



The Agile Process

Jeffery Payne
CEO, Coveros, Inc.
jeff.payne@coveros.com



Agenda

- Introductions & Expectations
- What is Agile?
- Why does Agile work?
- Myths about Agile
- Agile development process
- Wrap Up

- Coveros helps organizations accelerate the delivery of secure, reliable software
- Our consulting services:
 - Agile software development
 - Application security
 - Software quality assurance
 - Software process improvement
- Our key markets:
 - Financial services
 - Healthcare
 - Defense
 - National security

Corporate Partners



Introductions

Instructor – Jeffery Payne

Jeffery Payne is CEO and founder of Coveros, Inc., a software company that helps organizations accelerate the delivery of secure, reliable software. Coveros uses agile development methods and a proven software assurance framework to build security and quality into software from the ground up. Prior to founding Coveros, Jeffery was Chairman of the Board, CEO, and co-founder of Cigital, Inc. Under his direction, Cigital became a leader in software security and software quality solutions, helping clients mitigate the risk of software failure. Jeffery is a recognized software expert and popular speaker at both business and technology conferences on a variety of software quality, security, and agile development topics. He has also testified before Congress on issues of national importance, including intellectual property rights, cyber-terrorism, Software research funding, and software quality.

Class Attendees

Expectations

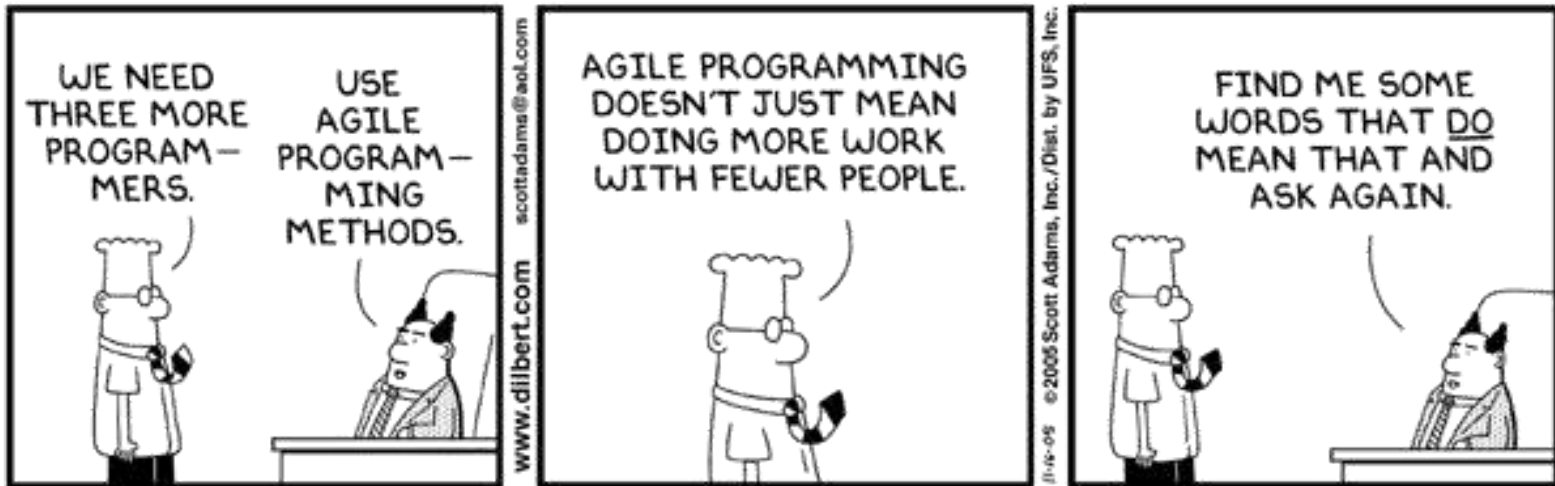
- What are your expectations for this class?
- What do you wish to learn?
- What questions do you want answered?

Objectives

The primary objectives of this course are to:

- Introduce you to Agile software development
- Discuss the major differences between Agile and traditional methodologies.
- Describe how Agile practices and principles improve the software development process.

What is Agile?

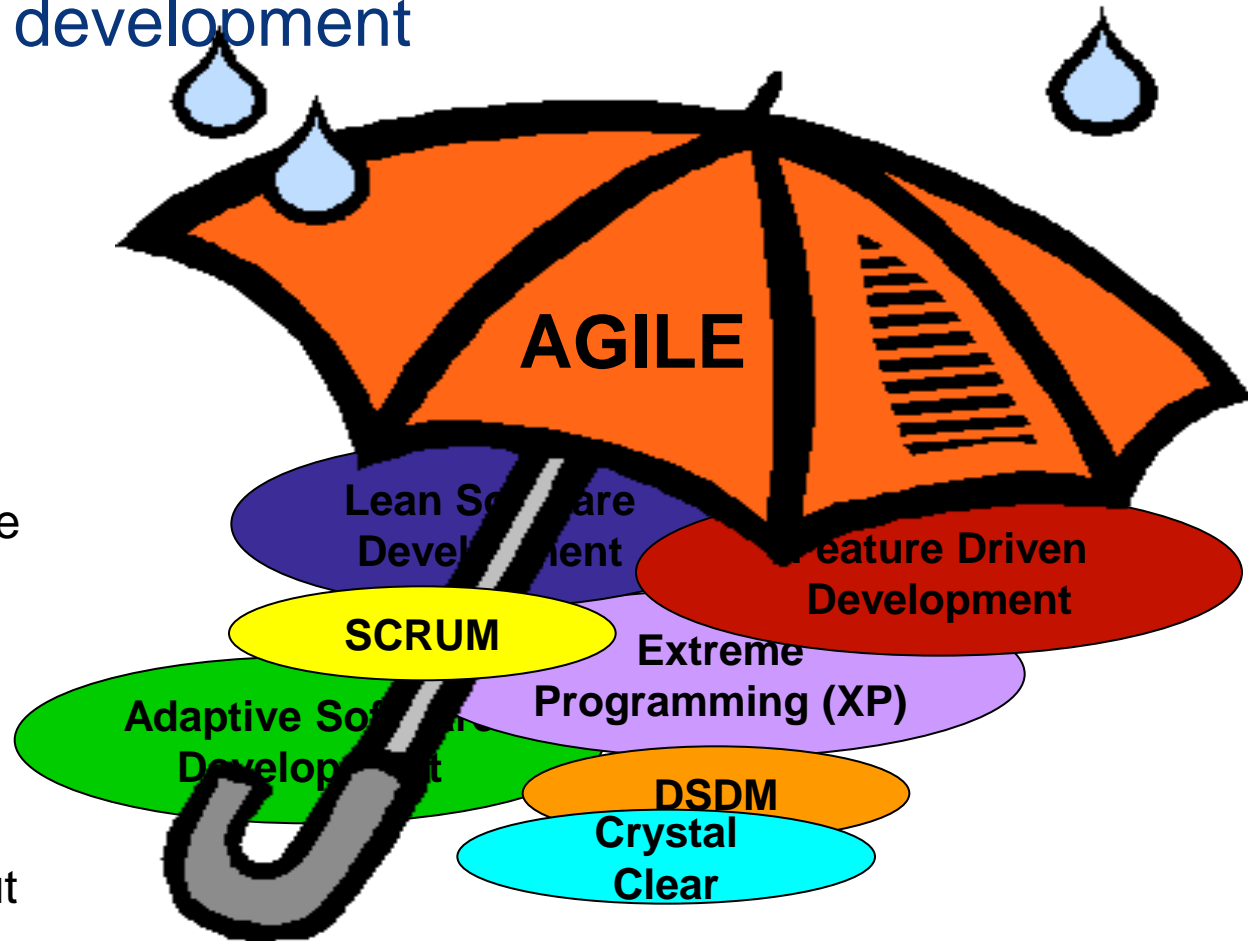


© Scott Adams, Inc./Dist. by UFS, Inc.

What is Agile?

