

StoryoTypes for Agile Development



Dan Rawsthorne
dan@danube.com
Danube Technologies, Inc.
<http://www.danube.com>



Who are We?

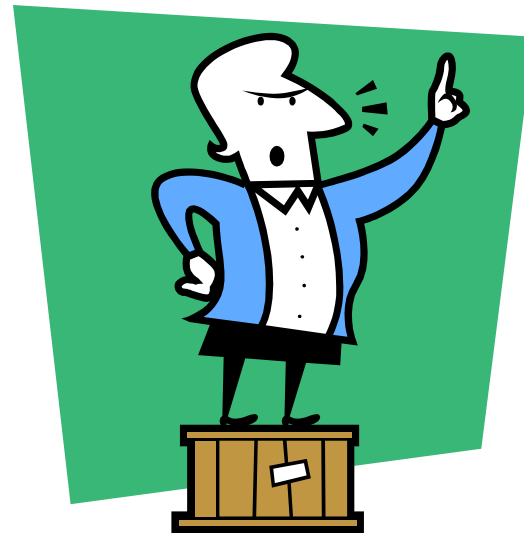
- Who is Danube?
 - On site process and technical consulting
 - ScrumWorks project management tool*
 - Agile process and technical training

- Who am I?

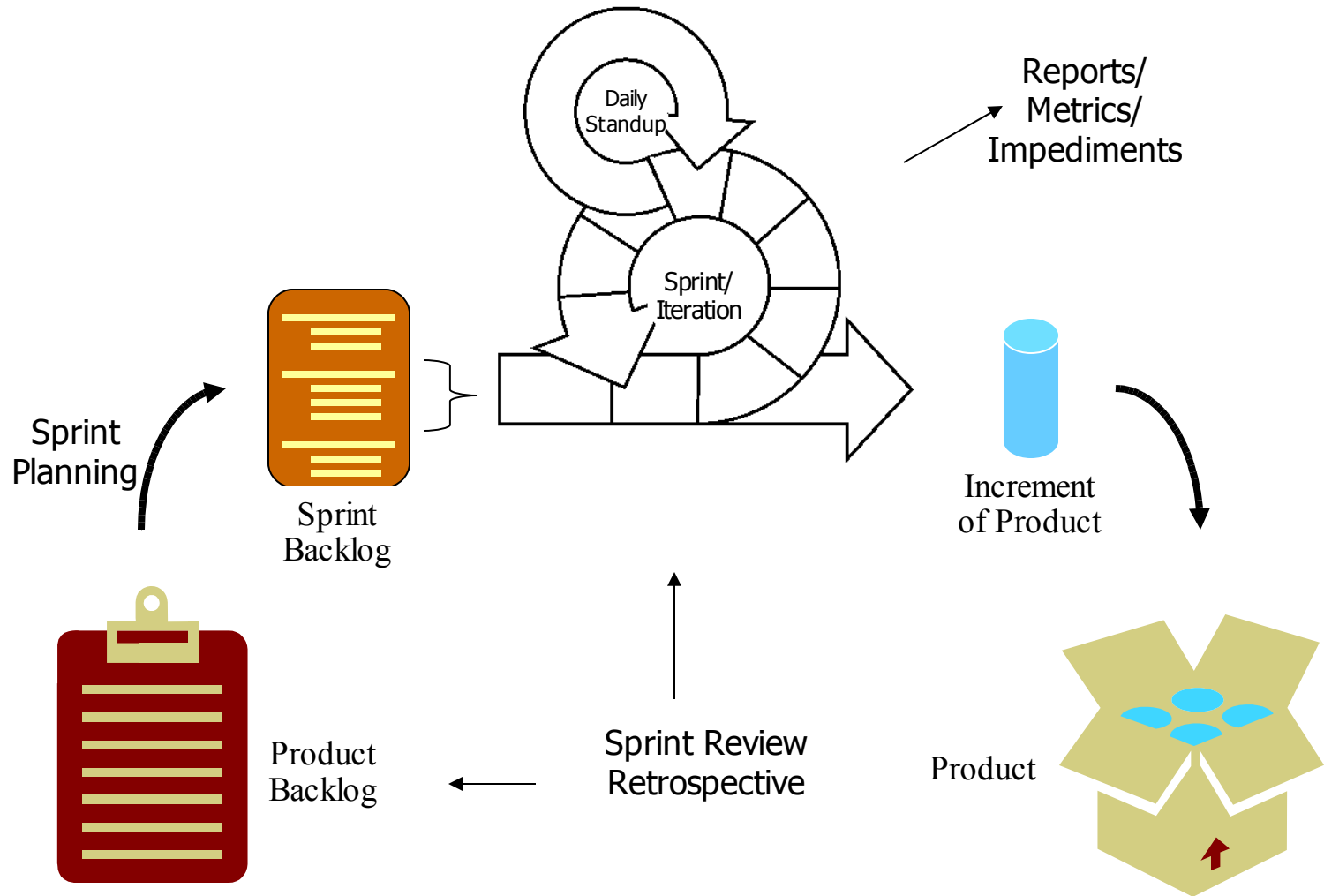
- What is this class?

