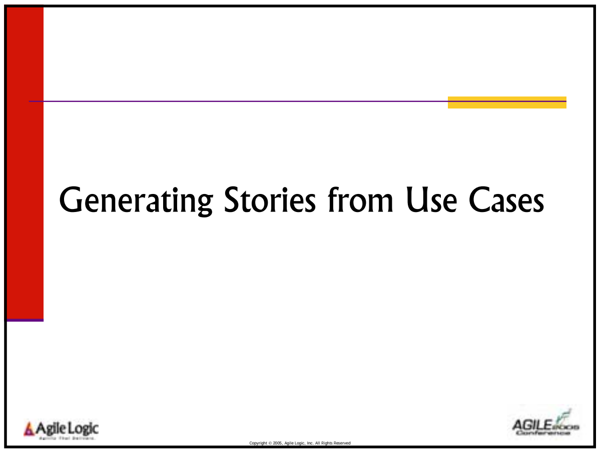
- Levels of use cases form a hierarchy
  - Summary levels represent feature sets and major features
  - User goal level represents individual features
- Hierarchy forms a “Feature Breakdown Structure” (FBS) for the product
  - Different than activity-based WBS
- FBS is useful for tracking overall completion of features or feature sets



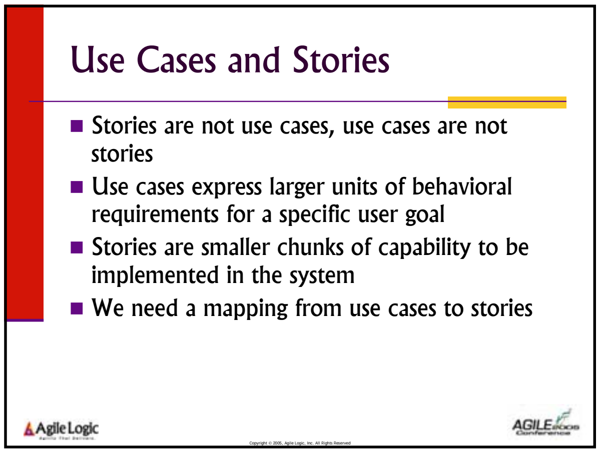
Example of an FBS



Use Case Writing Exercise

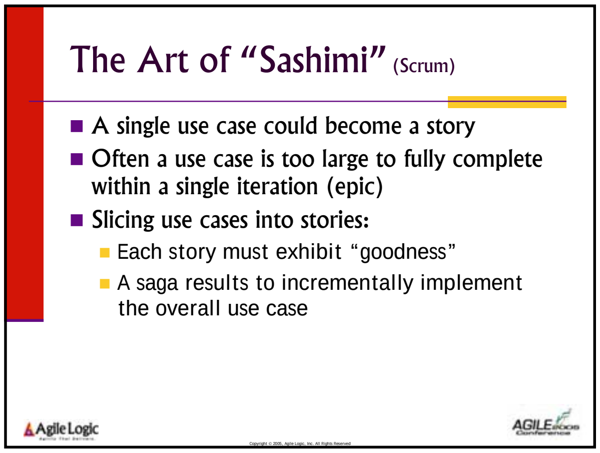


Generating Stories from Use Cases



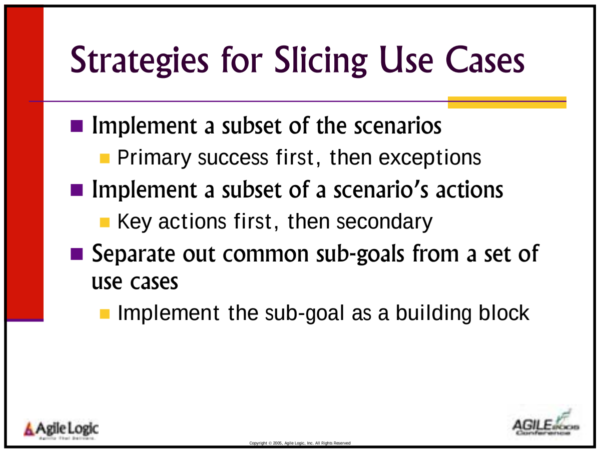
Use Cases and Stories

- Stories are not use cases, use cases are not stories
- Use cases express larger units of behavioral requirements for a specific user goal
- Stories are smaller chunks of capability to be implemented in the system
- We need a mapping from use cases to stories



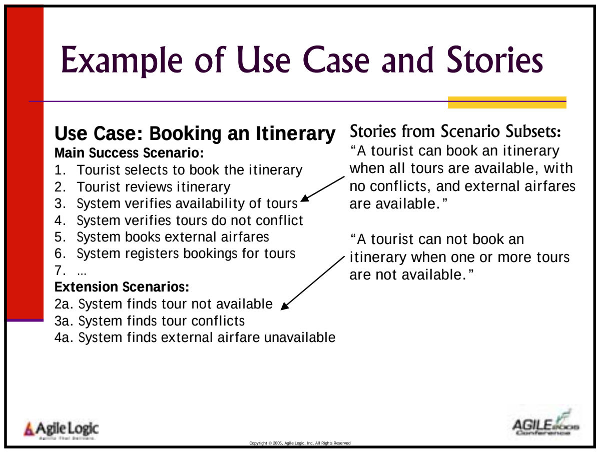
The Art of "Sashimi" (Scrum)

- A single use case could become a story
- Often a use case is too large to fully complete within a single iteration (epic)
- Slicing use cases into stories:
  - Each story must exhibit "goodness"
  - A saga results to incrementally implement the overall use case



Strategies for Slicing Use Cases

- Implement a subset of the scenarios
  - Primary success first, then exceptions
- Implement a subset of a scenario's actions
  - Key actions first, then secondary
- Separate out common sub-goals from a set of use cases
  - Implement the sub-goal as a building block