The agile movement began as a set of ideas for improving software development

- Close collaboration between programmers & business people
- Face-to-face communication
- Frequent delivery of deployable business value
- Self-organizing teams
- Crafting code & environment to support requirements changes
- The most important output of a project is working software



<http://www.agilemanifesto.org>

Different agile methodologies emphasize different practices

Scrum

- Product Backlog
- Sprint Backlog
- Daily Scrum
- Sprint Review
- Self-Directed Teams
- Chickens and Pigs

XP

- Test-Driven Development
- Refactoring
- Simple Design
- Pair Programming
- Collective Ownership
- Coding Standard
- Sustainable Pace
- Metaphor
- Continuous Integration
- The Planning Game
- Small Releases
- On-Site Customer

Lean

- Seeing waste
- Value stream mapping
- Set-based development
- Pull systems
- Queuing theory
- Motivation
- Measurements

DSDM

- Timeboxing
- Meta Modeling
- MoSCoW Method

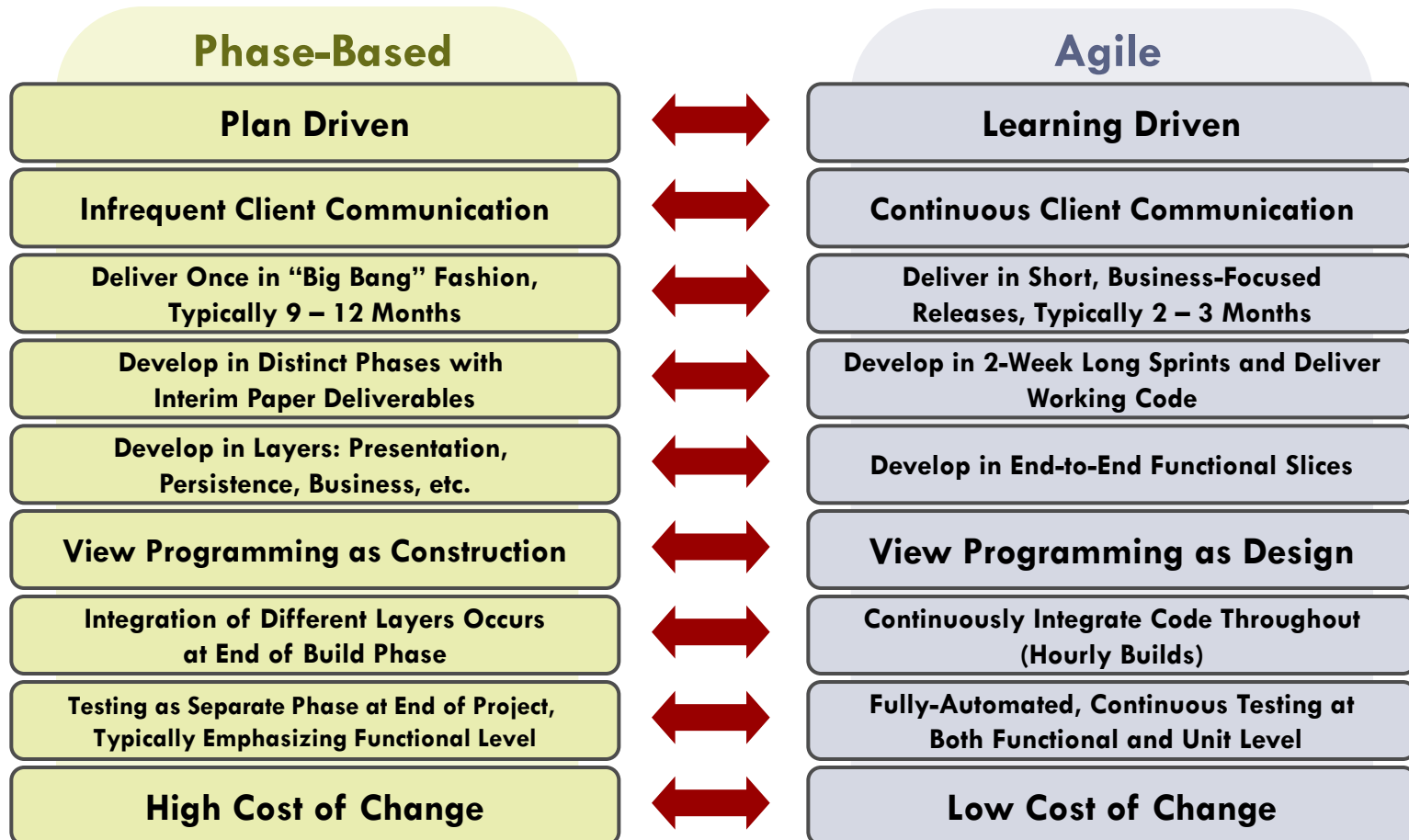
Crystal

- Reflective Improvement
- Osmotic Communication
- Easy Access to Expert Users

All agile methodologies adhere to some basic principles

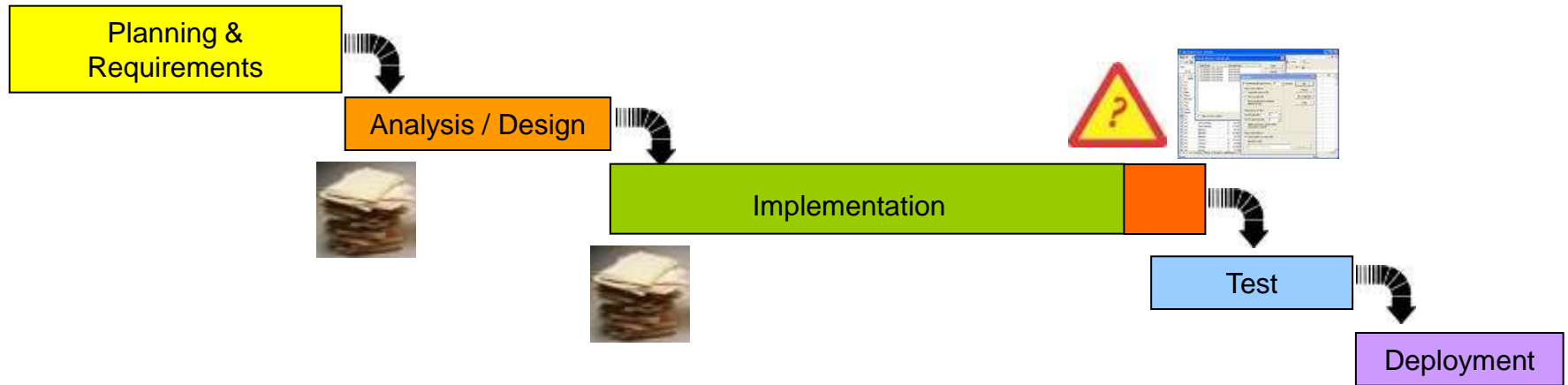
- Early and continuous delivery of valuable software
- Welcome changing requirements, even late in development
- Deliver working software frequently
- Business people and developers work together daily
- Build projects around motivated individuals and trust them to get the job done.
- Frequent conversation to convey information efficiently
- Working software as the primary measure of progress
- Sustainable development
- Continuous attention to technical excellence and good design
- Simplicity—maximizing the amount of work not done
- The best architectures, requirements, and designs emerge from self-organizing teams
- At regular intervals, the team reflects on, tunes, and adjusts its behavior