* No cost download and license: <http://danube.com/scrumworks>

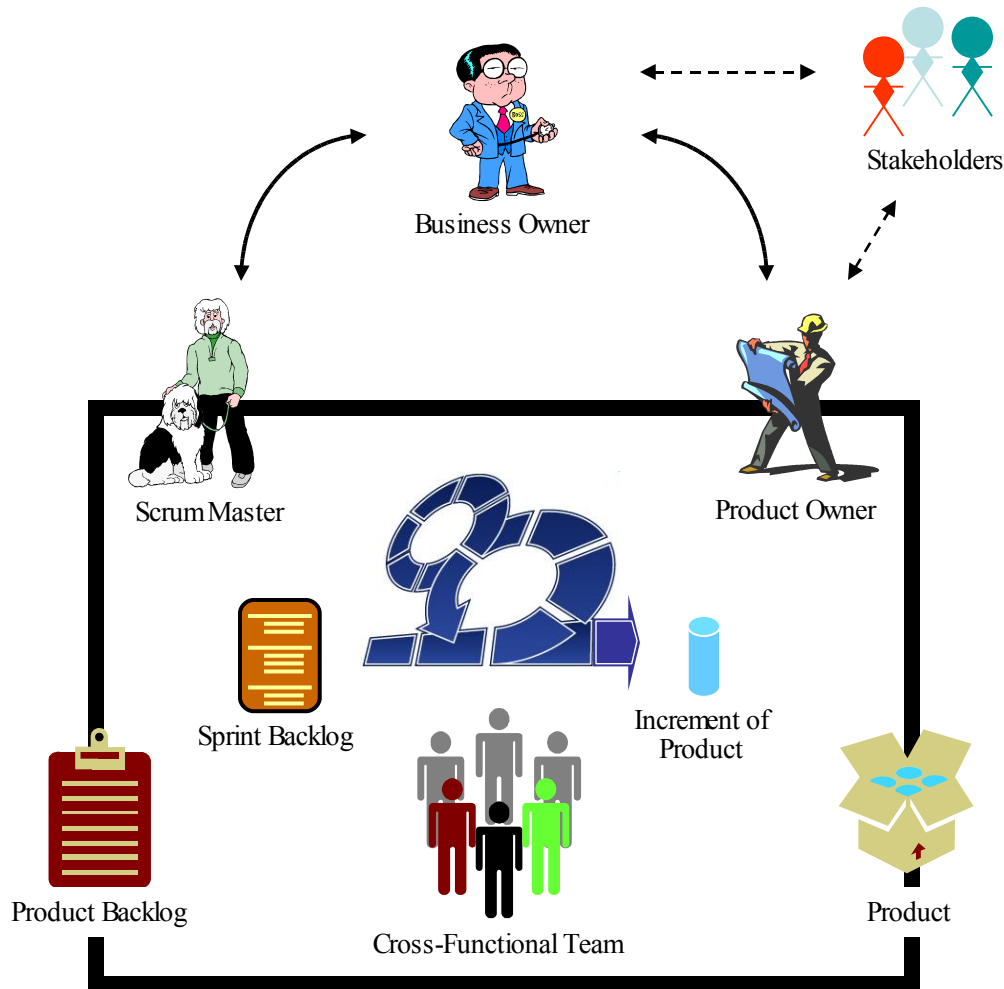
Background on Scrum



Scrum Flow Each Sprint



The Scrum Container – Responsibility



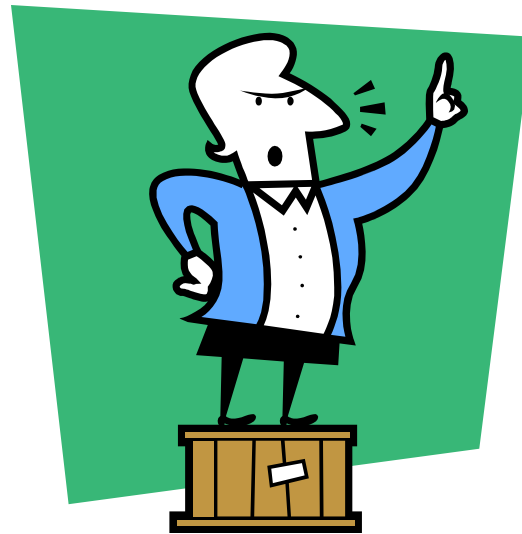
Walls of the “Container”:

- Time-boxing
- Impediment Resolution
- Cross-functionality
- Self-Organization
- Protection from “wolves”
- Clear Acceptance Criteria

This Talk Focuses on...

- The needs of large teams or organizations
 - Common code base for many small teams
 - Need for consistency of process and product
 - Need to move people around to make teams (matrix organization)
 - Formality, predictability, consistency
- However, the lessons in this talk can be used by small teams
 - Lessons learned by others

Background on Stories



What is a Story?

- A “promise for future conversations” – Cockburn
- A small unit of work that provides value
- Both a requirement and an activity
- Introduced by eXtreme Programming (XP) as a user story – something that a user needs
- Adopted by most scrum advocates – like myself – with appropriate changes

Here are my changes...