Example of Use Case and Stories

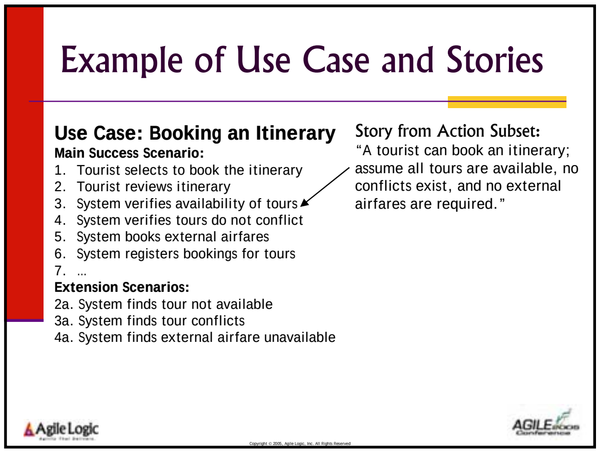
Use Case: Booking an Itinerary

Main Success Scenario:
1. Tourist selects to book the itinerary
2. Tourist reviews itinerary
3. System verifies availability of tours
4. System verifies tours do not conflict
5. System books external airfares
6. System registers bookings for tours
7. …

Extension Scenarios:
2a. System finds tour not available
3a. System finds tour conflicts
4a. System finds external airfare unavailable

Stories from Scenario Subsets:
"A tourist can book an itinerary when all tours are available, with no conflicts, and external airfares are available."

"A tourist can not book an itinerary when one or more tours are not available."



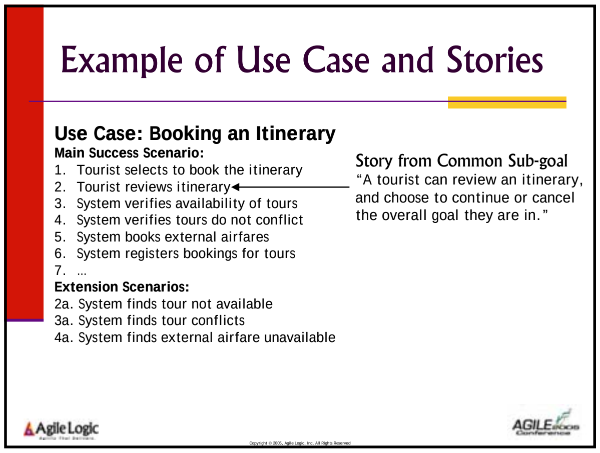
Example of Use Case and Stories

Use Case: Booking an Itinerary

Main Success Scenario:
1. Tourist selects to book the itinerary
2. Tourist reviews itinerary
3. System verifies availability of tours
4. System verifies tours do not conflict
5. System books external airfares
6. System registers bookings for tours
7. …

Extension Scenarios:
2a. System finds tour not available
3a. System finds tour conflicts
4a. System finds external airfare unavailable

Story from Action Subset:
"A tourist can book an itinerary; assume all tours are available, no conflicts exist, and no external airfares are required."



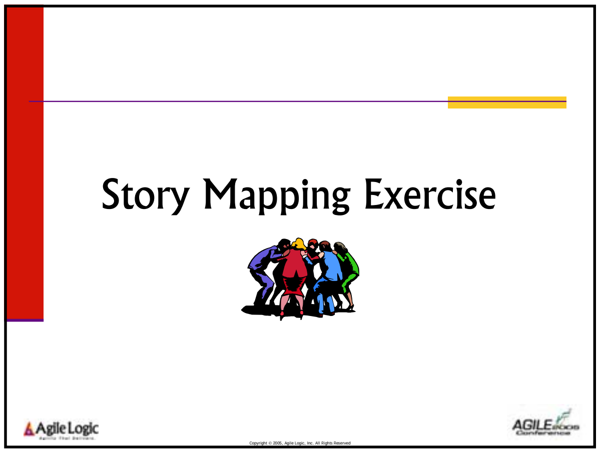
Example of Use Case and Stories

Use Case: Booking an Itinerary

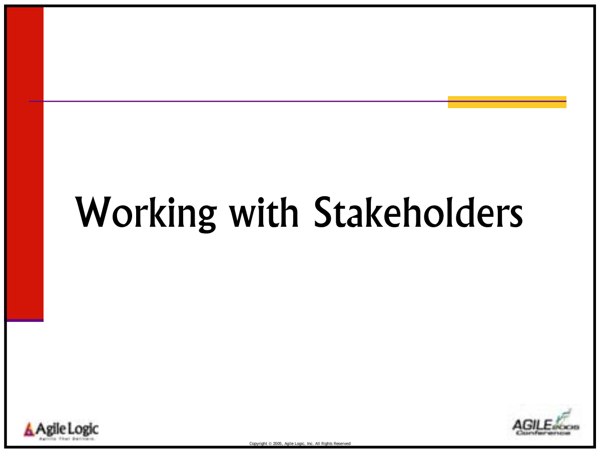
Main Success Scenario:
1. Tourist selects to book the itinerary
2. Tourist reviews itinerary
3. System verifies availability of tours
4. System verifies tours do not conflict
5. System books external airfares
6. System registers bookings for tours
7. …

Extension Scenarios:
2a. System finds tour not available
3a. System finds tour conflicts
4a. System finds external airfare unavailable

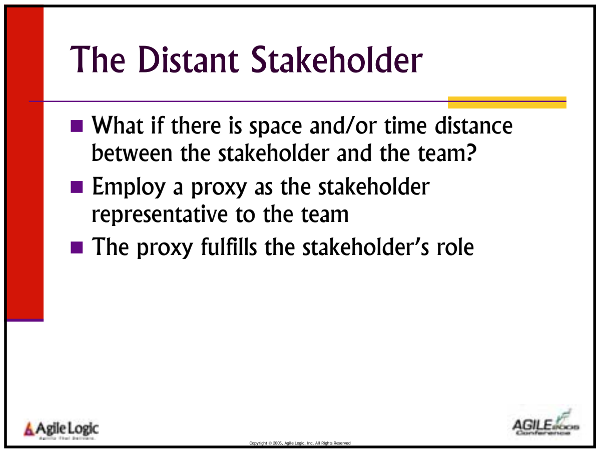
Story from Common Sub-goal
"A tourist can review an itinerary, and choose to continue or cancel the overall goal they are in."