The agile approach is a different way of *thinking* about SW projects

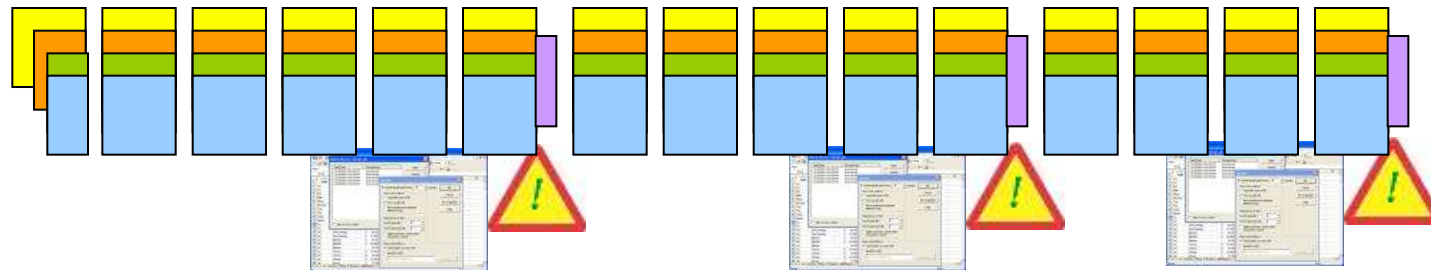


Agile Rearranges Key Development Activities

Phase-Based approach

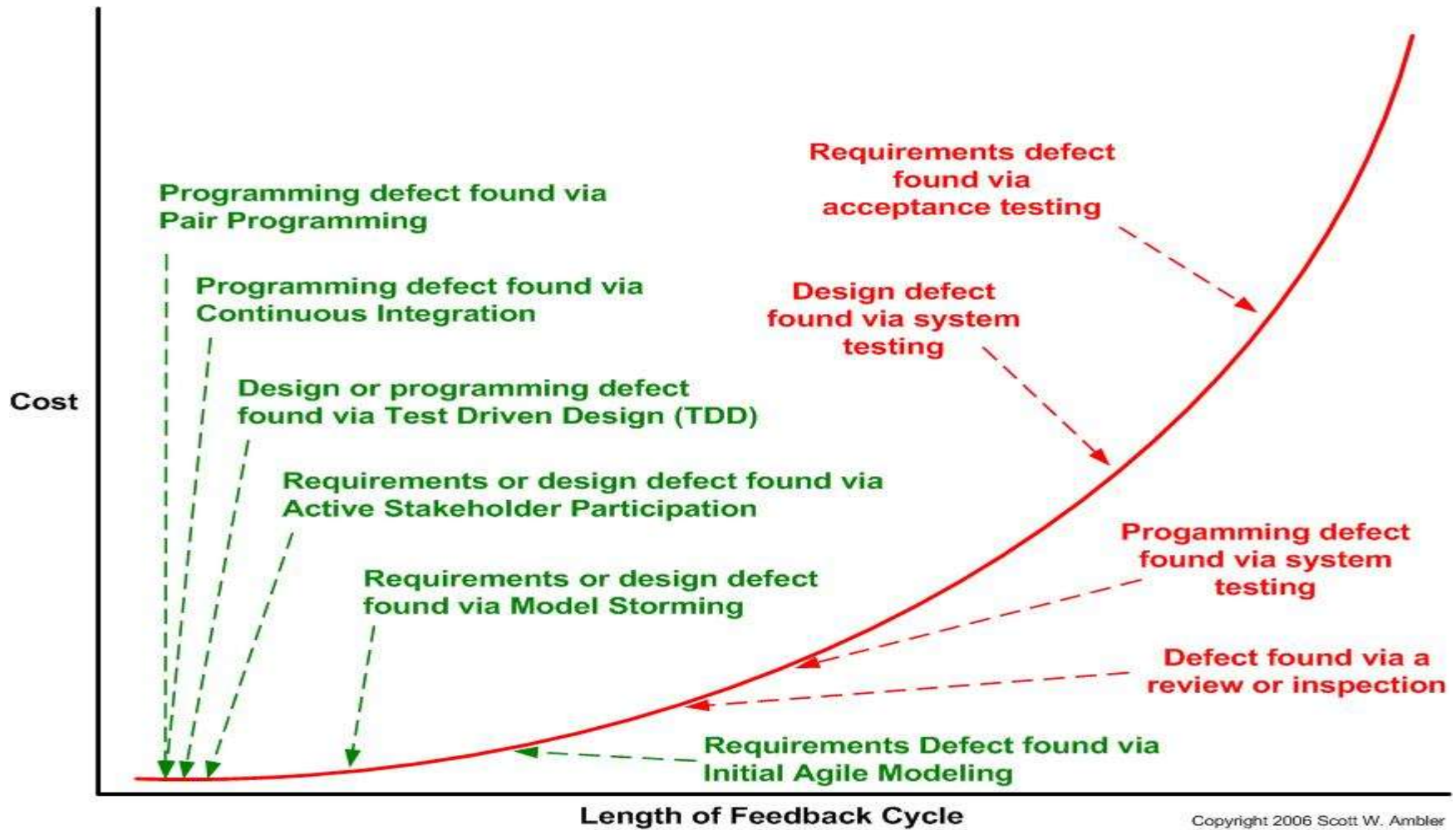


Incremental/Agile approach



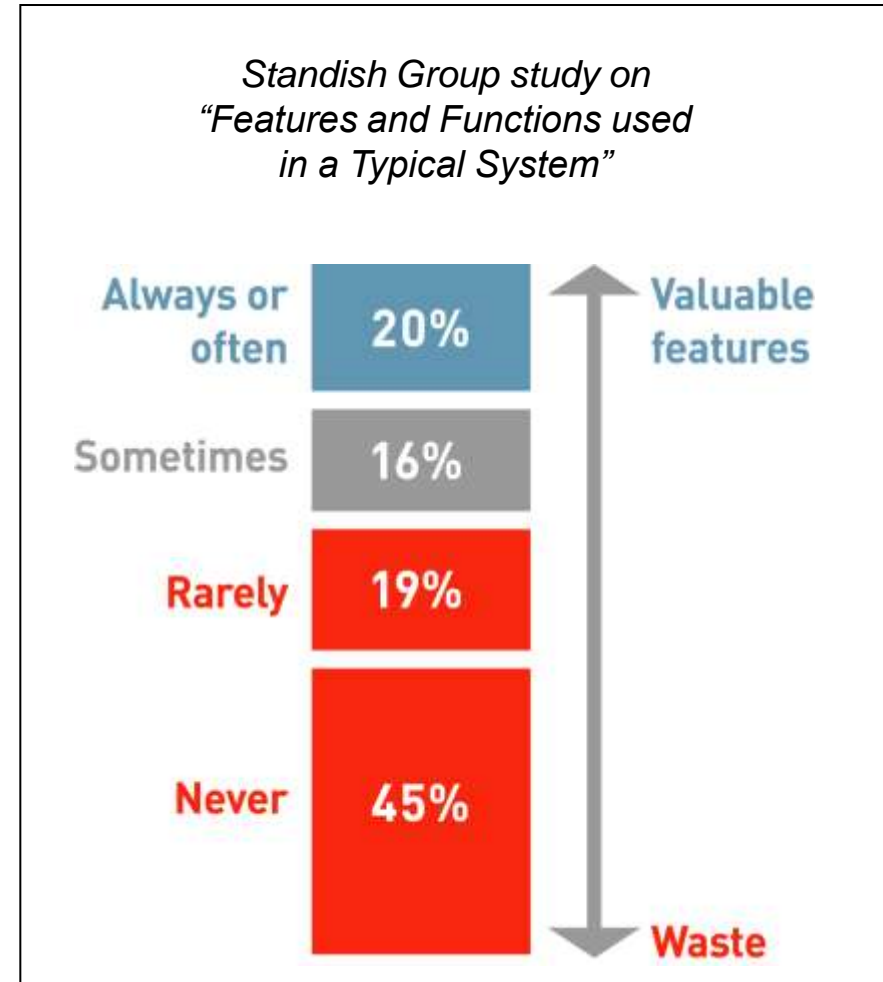
Note: Agile is not simply a set of “small waterfalls”