- We have many different kinds of potential stories:
 - Analysis stories
 - Development stories (user stories)
 - Infrastructure / environmental stories
 - Business Support stories
 - Etc
- So, we change the definition of story to be
 - A small unit of work that provides “project value”
 - Promise for future conversations

What about StoryoTypes?

- Gerard Meszaros gave us the word “storyotype” to mean (story stereotype) in a lovely article called “Using Storyotypes to Split Bloated User Stories,” Proceedings of the 2004 XP/Agile Universe conference
- He described storyotypes of user stories, while we describe storyotypes of each of the different categories
 - He focused on XP, we focus on Scrum

Questions So Far?



Let Me Tell You a Tale



Suppose ..

- You just won a contract to re-develop (from scratch) a web-based reservation system for an airline
- You will be onsite, co-located with your customer
 - Working in an agile environment
- What do you do?
 - Assemble a team (including customer representatives)
 - Find a Room to work in
 - And move in...

Sprint Zero

- Goals
 - Be able to write “real code” next sprint
 - Prove that you can write code at all
- Sprint Backlog (our initial stories)
 - Get some “real stuff” on the Backlog (prioritized)
 - Get an environment set up
 - Write some code, preferably real, but it doesn’t have to be
- We decompose these stories to smaller stories and tasks, and do them... let’s look at them

Backlog Looks Like (Product only)

- Buying and Reserving Flights (capability)
 - Find a Flight (epic)
 - Round-Trip Flights (?)
 - One-Way Flights (?)
 - Multi-Leg Flights (?)
 - Purchase the Flight (epic)
 - Simple Purchase a Round-Trip Flight (right-sized)
 - Choose A Seat (right-sized)
 - Etc.
 - Reserve Flight (?)
 - Purchase a Reserved Flight (?)
 - Send out e-ticket (?)
 - Update Mileage Plan information (?)
- Managing the Mileage Plan (capability)
- Hotels / Cars through Partners (capability)

One of the stories in the Backlog

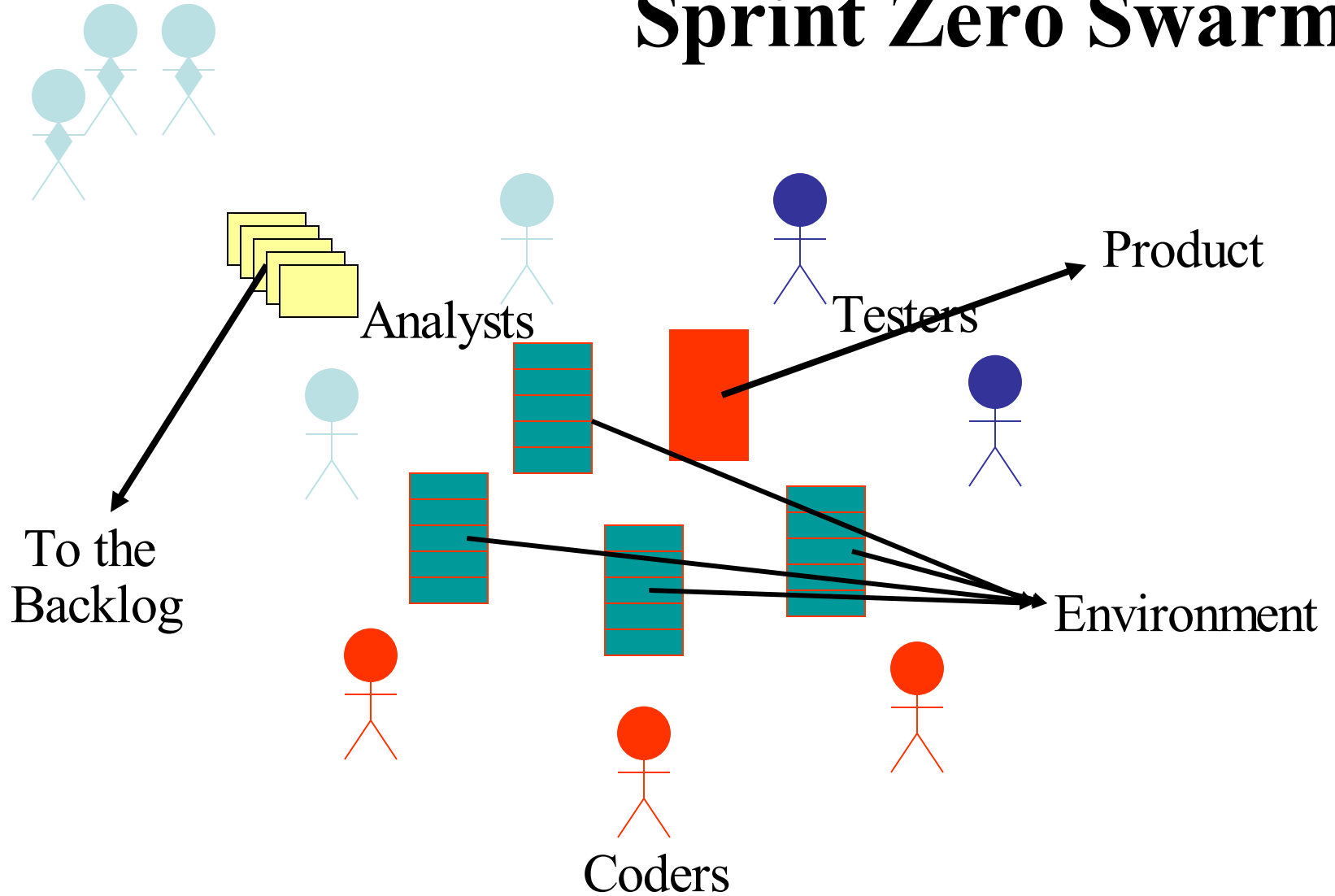
Title	Simple Purchase a Round Trip Flight
Description	As a <ticket buyer> I want <to purchase a round trip flight in the simplest possible way> so that <I can prove that the website works>
Size	Large
Done Criteria	This story will be done when: <ul style="list-style-type: none">- A user can purchase an already-found round-trip ticket through the web interface- The code has been peer reviewed- The code is protected by unit tests, that all pass in the integrated environment
Notes	<ol style="list-style-type: none">1. Purchasing is done AFTER a flight is searched for and returned, we already have a list of round trip flights in front of us to choose from2. Simplifying assumptions for story: Single leg, e.g., Seattle to Burbank and back; Single Customer, paying full fare, No luggage, pets, etc., Credit Card Payment validation is "stubbed out" and just works,3. Don't worry about a good-looking interface; just select the flight you want and go...