Story Mapping Exercise

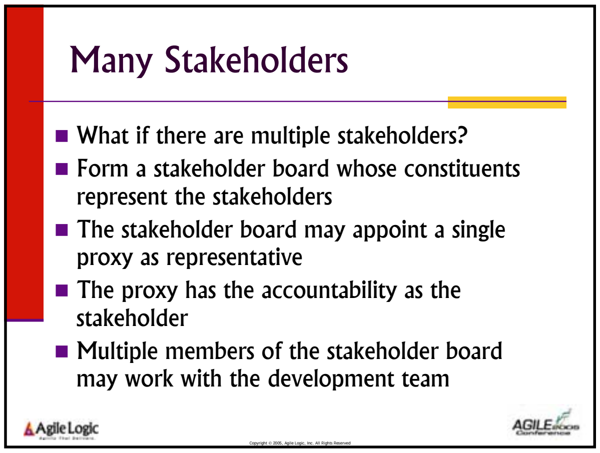


Working with Stakeholders



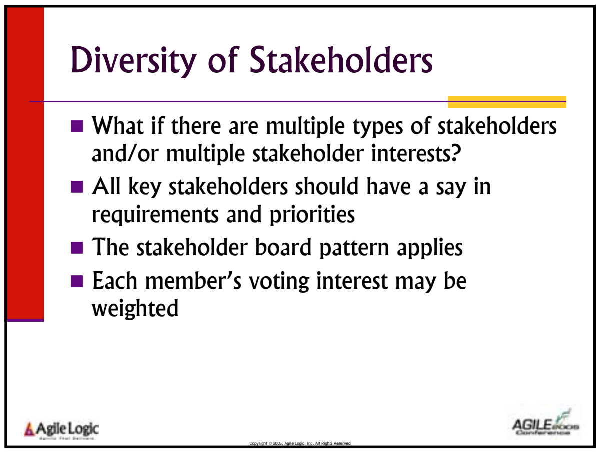
The Distant Stakeholder

- What if there is space and/or time distance between the stakeholder and the team?
- Employ a proxy as the stakeholder representative to the team
- The proxy fulfills the stakeholder's role



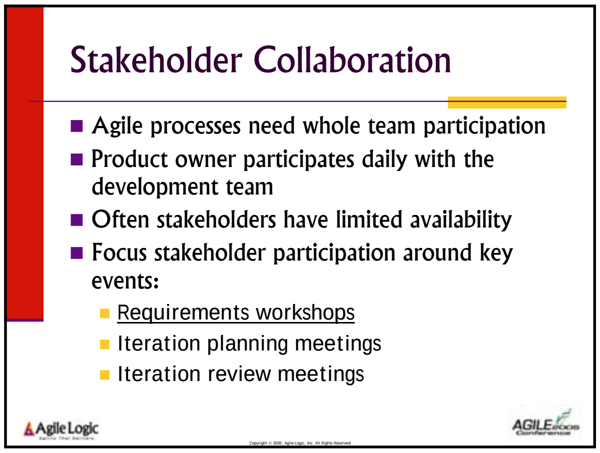
Many Stakeholders

- What if there are multiple stakeholders?
- Form a stakeholder board whose constituents represent the stakeholders
- The stakeholder board may appoint a single proxy as representative
- The proxy has the accountability as the stakeholder
- Multiple members of the stakeholder board may work with the development team



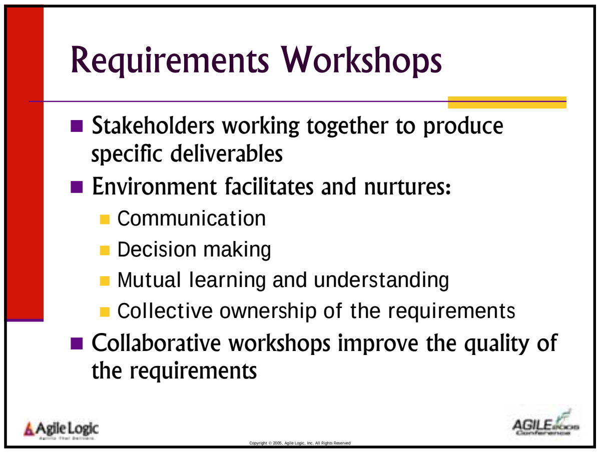
Diversity of Stakeholders

- What if there are multiple types of stakeholders and/or multiple stakeholder interests?
- All key stakeholders should have a say in requirements and priorities
- The stakeholder board pattern applies
- Each member's voting interest may be weighted



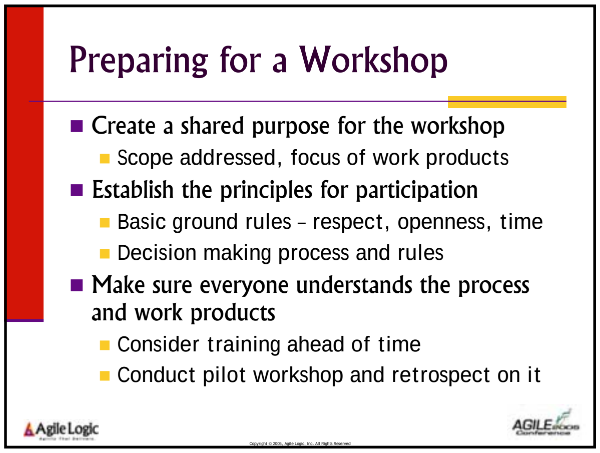
Stakeholder Collaboration

- Agile processes need whole team participation
- Product owner participates daily with the development team
- Often stakeholders have limited availability
- Focus stakeholder participation around key events:
  - Requirements workshops
  - Iteration planning meetings
  - Iteration review meetings



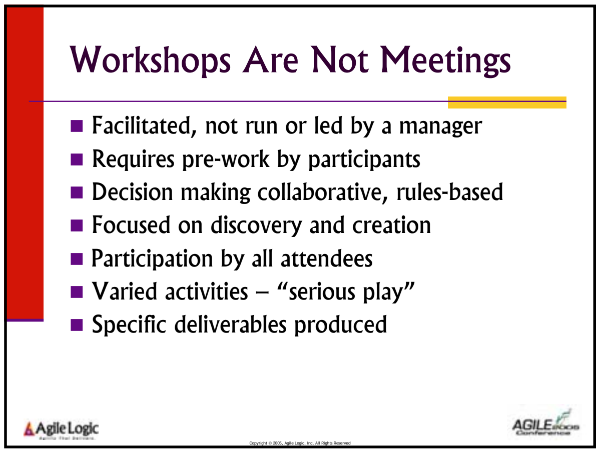
Requirements Workshops

- Stakeholders working together to produce specific deliverables
- Environment facilitates and nurtures:
  - Communication
  - Decision making
  - Mutual learning and understanding
  - Collective ownership of the requirements
- Collaborative workshops improve the quality of the requirements