Why Projects Fail: Cost of change

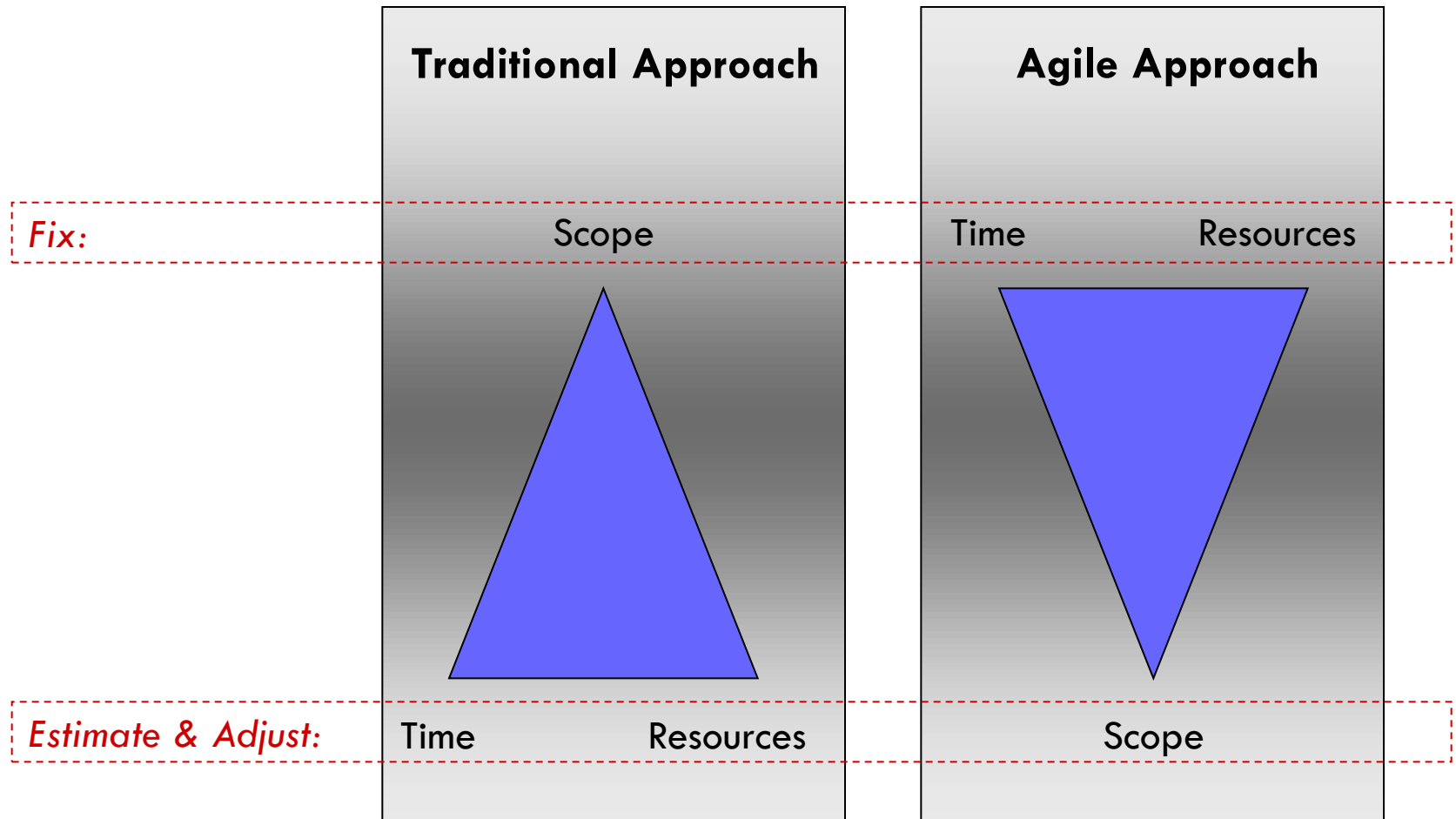


Why Projects Fail: Poor requirements management

Much of present-day software acquisition procedures rests upon the assumption that one can specify a satisfactory system in advance. . . . I think **this assumption is fundamentally wrong**, and that many software acquisition problems spring from that fallacy. --Fred Brooks, 1986



Addressing these Issues with Agile Planning



Addressing these Issues with Agile Software Delivery

- Deliver incremental change in order to maximize feedback.
- Accept change continuously in order to minimize waste.

THE AGILE BET

If the cost of change can be kept low over time, the cost savings that result from **early feedback will far outweigh** the added costs of early change.

Myths about Agile

- There are many myths floating around about Agile Development
- These myths are often due to:
 - A lack of understanding of Agile
 - Early thinking within the Agile community that proved to be wrong
 - Trying to implement Agile in a manner that will not work
 - Relying upon consultants who know the theory but can't apply it pragmatically
- Regardless, Agile is not a silver bullet that will magically transform your organization without a lot of hard work

No planning takes place on Agile projects

- Reality:
 - Agile teams spend as much, if not more, time planning development activities.
 - The major difference is that the planning is spread throughout the entire lifecycle of the project.
 - Traditional methodologies emphasize lots of upfront planning. Agile teams do some planning upfront, but only enough to understand the major milestones and dependencies.
 - Agile is designed to embrace change and uncertainty, so most planning is done in a continuous, 'just-in-time' fashion.
- Planning Pragmatics
 - Define your wish list / vision up front
 - Define an initial, high level architecture up front
 - Wireframe your user interface look and feel
 - Detail out 1-2 Sprints of work
 - Detail out additional Sprints prior to the start

No documentation is written on Agile projects

- Reality:
 - Agile emphasizes working software over comprehensive documentation
 - Agile encourages the “right” amount of documentation, that is, documents that are of value to the project and downstream maintenance
 - The creation of a document is treated as a requirement, which in turn must be estimated. This forces the team to carefully consider the costs of documentation and focus only on the development of concise, valuable documentation.
- Documentation Pragmatics
 - Document as necessary for communications
 - Document as necessary for support and maintenance
 - Document as necessary for corporate policy and/or regulatory compliance

Software testers aren't needed on Agile projects

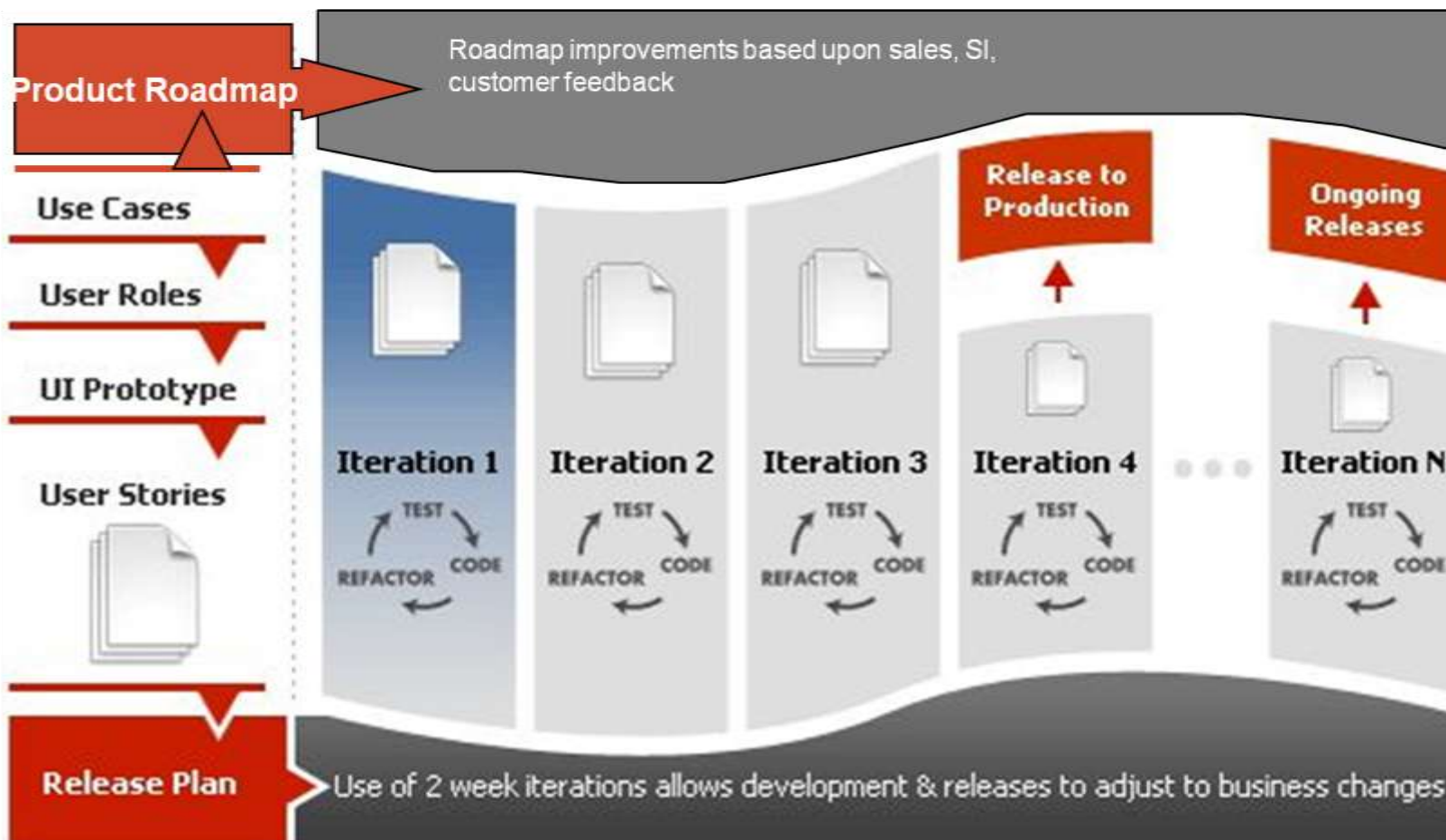
- Reality:
 - In Agile development, just like in traditional methods, the development and test team share responsibility for code quality.
 - More frequent code deployment to the test environment requires enhanced methods to ensure quality, such as test automation and functional test suites.
 - These activities require a skilled and capable test team to execute successfully.
 - Agile does rely on more automated testing and testing inline with development vs. post development so testers do test in a different way
- Testing Pragmatics
 - Unit testing by development is a necessity
 - A test automation strategy should be used to dictate where the ROI point is for automation
 - Automated testing is integrated with continuous integration to support rapid build, test, fix cycles
 - Full integration and system testing is done mostly prior to release