Get an Environment Set up

- Here is some more stuff to do, that is listed in the backlog:
 - As a <developer> I want <to have an IDE on a laptop> so that <I can actually get some work done>
 - As a <team member> I want <a configuration management system> so that <my product will be under control>
 - As a <developer> I want <an integration machine> so that <I can integrate my code with other's>
 - As a <dev lead> I want <my team to have test tools> so that <we will reduce the risk of failure>
 - As a <tester> I want <a test lab> so that <we can prove we've done what we say we have>

PO/Stakeholders

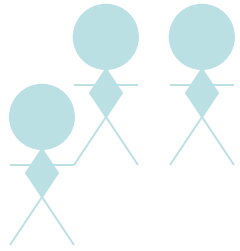
Sprint Zero Swarm



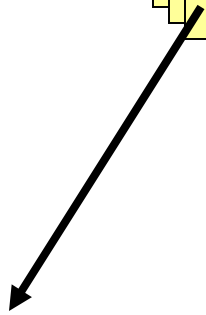
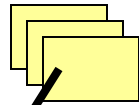
Sprint 1-n

- After Sprint 0, the following sprints are essentially the same
 - Work with the business and SMEs to create more stories and prioritize them
 - Beef up the environment, as necessary
 - Add more functionality to the system

PO/Stakeholders



Team Swarm



To the Backlog

Analysts



Testers



To The Product



Coders



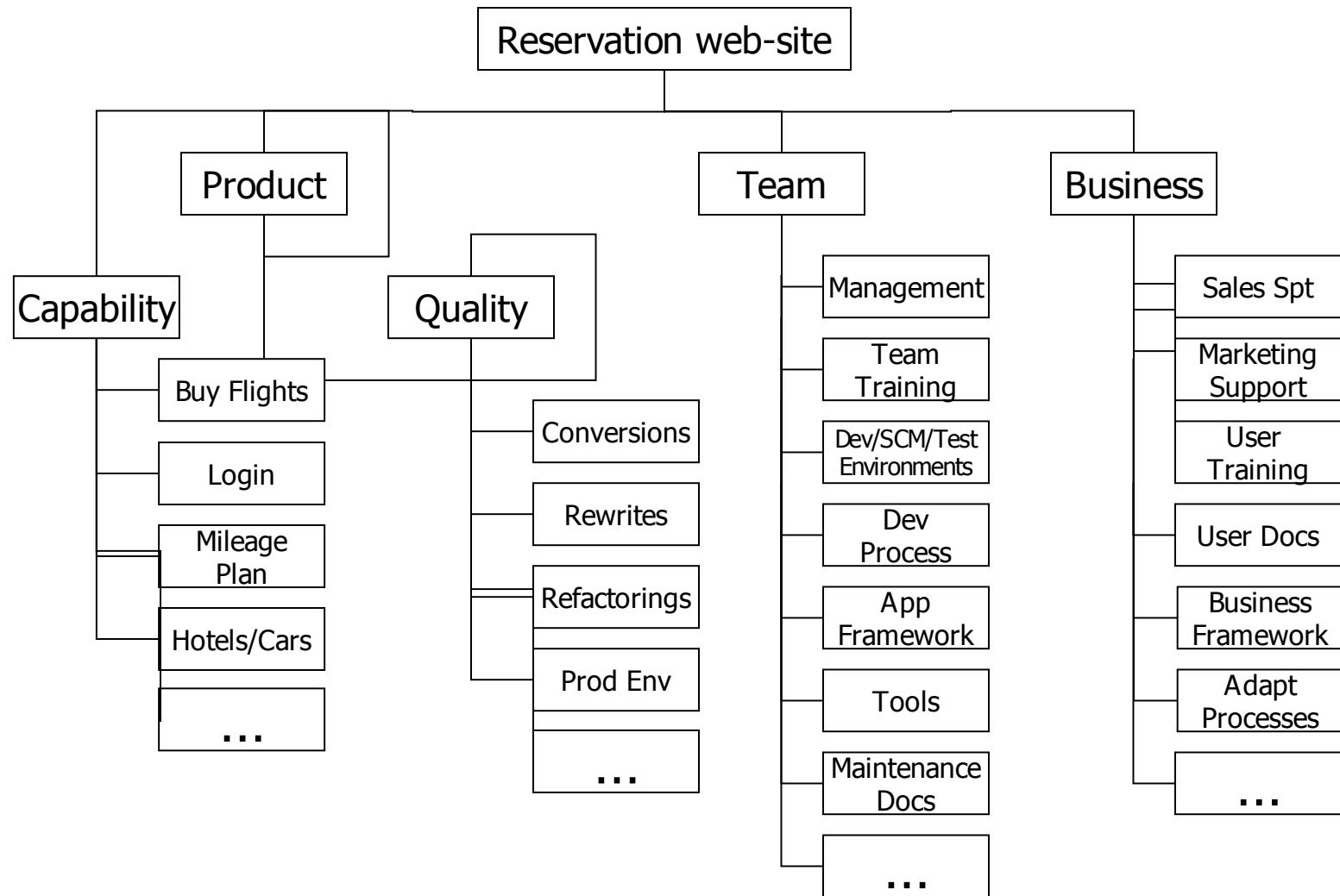
Near Release

- After a while, we have developed lots of stories against the epic “Buy and Reserve Flights” and we want to get it ready for release
- We do an interim release to QA and have them do some ad hoc testing in order to find holes in the functionality, document them as stories, and we fill “the ones that count”
- We then do a usability inspection against the product, find usability holes, fix them, and get ready to release...

What did we learn from this tale?

- This is common behavior, right?
- There is a common pattern, so we should be able to capture it
- There were common types of stories
 - Analysis stories
 - Environment stories
 - Development stories
- The Backlog is potentially complicated

Sample Backlog Structure



Different Kinds of Stories

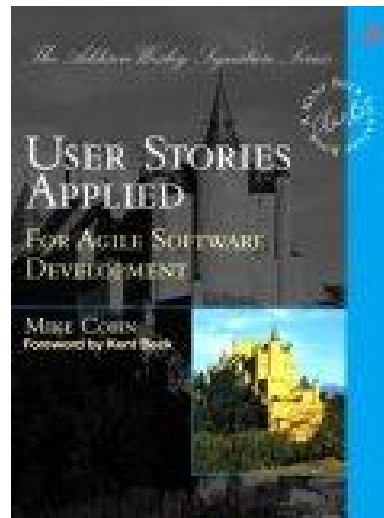