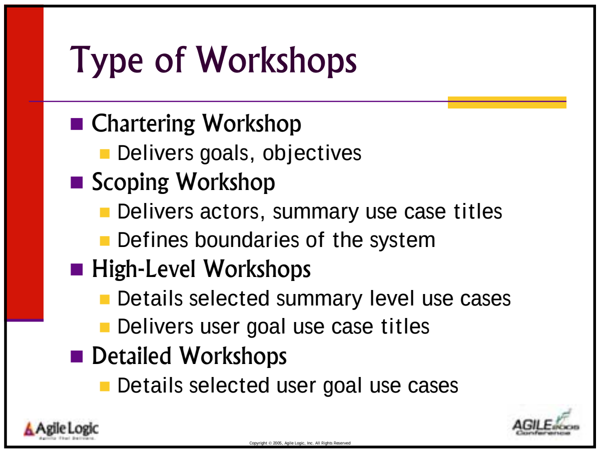
Preparing for a Workshop

- Create a shared purpose for the workshop
  - Scope addressed, focus of work products
- Establish the principles for participation
  - Basic ground rules – respect, openness, time
  - Decision making process and rules
- Make sure everyone understands the process and work products
  - Consider training ahead of time
  - Conduct pilot workshop and retrospect on it



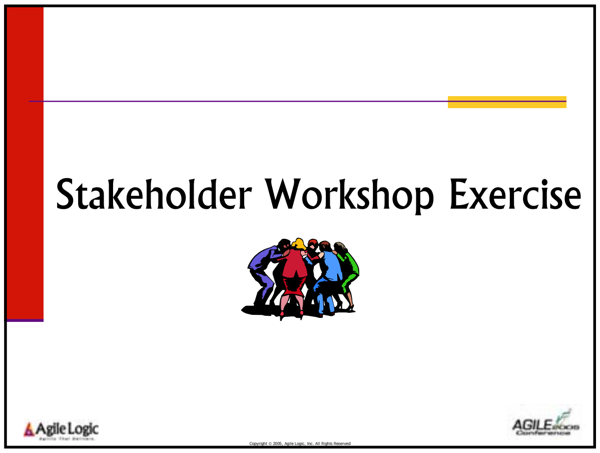
Workshops Are Not Meetings

- Facilitated, not run or led by a manager
- Requires pre-work by participants
- Decision making collaborative, rules-based
- Focused on discovery and creation
- Participation by all attendees
- Varied activities – "serious play"
- Specific deliverables produced

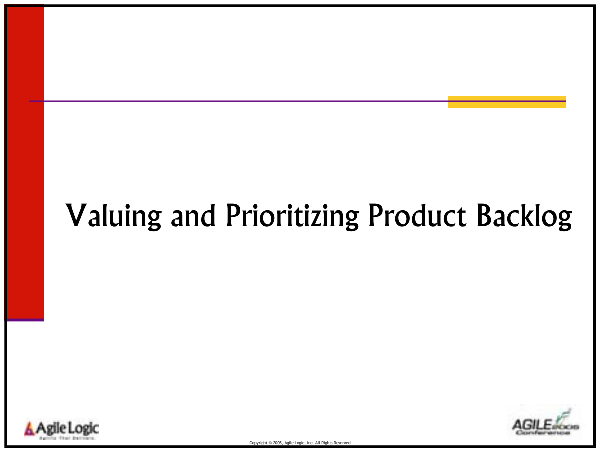


Type of Workshops

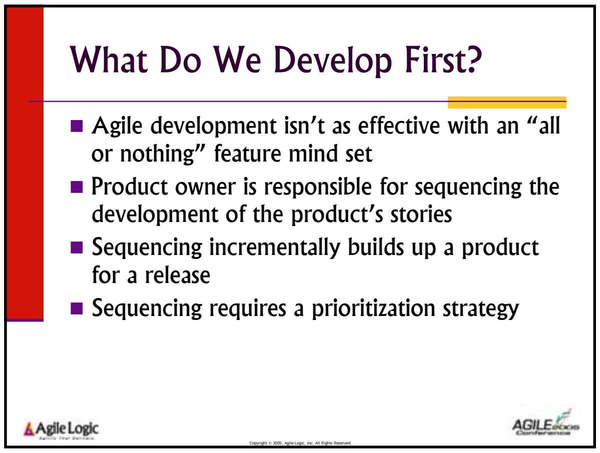
- Chartering Workshop
  - Delivers goals, objectives
- Scoping Workshop
  - Delivers actors, summary use case titles
  - Defines boundaries of the system
- High-Level Workshops
  - Details selected summary level use cases
  - Delivers user goal use case titles
- Detailed Workshops
  - Details selected user goal use cases



Stakeholder Workshop Exercise

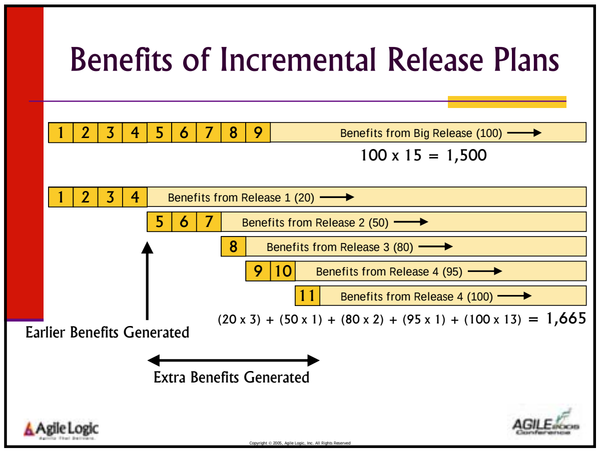


Valuing and Prioritizing Product Backlog



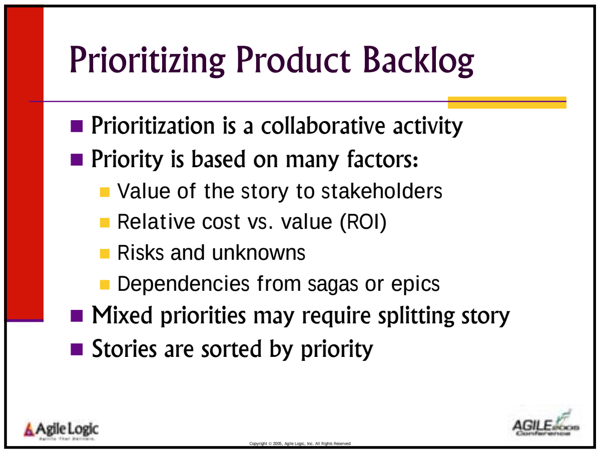
What Do We Develop First?