Agile Product Development Process



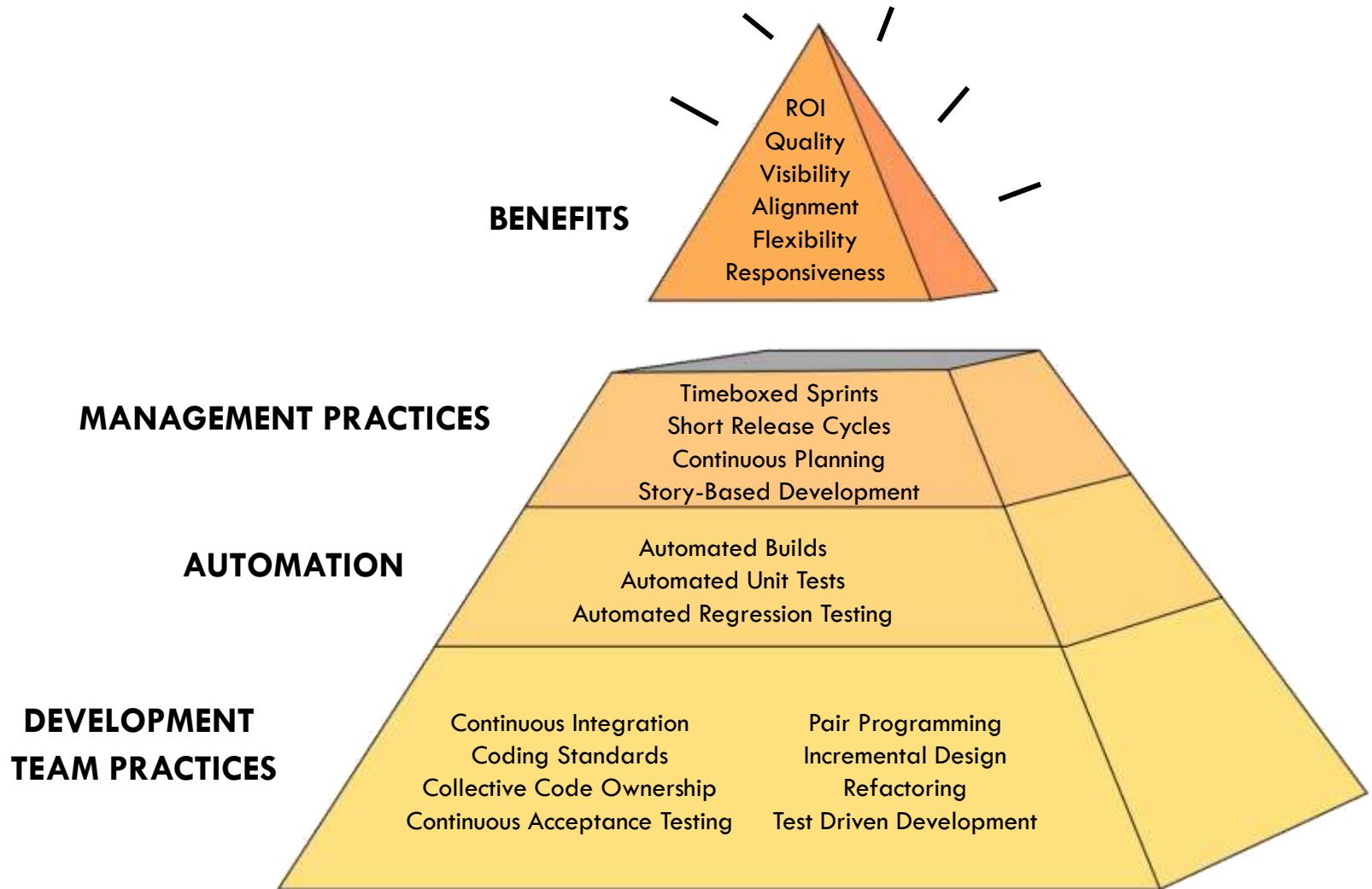
Initial Planning

On-going Planning, Implementation & Release

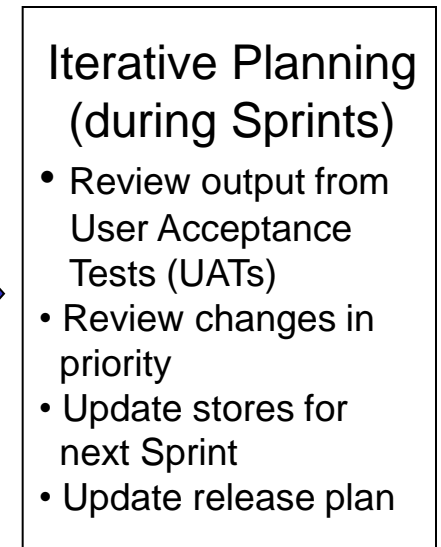
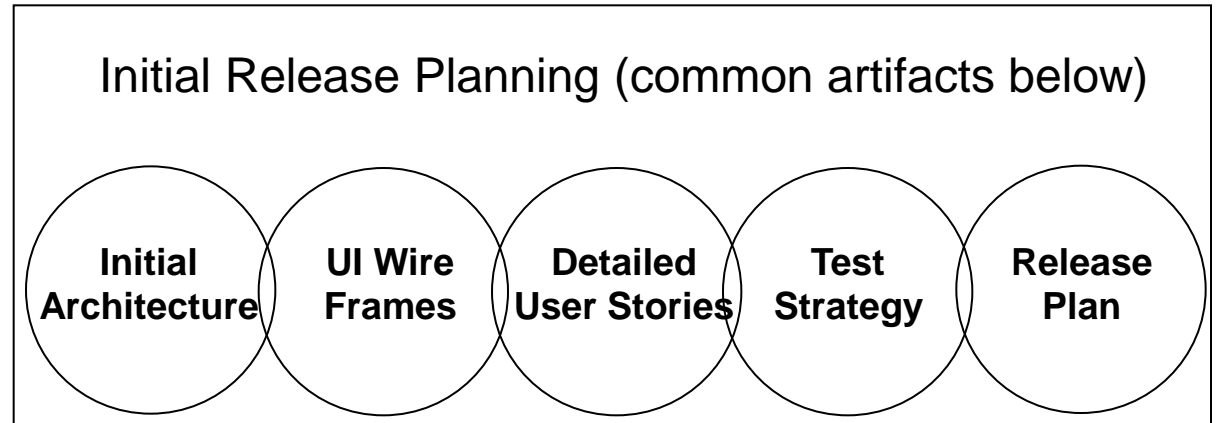
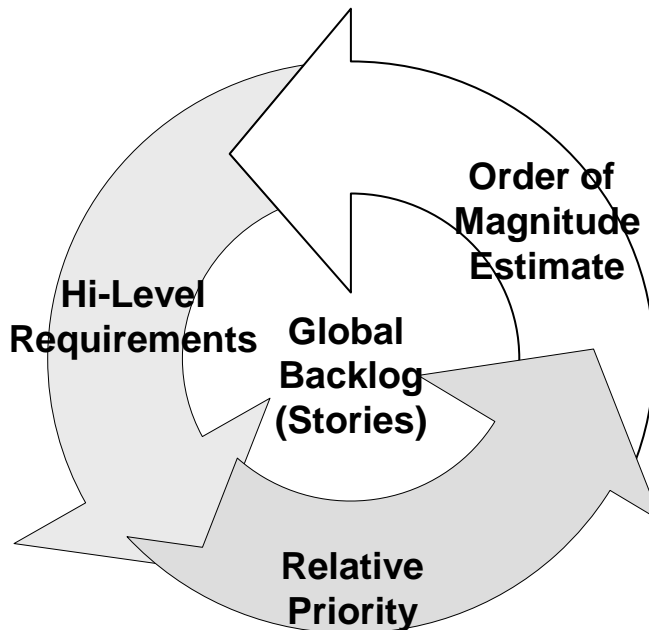


- Incremental product delivery process that encompasses all aspects of the organization
- Team-oriented with day-to-day interactions between all functions

Agile Best Practices



Agile Planning Process



Roles of Core Planning Team Members

- *Business Stakeholder(s)* – Own the product vision and helps make sure the requirements meet the needs of the end customer
- *Project Manager* – Responsible for the end-to-end planning process
- *Lead Architect* – Responsible for the initial architecture and scoping
- *Lead Business Analyst* – Acts as SME and documents requirements
- *QA Lead* – Responsible for overall test strategy
- *Web Designer* – Optional depending upon whether UI prototyping is involved