- There are different “legs” of this backlog structure
- Different legs have different kinds of stories

- This stuff is why we identify and study storyotypes...

Questions So Far?



First, a little more on Stories



Story Template

- Remember that my focus is on large teams that need some formality, so my Story Template is fairly robust, and here it is:
 - **Title** (required) – unique, descriptive, moniker for the story
 - **Description** (required) – often in Cohn form: ‘as a <> I want <> so that <>’
 - **Story Boss** (optional) – the team member responsible for managing the story to completion
 - **Origin** (optional) – the source of the story; includes its parent epic
 - **Dependencies** (optional) – other stories that depend on this one, or that this one depends on
 - **SMEs** (optional) – the people who “know about” this story
 - **Size** (required) – an estimate of effort for the story, usually represented in NUTs (Nebulous Units of Time) like Story Points, t-shirt sizes, etc
 - **Value** (optional) – a statement of how much this story is “worth”
 - **“Done” Criteria** (required) – our actual contract, our validation criteria
 - **Notes** (optional) – whatever you want to help in the conversations the story represents
 - **Tasks** (required) – eventually, you actually need to do some work...

Sample Story 1 (Epic)

Title	Shop for Flights
Description	This story represents the whole searching and shopping experience on the Airline Ticketing website.
SMEs	The following are the main SMEs for the various areas covered in this epic: <ul style="list-style-type: none">- Joe, expert on SABRE, the reservation system "in the sky" we interact with- Sam, for specials, discounts, coupons, and all things about costing- Sandra, for luggage, special needs, pets, etc
Size	Epic
Value	Essential

- Note that not all the fields are here, we just do what we need to to. We try not to be “stupid on purpose”

Sample Story 2 (Get Smart)