- Agile development isn't as effective with an "all or nothing" feature mind set
- Product owner is responsible for sequencing the development of the product's stories
- Sequencing incrementally builds up a product for a release
- Sequencing requires a prioritization strategy



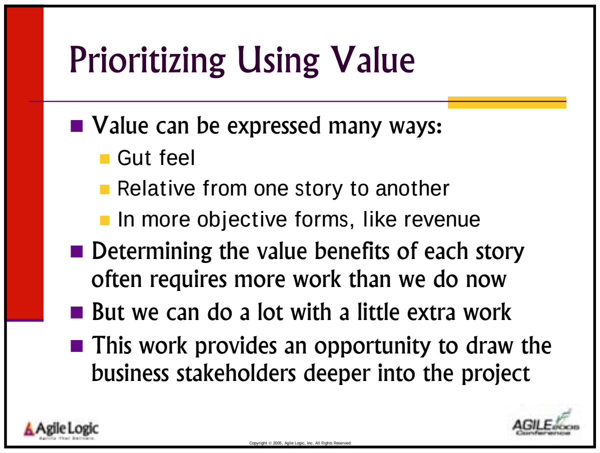
Benefits of Incremental Release Plans

1 2 3 4 5 6 7 8 9
Benefits from Big Release (100)
100 x 15 = 1,500
1 2 3 4
Benefits from Release 1 (20)
5 6 7
Benefits from Release 2 (50)
8
Benefits from Release 3 (80)
9 10
Benefits from Release 4 (95)
11
Benefits from Release 4 (100)
(20 x 3) + (50 x 1) + (80 x 2) + (95 x 1) + (100 x 13) = 1,665
Earlier Benefits Generated
Extra Benefits Generated



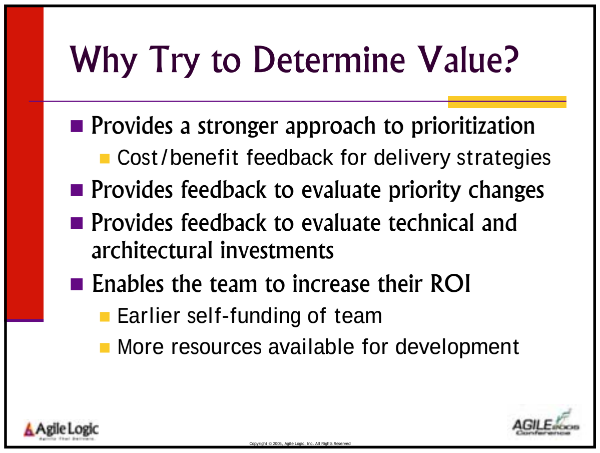
Prioritizing Product Backlog

- Prioritization is a collaborative activity
- Priority is based on many factors:
  - Value of the story to stakeholders
  - Relative cost vs. value (ROI)
  - Risks and unknowns
  - Dependencies from sagas or epics
- Mixed priorities may require splitting story
- Stories are sorted by priority



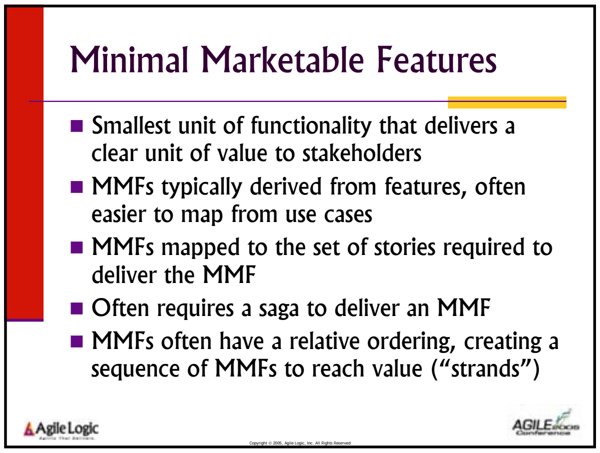
Prioritizing Using Value

- Value can be expressed many ways:
  - Gut feel
  - Relative from one story to another
  - In more objective forms, like revenue
- Determining the value benefits of each story often requires more work than we do now
- But we can do a lot with a little extra work
- This work provides an opportunity to draw the business stakeholders deeper into the project



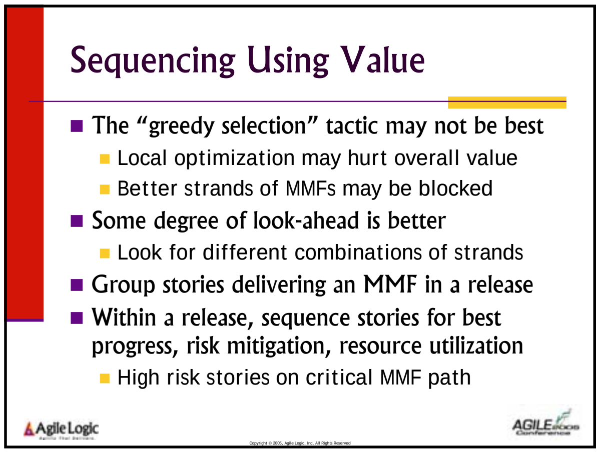
Why Try to Determine Value?