Others participate in the process as required and necessary

Creating a Product Wish List

- A 'Wish List' is a list of all possible features & functions that a particular product might encompass over time
- Inputs into the Wish List should come from everywhere within and outside of the organization that has a stake in the product
- Organization's often encompass a Wish List within a Product Vision document
- Product management typically owns the Wish List

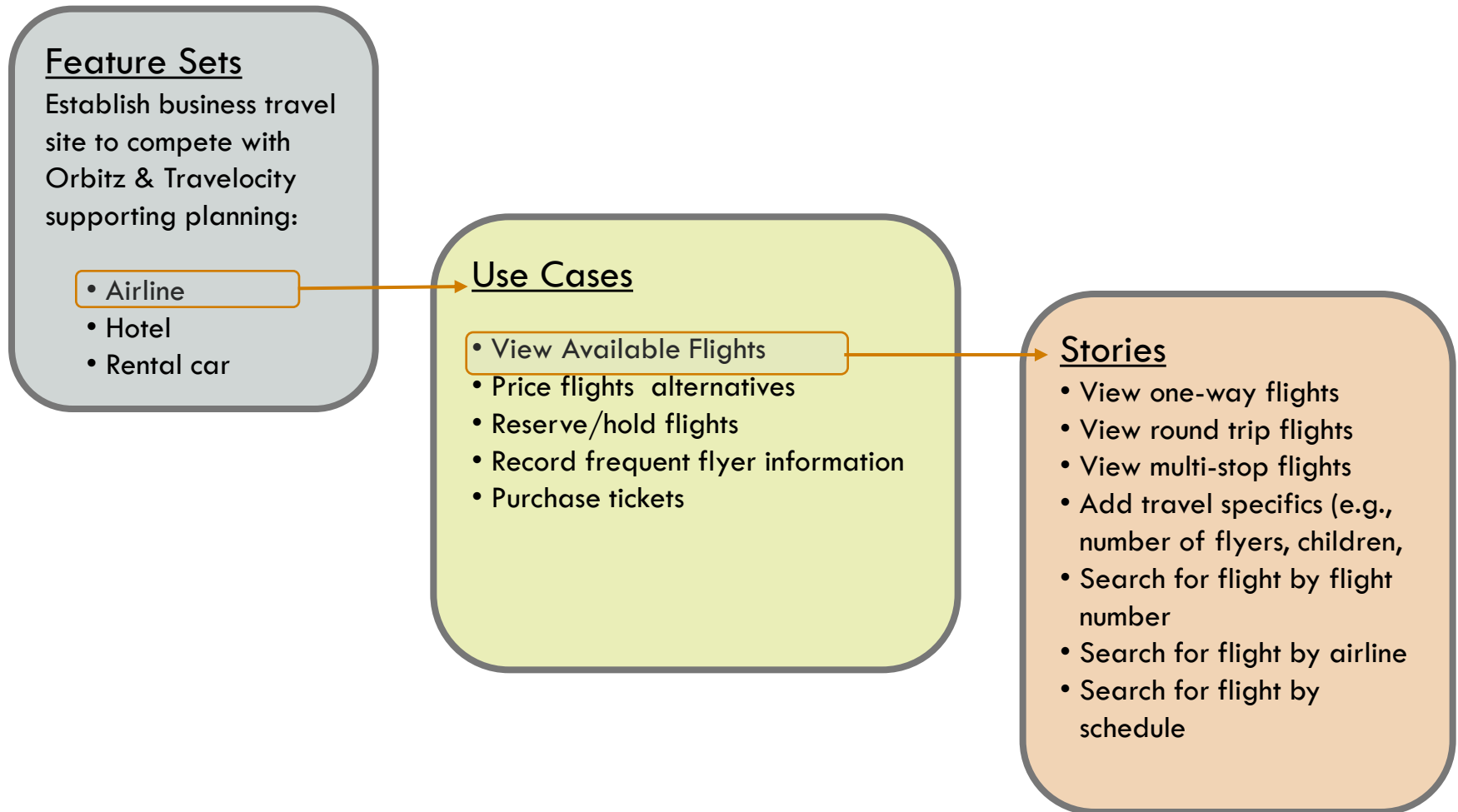
Creating a Global Backlog

- A Global Backlog encompasses all items from the Wish List that Product Management deems the highest priority for inclusion in upcoming releases.
- Backlog items are prioritized and assigned an Order of Magnitude (OOM) estimate
- OOM can be used for budgeting purposes **BUT ONLY IF THE TOP END VARIANCE ESTIMATE IS USED**
- A Global Backlog contains User Stories

User Stories

- A story represents some small slice of visible, usable functionality—typically something a user can do with the system
- A well-written story possesses the following characteristics:
 - Understandable
 - Testable
 - Valuable to the customer
 - Independent of each other
 - Small enough to build a handful each Sprint
- Stories are written during initial planning or during a Sprint planning meeting once the project has begun.
- Although the idea for a story will most likely originate from the business stakeholders, many team members may have a hand in authoring the story card, including project managers, tech leads, analysts, and testers.

Transforming Feature Sets into User Stories



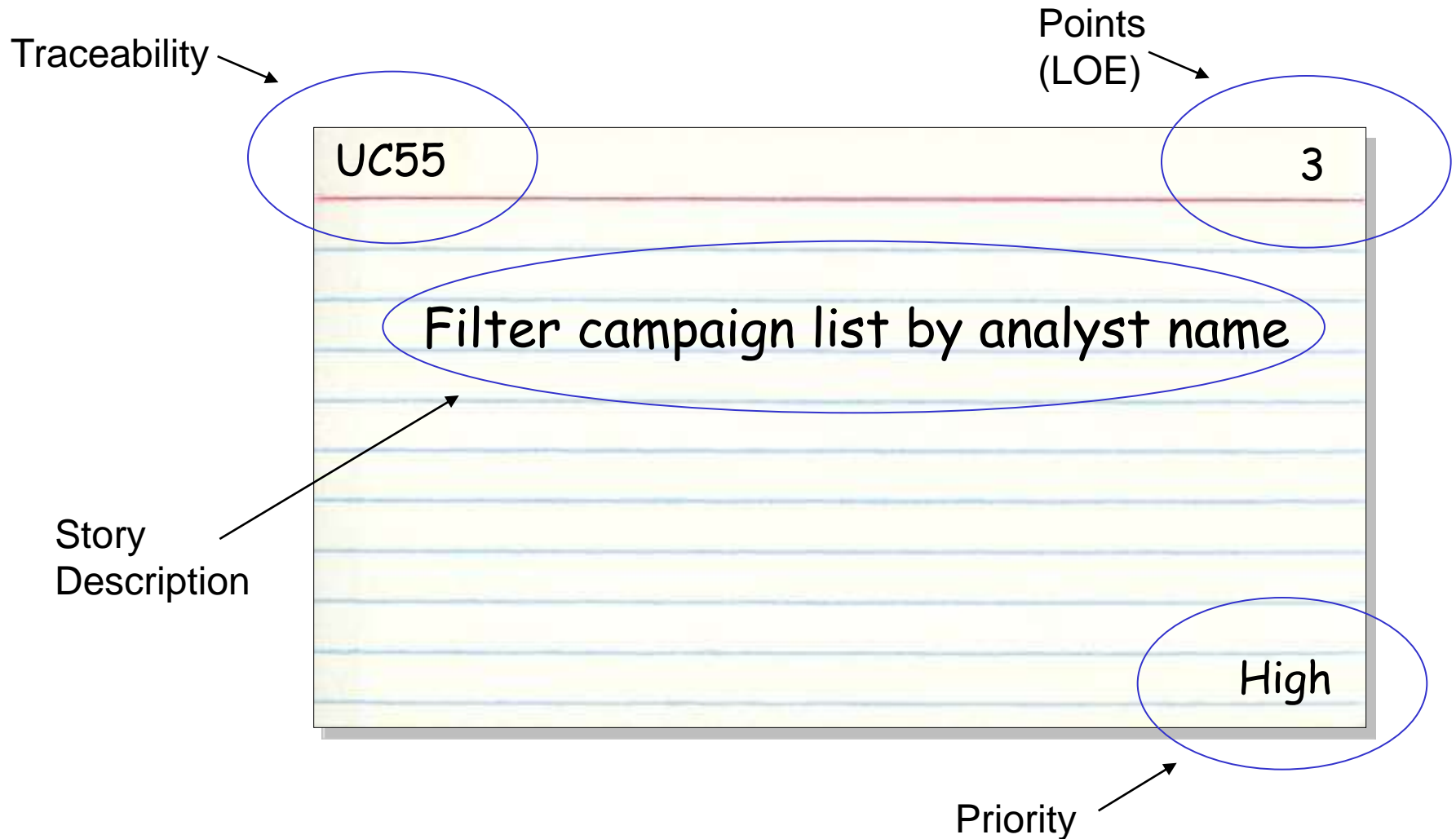
Initial Release Planning

- Planning within Agile is iterative
- Regardless, some planning must be done up front for a variety of reasons:
 - Build a release plan the organization can plan around
 - Resolve upfront architectural tradeoffs so implementation conforms to an overall architectural vision
 - Prototype / wireframe a UI to get early feedback from stakeholders on the requirements
 - Prepare development & testing platforms accordingly
- Detail out initial Sprint(s) and projected releases for more detailed budgeting purposes

Am I Ready to Begin?