Title	Get Smart about Shopping for Flights
Story Boss	Joe (the analyst)
SMEs	Sam knows about this stuff...
Size	Timebox 2 days for this
Done Criteria	<p>The goals for this story are:</p> <ul style="list-style-type: none">- Identify SMEs for "Shopping for Flights" and document in epic- Meet with the SMEs and discuss the issues, document what you get in the Wiki- Generate the "skinny" version of this epic (the backbone version)- Generate at least one validated (with the SMEs) Candidate Production Story based on this backbone story
Tasks	<p>Here's a tentative to-do list:</p> <ul style="list-style-type: none">- Ask Joe who to invite- Set up a 2-hour meeting- Facilitate a discussion about shopping for flights – try to determine the main use cases, and get them in the Wiki- Dig into at least one use case to get a skinny version and an actual candidate production story, also documented in the Wiki

Sample Story 3 (Capability Backbone)

Title	Buying a Flight Backbone
Description	As a <business analyst> I want <to describe a complete (though simple) Buying a Flight experience> so that <I can get my team moving in the right direction>
Origin	Analysis of "Buying and Reserving Flights," in meeting, June 14
Size	Capability Backbone Epic
Notes	<ol style="list-style-type: none"> 1. The overall process of "Buying and Reserving Flights" is: Search for Flights (enter flight parameters, request flights from SABRE, present top 20 to shopper) <ul style="list-style-type: none"> - Purchase or Reserve a flight (pick one, either Purchase it or Reserve it) - (for reserved flights) either purchase within 24 hours or cancel - After purchased, send shopper e-ticket, and update mileage plan 5. We decided the backbone has no reservations or mileage plan info, and identified and documented the following stories: <ul style="list-style-type: none"> - "Simple Shop for Round-Trip Flights" - "Simple Purchase a round-trip Flight" - "Simple Issue an e-ticket"

Sample Story 4 (Basic Capability)

Title	Simple Purchase a Round Trip Flight
Description	As a <ticket buyer> I want <to purchase a round trip flight n the simplest possible way> so that <I can prove that the website works>
Origin	Analysis of "Buying and Reserving Flights," in meeting, June 14, part of "Buying a Flight Backbone"
Size	Large
Done Criteria	<p>This story will be done when:</p> <ul style="list-style-type: none"> - A user can purchase an already-found round-trip ticket through the web interface - The code has been peer reviewed - The code is protected by unit tests, that all pass in the integrated environment
Notes	<ol style="list-style-type: none"> 1. Purchasing is done AFTER a flight is searched for and returned, we already have a list of round trip flights in front of us to choose from 2. Simplifying assumptions for story: Single leg, e.g., Seattle to Burbank and back; Single Customer, paying full fare, No luggage, pets, etc., Credit Card Payment validation is "stubbed out" and just works, 3. Don't worry about a good-looking interface; just select the flight you want and go...

Sample Story 5 (Testing as Analysis)

Title	Test "Buying a Round Trip Ticket"
Description	As a <coder> I want <Diane to bang on the system> in order to <find new stories that 'seal the deal'>
Story Boss	Diane
Estimate	Timeboxed to 6 hours
"Done" Criteria	All defects found will be documented as stories (title, description, done criteria, and list yourself as SME) and placed on the backlog
Notes	Beat on the following areas: <ul style="list-style-type: none"> - Buying a round trip ticket, including - Updating a passenger's mileage plan
Tasks	Just make sure that: <ul style="list-style-type: none"> - We test in the integrated environment - We document the results as stories - We review the results as we go with the development team and SMEs

Questions So Far?



We Will Discuss StoryoTypes

- Development StoryoTypes – produce product
- Analysis StoryoTypes – produce knowledge and stories
- Infrastructure/Environmental StoryTypes – provide an environment within which the team can do work
- Business Support StoryoTypes – provide support to the business so they can market, sell, or otherwise use the product