- Provides a stronger approach to prioritization
  - Cost/benefit feedback for delivery strategies
- Provides feedback to evaluate priority changes
- Provides feedback to evaluate technical and architectural investments
- Enables the team to increase their ROI
  - Earlier self-funding of team
  - More resources available for development



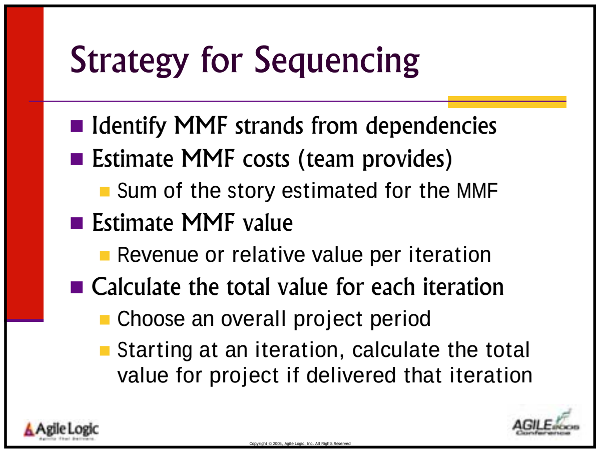
Minimal Marketable Features

- Smallest unit of functionality that delivers a clear unit of value to stakeholders
- MMFs typically derived from features, often easier to map from use cases
- MMFs mapped to the set of stories required to deliver the MMF
- Often requires a saga to deliver an MMF
- MMFs often have a relative ordering, creating a sequence of MMFs to reach value (“strands”)



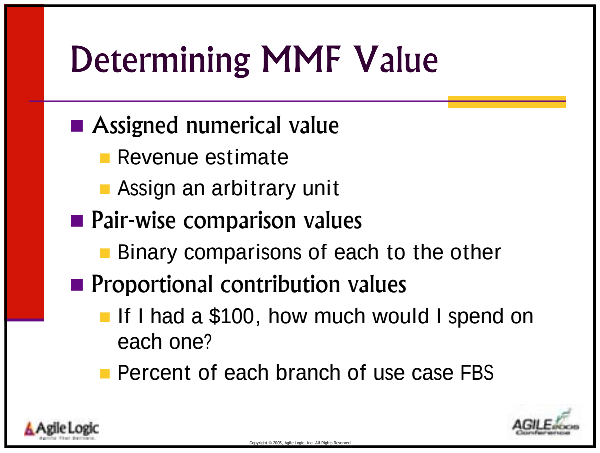
Sequencing Using Value

- The “greedy selection” tactic may not be best
  - Local optimization may hurt overall value
  - Better strands of MMFs may be blocked
- Some degree of look-ahead is better
  - Look for different combinations of strands
- Group stories delivering an MMF in a release
- Within a release, sequence stories for best progress, risk mitigation, resource utilization
  - High risk stories on critical MMF path



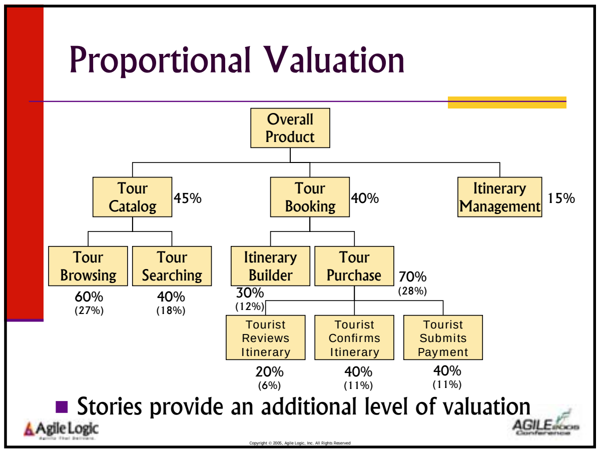
Strategy for Sequencing

- Identify MMF strands from dependencies
- Estimate MMF costs (team provides)
  - Sum of the story estimated for the MMF
- Estimate MMF value
  - Revenue or relative value per iteration
- Calculate the total value for each iteration
  - Choose an overall project period
  - Starting at an iteration, calculate the total value for project if delivered that iteration

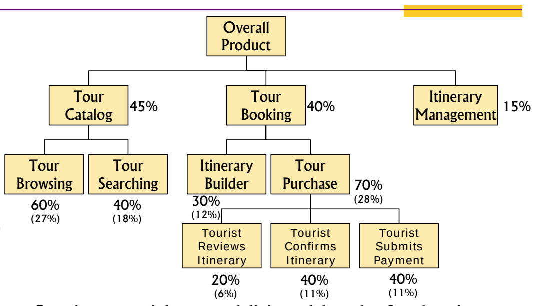


Determining MMF Value

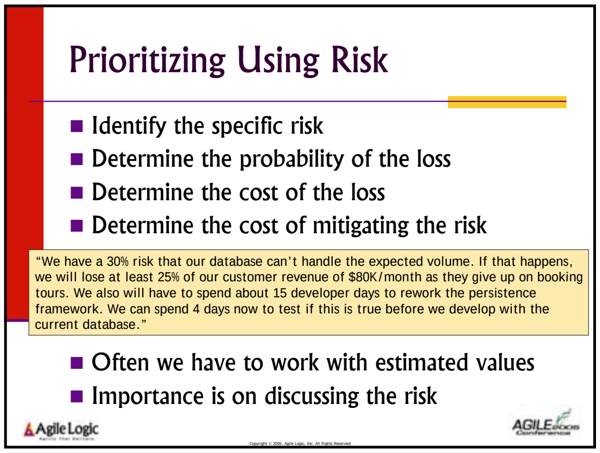
- Assigned numerical value
  - Revenue estimate
  - Assign an arbitrary unit
- Pair-wise comparison values
  - Binary comparisons of each to the other
- Proportional contribution values
  - If I had a $100, how much would I spend on each one?
  - Percent of each branch of use case FBS