- Is there a clear understanding of the reasons that the desired software is being developed?
- Is there an understanding of constraints under which the delivery team will have to work?
- Do the product owners have a clear vision of the desired software down to the theme or module level?
- Are the product owners and/or stakeholders identified and given full authority to make decisions on the tactical direction of the software to be developed?
- Are dependencies on people, processes or systems well understood?

Detailed User Stories



Helpful tips for user stories










1. Think in terms of inputs & outputs
 - What data does a person/system put in?
 - What data comes out?
 - **How will we test this?**

2. Think in terms of vertical slices
 - What is a minimal version of the desired functionality?
 - Can you exercise multiple layers of the system?
 - Be careful with “user views ...” stories

3. Don't worry too much about getting it right
 - It's OK to rip up a card and start over
 - You will get many chances to look at a story

Example User Stories

- Which are good stories and which are not so good?

	User can use webmail.
	User views a message list with no messages.
	User views all their messages.
	System uses Log4J to log all error messages.
	Graphing and charting shall be done using Business Objects.
	User configures the number of messages displayed on the page.
	User exports their resume to Microsoft Word.
	Develop the persistence framework.
	Develop the resume view JSP.

Story Estimation using Points

- A point is a unit of measurement that is used to communicate the level of effort of a user story.
- A point is equal to one day of development/test time for a single developer/tester
- As every developer is different (level of experience, skill set, etc.), you must assign points based upon the “average” throughput of your team
- Velocity is the total points that can be complete in one Sprint

Determine Velocity for Sprints

1. Determine the Maximum Velocity (MV)

- $MV = (\text{Sprint duration} - 2) \times \text{number of developers}$
- Sprints are reduced by two days to account for Kickoff and UAT

2. Determine the Realistic Maximum Velocity (RMV)

- $RMV = MV * \text{velocity multiple}$
 - Velocity multiple accounts for hours spent on overhead, reviews, vacation plans, all-hands, staff meetings, etc.
 - Velocity multiple is typically between .5 and .8 depending upon the organization
- Determine Velocity for Sprints
 - Assume a ramp-up as teams get acclimated
 - Typically use 50% of RMV for first Sprint Velocity, 75% of RMV for second Sprint Velocity, and 100% of RMV for all remaining Sprints

Velocity Calculation Example

- Assumptions
 - Two week Sprints
 - 4 Developers
 - High overhead organization
- Maximum Velocity = $(10 - 2) \times 4 = 32$ Points
- Realistic Maximum Velocity = $32 \times .5 = 16$ Points
- Sprint Velocity
 - 1st Sprint = $16 \times 0.5 = 8$ Points
 - 2nd Sprint = $16 \times .75 = 12$ Points
 - 3rd Sprint = $16 \times 1.0 = 16$ Points

Building a Release Plan

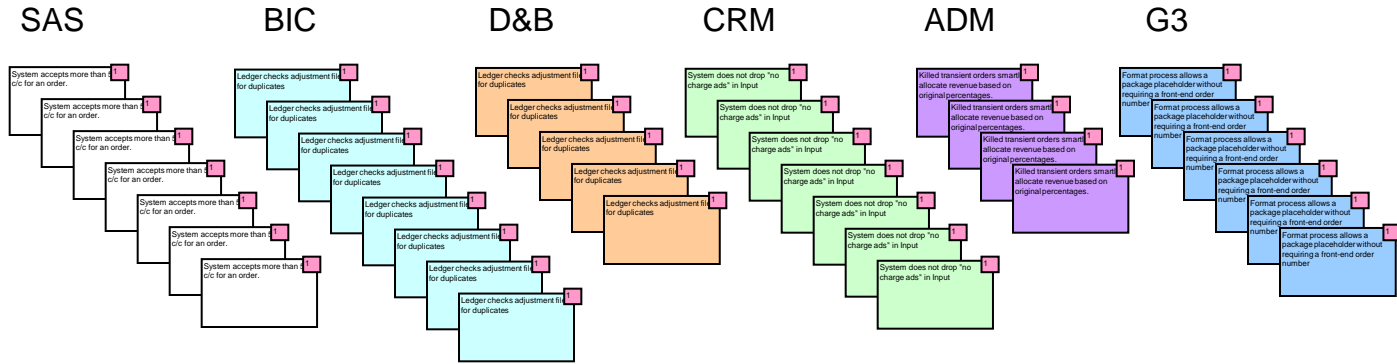
- Release plans that include Sprints* are built using:
 - Prioritized stories that include Points estimates
 - Team size
 - A decision around Sprint duration
 - A decision around how much functionality is enough to justify a release
- Sprint duration
 - Tradeoff between cost of change and organization's agility
 - Typically 2 – 4 weeks in duration
- Release decision
 - Tradeoff between cost of deployment/release and market dynamics
 - Releases vary from daily to 3 months (huge variation!)
 - Releases are now a business decision

**A Sprint is a development iteration in SCRUM terminology*

Building a Release Plan



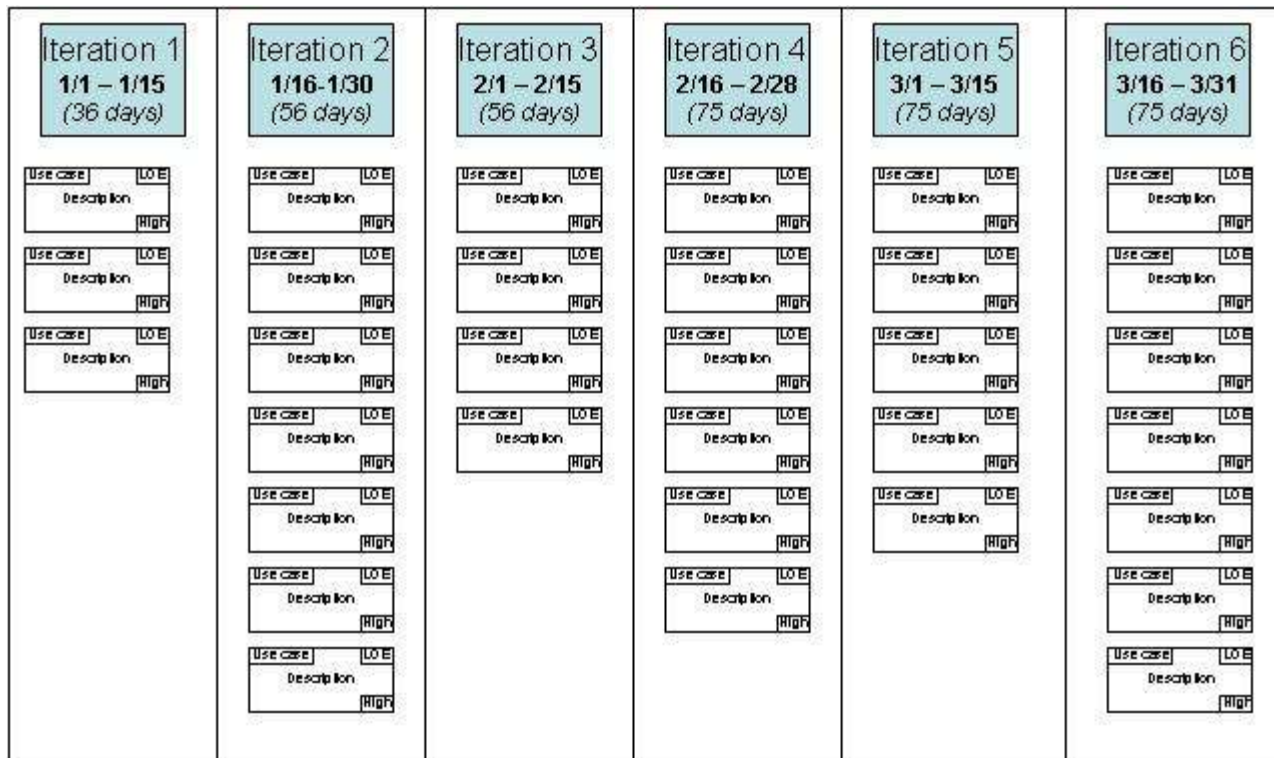
Story "Backlog"