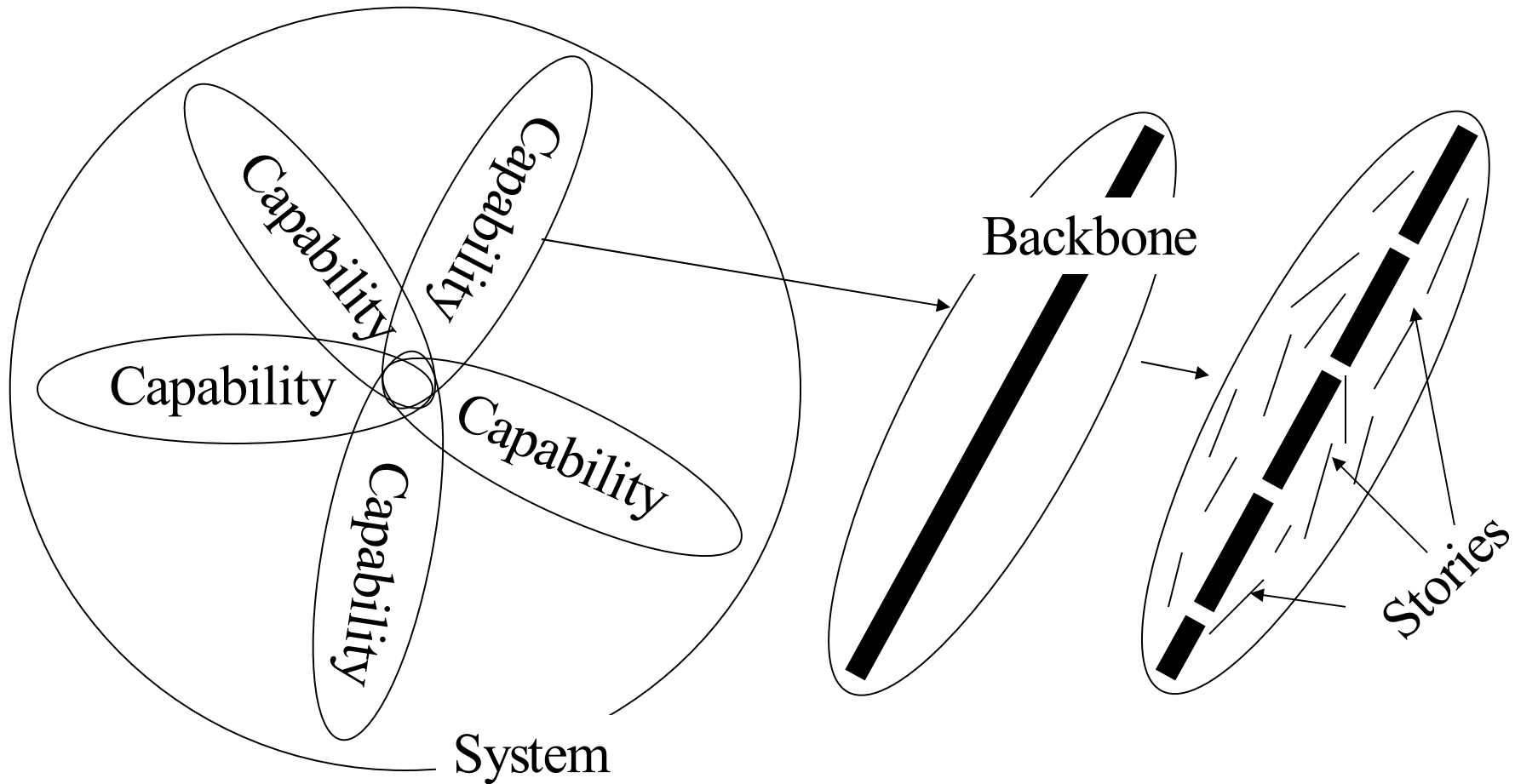
Development StoryoTypes



Development Stories

- The main thing we do in our projects is write code, so we'll discuss the code-writing storytypes first. They are:
 - Development Epic
 - Capability Backbone
 - Basic Functionality
 - Variation of Existing Functionality
 - New Business Rule
 - Improve Interface

Capabilities, Backbones, and Stories



Development Epic StoreoType

StoryoType Name: <i>Development Epic</i>	
<i>Purpose</i>	<p>An Epic is a story that is too big or complicated to do all at once. Typically, epics are large and fuzzy, with unknown scope that needs to be explored.</p> <p>We use epics as organizational constructs. Most of our right-sized development and analysis stories belong to some epic; that is, they are either analyzing or building the capability the epic represents.</p>
<i>Examples</i>	<p>“Buy Tickets on the Web”</p> <p>“As the <auditor> I want <financial reports> so that <I can track what’s going on>”</p> <p>“Manage Customer Information”</p> <p><i>Note: epic descriptions are usually free form and ambiguous. This is how you can tell it’s an epic ☺</i></p>
<i>Details</i>	<p>Usually, an epic starts off like any other story, with simply a title and a description. An epic is often identified when talking to SMEs or in casual discussion amongst the team. Analysis of, and conversations about, epics often generates useful information, including more stories for the backlog, UI diagrams, logical designs, business rules, and so on. It is useful to document all this information as part of the epic, in order to keep it in one place.</p>
<i>Discussion</i>	<p>According to Cohn, there are two types of epics: the compound story and the complex story. In the first case our analytic chore is to “carve up” the epic into stories that are right-sized. In the second case our chore is to figure out what the epic is all about before carving it up.</p>

Capability Backbone StoryoType

StoryoType Name: <i>Backbone Story</i>	
<i>Purpose</i>	<p>The Backbone Story represents the smallest ‘deliverable’ kernel of a System or Capability that subsequent stories build upon incrementally</p> <p>Backbone stories are the unifying theme for the stories they contain. The backbone is a large common thread through the system, that the internal stories implement in pieces.</p>
<i>Examples</i>	<p>“Buying a Flight Backbone” – As a <customer> I want <to have a complete (though simple) ‘Buying a Flight’ experience> so that <I can see that the system works></p> <p>“Simple Cut a Paycheck” – As a <paymaster> I want <to hire a new employee and cut his first paycheck> so that <I can have the first step towards a working payroll system></p>
<i>Details</i>	<p>When analyzing a Backbone story, we use the “notes” section of the backbone story to document:</p> <ul style="list-style-type: none"> •Unifying assumptions, test cases, and designs to be used across all the stories in the backbone story •The titles of the stories that represent the “parts” of the backbone that will be independently developed and integrated together
<i>Discussion</i>	<p>A backbone story is not completed until all of its internal stories are completed, integrated, and tested. This could take multiple iterations, and is risky because of the integration.</p>