Proportional Valuation



- Stories provide an additional level of valuation

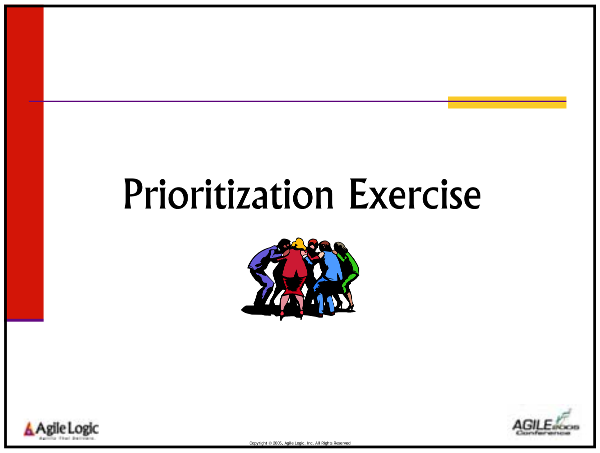


Prioritizing Using Risk

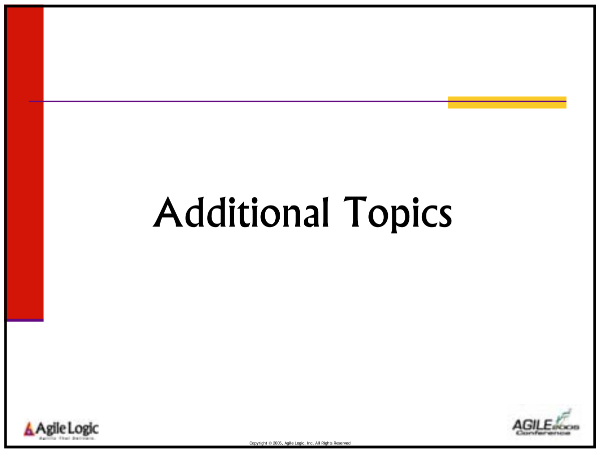
- Identify the specific risk
- Determine the probability of the loss
- Determine the cost of the loss
- Determine the cost of mitigating the risk

"We have a 30% risk that our database can't handle the expected volume. If that happens, we will lose at least 25% of our customer revenue of $80K/month as they give up on booking tours. We also will have to spend about 15 developer days to rework the persistence framework. We can spend 4 days now to test if this is true before we develop with the current database."

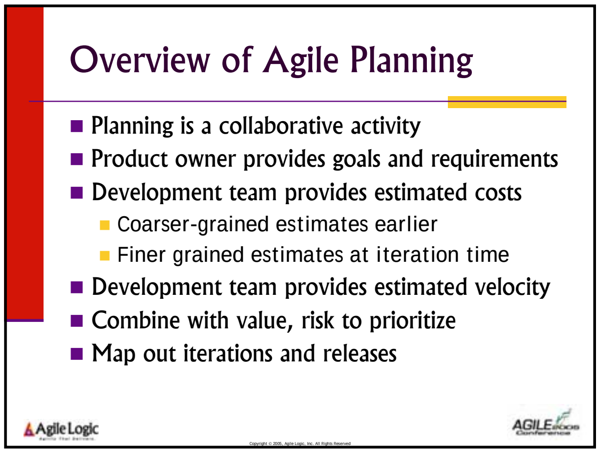
- Often we have to work with estimated values
- Importance is on discussing the risk



Prioritization Exercise

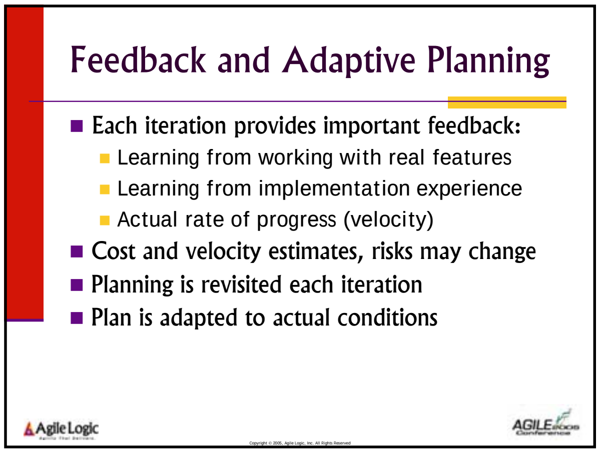


Additional Topics



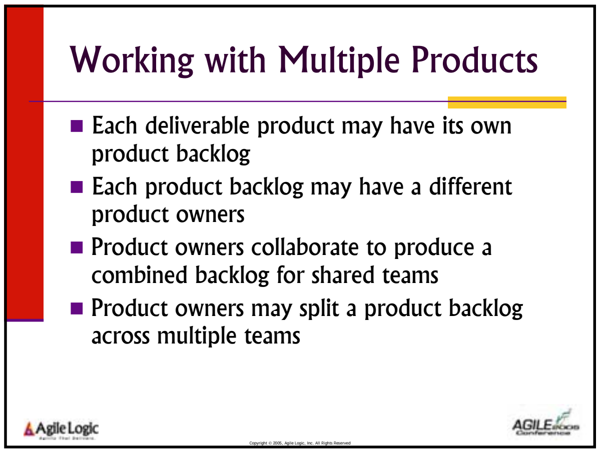
Overview of Agile Planning

- Planning is a collaborative activity
- Product owner provides goals and requirements
- Development team provides estimated costs
  - Coarser-grained estimates earlier
  - Finer grained estimates at iteration time
- Development team provides estimated velocity
- Combine with value, risk to prioritize
- Map out iterations and releases



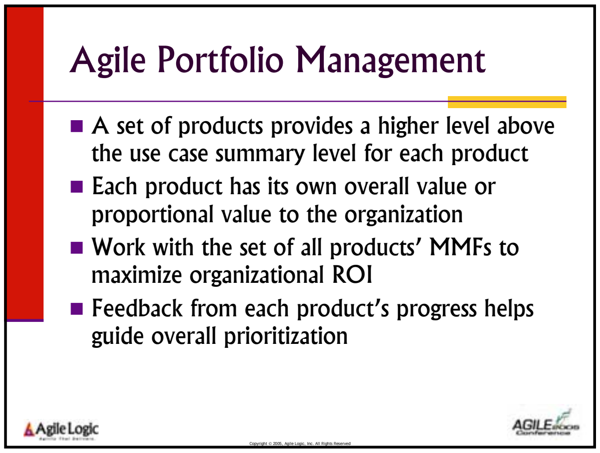
Feedback and Adaptive Planning

- Each iteration provides important feedback:
  - Learning from working with real features
  - Learning from implementation experience
  - Actual rate of progress (velocity)
- Cost and velocity estimates, risks may change
- Planning is revisited each iteration
- Plan is adapted to actual conditions