= team velocity

	Sprint 302	Sprint 303	Sprint 304
SAS			
BIC			
D&B			
CRM			
ADM			
G3			
Total Points	42	42	42

User Stories by Sprint



The Team Room Approach to Release Plan Mgmt



Pros – very visible and tangible, great for co-located teams, easy to modify

Cons – not under version control, harder for distributed teams to visualize, takes space

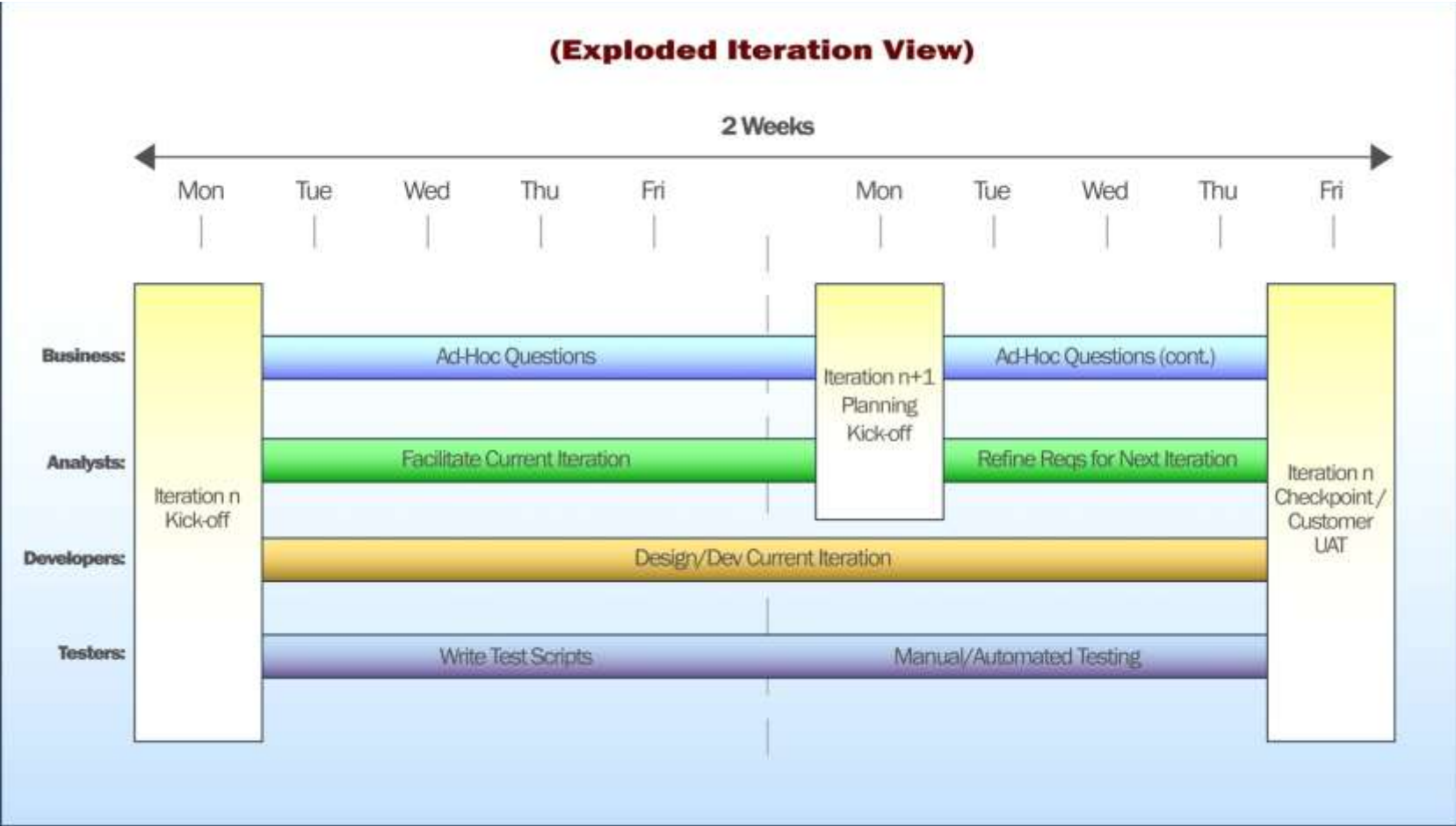
The Virtual Approach to Release Plan Management

Iteration	Story	Points
1		12
	Search for one-way flights by origin & destination	6
	Search for one-way flights by time	3
	See response from credit card processing service	3
2		15
	Allow city names for origin & destination	4
	Display complete flight information on results	5
	Validate front-end search criteria, add default values for dropdowns	2
	Spike hookup to pricing engine	1
	Search for one-way flights by date	3
3		21
	Search for round-trip flights	3
	Display credit card entry page for an itinerary	2
	Validate credit card input	2
	Submit valid CC information to CC service	5
	Display prices for flights (unknown size, plan for 6)	6
	Complete sale with MC or Visa	3
4		20
	Search for multi-stop flights	5
	Constrain search by other parameters (class, carrier, # of connections)	2
	Specify number of travelers in search	1
	Page between search results	4
	Complete sale with AmEx or Discover	1
	Generate simple report for ops	7
5		20
	Maintain sessions for 1/2 hour	1
	Generate full txn report	8
	Book flight(s) on single airline	2
	Book flight(s) on multiple airlines	3
	Put reservation on hold	2
	Add F.F. number to reservation	1
	Retrieve on-hold reservation	2
	Book on-hold reservation	1
Grand Total		88

Detail Initial Sprint(s)

- Build detailed requirements from User Stories for initial Sprint(s)
 - Typically captured in a requirements document
 - Traceable to User Stories
- Build a test plan for testing requirements and user stories
 - Define test cases and scripts
 - Test requirements and end-to-end scenarios

Iterative Development Process



Typical Roles

- Project Manager – responsible for the day to day functional delivery of the software, managing project schedule and priorities, and working with stakeholders to resolve any project issues
- Architect – responsible for coding, design and architecture standards review and compliance, solutions definition and overall performance characteristic of the software
- Analyst – supports project manager in the proper definition of requirements
- Development Lead – responsible for day to day technical implementation of the software and technical management of developers
- Developers – responsible for technical implementation of the software
- QA/Test Lead – responsible for the day to day testing, verification and validation of the software, compliance, management of the testers and automation of the test cases
- Testers – responsible for the testing, verification and validation of the software and the automation of the test cases
- Business stakeholder or proxy – available when needed to answer questions regarding the product, market, customer needs

Team Communication during Agile Development

- Effective communication between all team members is absolutely critical to a successful Agile project
- A meeting rhythm should be established to assure communication happens at least at key Sprint junctures
- Important team meetings include:
 - Sprint kickoffs
 - Daily standups
 - User acceptance testing
 - Retrospectives

A blue-tinted background image showing a man and a woman looking at a laptop screen. The man is in the foreground, and the woman is leaning over his shoulder from the right. The background is a blurred office setting with windows.

Wrap-Up

Agile books we recommend

- Beck, Kent, “Extreme Programming – Embracing Change”, Addison-Wesley Professional, 2004
- Cohn, Rob, “User Stories Applied”, Addison-Wesley Professional, 2004
- Cohn, Rob, “Agile Estimating & Planning”, Prentice Hall PTR, 2005
- Crispin, Lisa, “Agile Testing – A Practical Guide for Testers & Agile Teams”, Addison-Wesley Professional, 2009
- Duvall, Paul, “Continuous Integration: Improving Software Quality and Reducing Risk”, Addison-Wesley Professional, 2007

Questions?

- Contact information:
 - Jeffery Payne, Coveros, Inc.
 - 703-431-2920
 - jeff.payne@coveros.com