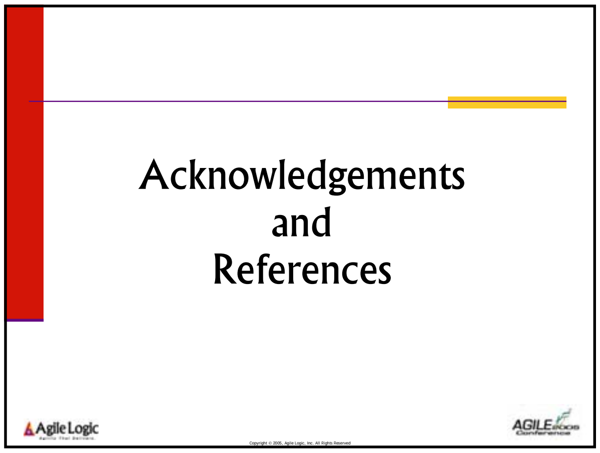
Working with Multiple Products

- Each deliverable product may have its own product backlog
- Each product backlog may have a different product owners
- Product owners collaborate to produce a combined backlog for shared teams
- Product owners may split a product backlog across multiple teams

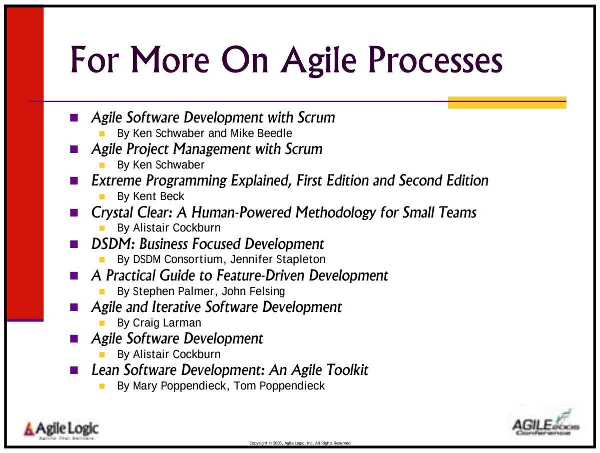


Agile Portfolio Management

- A set of products provides a higher level above the use case summary level for each product
- Each product has its own overall value or proportional value to the organization
- Work with the set of all products' MMFs to maximize organizational ROI
- Feedback from each product's progress helps guide overall prioritization

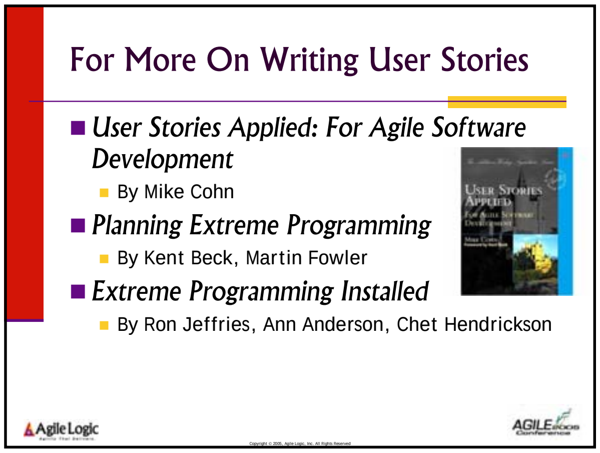


Acknowledgements and References



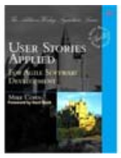
For More On Agile Processes

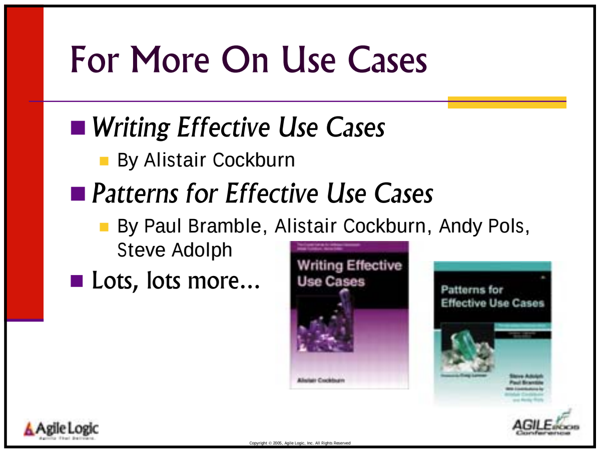
- *Agile Software Development with Scrum*
  - By Ken Schwaber and Mike Beedle
- *Agile Project Management with Scrum*
  - By Ken Schwaber
- *Extreme Programming Explained, First Edition and Second Edition*
  - By Kent Beck
- *Crystal Clear: A Human-Powered Methodology for Small Teams*
  - By Alistair Cockburn
- *DSDM: Business Focused Development*
  - By DSDM Consortium, Jennifer Stapleton
- *A Practical Guide to Feature-Driven Development*
  - By Stephen Palmer, John Felsing
- *Agile and Iterative Software Development*
  - By Craig Larman
- *Agile Software Development*
  - By Alistair Cockburn
- *Lean Software Development: An Agile Toolkit*
  - By Mary Poppendieck, Tom Poppendieck



For More On Writing User Stories

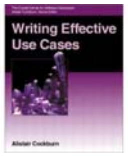
- *User Stories Applied: For Agile Software Development*
  - By Mike Cohn
- *Planning Extreme Programming*
  - By Kent Beck, Martin Fowler
- *Extreme Programming Installed*
  - By Ron Jeffries, Ann Anderson, Chet Hendrickson





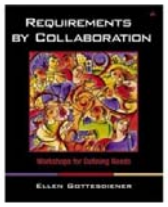
For More On Use Cases

- *Writing Effective Use Cases*
  - By Alistair Cockburn
- *Patterns for Effective Use Cases*
  - By Paul Bramble, Alistair Cockburn, Andy Pols, Steve Adolph
- Lots, lots more…



For More on Collaborative Workshops

- *Requirements by Collaboration: Workshops for Defining Needs*
  - By Ellen Gottesdiener



For More on Valuing and Prioritizing Requirements

- *Software by Numbers: Low-Risk, High-Return Development*
  - By Mark Denne, Jane Cleland-Huang

Software by Numbers
Low-Risk, High-Return Development
Mark Denne · Jane Cleland-Huang

Thank You for Attending!

Enjoy the Conference!