Common “done” criteria for coding stories

- UI design reviewed in ‘paper’ form
- Logical Data Model reviewed
- Functional Test strategy (test cases) reviewed
- Periodic reviews with SME/Analyst
- Functional Tests running on Integration box
- Have automated unit tests protecting all class interfaces
- Regression tests (consisting of everybody’s unit tests) pass on Integration box (green bar)
- Information needed for documentation, training materials, and the like are captured
- Frequent pairing or peer reviews so that we have a second pair of eyes on ‘everything’
- Do the XP practices as much as you are able
- Peer reviews to assure code and design standards are met

Basic Functionality StoryoType

StoryoType Name: <i>Basic Functionality</i>	
<i>Purpose</i>	<p>This story “codes up” a simple, demonstrable version of new functionality. This could be a simple happy path of a use case, a create/retrieve round-trip to a database, or any other atomic action from a business perspective.</p> <p>There should be no conditional processing, and the UI should be barely sufficient to show the functionality works.</p>
<i>Examples</i>	<p>“As a <customer> I want <to purchase a previously defined ticket with a credit card></p> <p>“Insert and retrieve an employee’s name and address”</p>

- This is where we finally start delivering some code
- We are probably in the context of some capability, so make sure the stories fit together nicely

Variation of Existing Functionality

StoryoType

StoryoType Name: <i>Variation of Existing Functionality</i>	
<i>Purpose</i>	<p>This story introduces variations of functionality already in the system. In use case terms, this can be thought of as secondary paths, alternative paths, extensions, failures, and so on.</p> <p>This type of story is used for introducing conditional logic into the system as well as extending the original story – perhaps by adding more attributes to a structure.</p> <p>UI development for this storyotype should be minimal, with just enough interface to introduce the information that causes the variation to occur.</p>
<i>Examples</i>	<p>“As a <customer> I want <to make a seat selection when I purchase my ticket></p> <p>“Handle the case when credit card is denied”</p> <p>“Insert and retrieve an employee’s phone and email address”</p> <p>“Insert and retrieve an employee’s second address”</p>

- This is where most of the interesting stuff happens in a system
- Remember that these stories are part of the Capability

New Business Rule StoryoType

StoryoType Name: <i>New Business Rule</i>	
<i>Purpose</i>	<p>These stories extend existing functionality be adding additional constraints that the system must enforce. This kind of story typically involves checking some condition and throwing an error if it is true.</p> <p>Later on, if one wishes the software to take care of the condition without throwing an error, we can do a Variation of Existing Functionality story to handle the error.</p> <p>Any UI development done for this storyotype should be merely to present the error to the user along with any guidance about it.</p>
<i>Examples</i>	<p>“Determine if seat selection is available”</p> <p>“Validate check-sum on Credit Card”</p>

- This work is often combined with the Variation of Existing Functionality story that results from the error
- It is a separate storyotype so that the work can be chunked into smaller bits

Improve Interface StoryoType

StoryoType Name: <i>Improve Interface</i>	
<i>Purpose</i>	<p>One of the oldest maxims in software is “first make it work, then make it pretty.” It is very easy to let UI design take over a development to the detriment of the actual functionality being represented. This is why we have a separate storyotype for UI improvements, and do the simplest possible interfaces in all other stories.</p> <p>Each Improve Interface story should focus on only one thing, or closely related set of things, in order to maintain the focus we need in a good story.</p>
<i>Examples</i>	<p>“Replace ‘Supervisor’ TextBox with Dropdown” “Rework the home page to match the new branding guidelines” <i>Note: sometimes modifying the UI requires changes “behind the glass.” This gets tricky, as you need to decide if there is actually a Variation story here, of if they can be combined.</i></p>

- The real lesson here is “do only one thing at a time” – a separation of concerns is a very important thing when trying to be fast and agile

Other Production StoryoTypes

- *Figure it Out* – a research story, usually involving coding. It can easily turn into an epic or a project...
- *Documentation* – assemble the documentation details into a document for use, either for users, maintainers, whatever
- *Training Materials* – similar to documentation
- *Bug Fix* – a story unrelated to your project, but it needs to be on your backlog as it takes your time
- *Integration across Teams* – in large teams (multiple small teams) we need a story to explicitly integrate pieces produced by the different teams

Questions So Far?



Analysis StoryoTypes



Analysis StoryoTypes

- First of all, Analysis is studying something to figure out what is going on, what needs to be done, it's about the what, not the how...
- Analysis is typically thought of as “overhead” in an agile project
 - But we want to manage ALL our work, remember, so analysis MUST be on the backlog somewhere, either as stories or tasks
 - Some teams try to do their analysis just as tasks, but I find that too hard...
 - So, given that, there must be analysis stories... and hence storyotypes

Common Traits

- There are some common traits about analysis stories:
 - They are usually done by the “business/customer/product owner” side of the team, with development support for validation and verification
 - One of the main things about analysis is that the results must represent something “doable” that fits within our architectural and design schemes
 - They are usually timeboxed, in order to avoid analysis paralysis
 - Typically, we do some analysis, look to see where we are, and then determine if we need to do some more

Reasons for Analysis

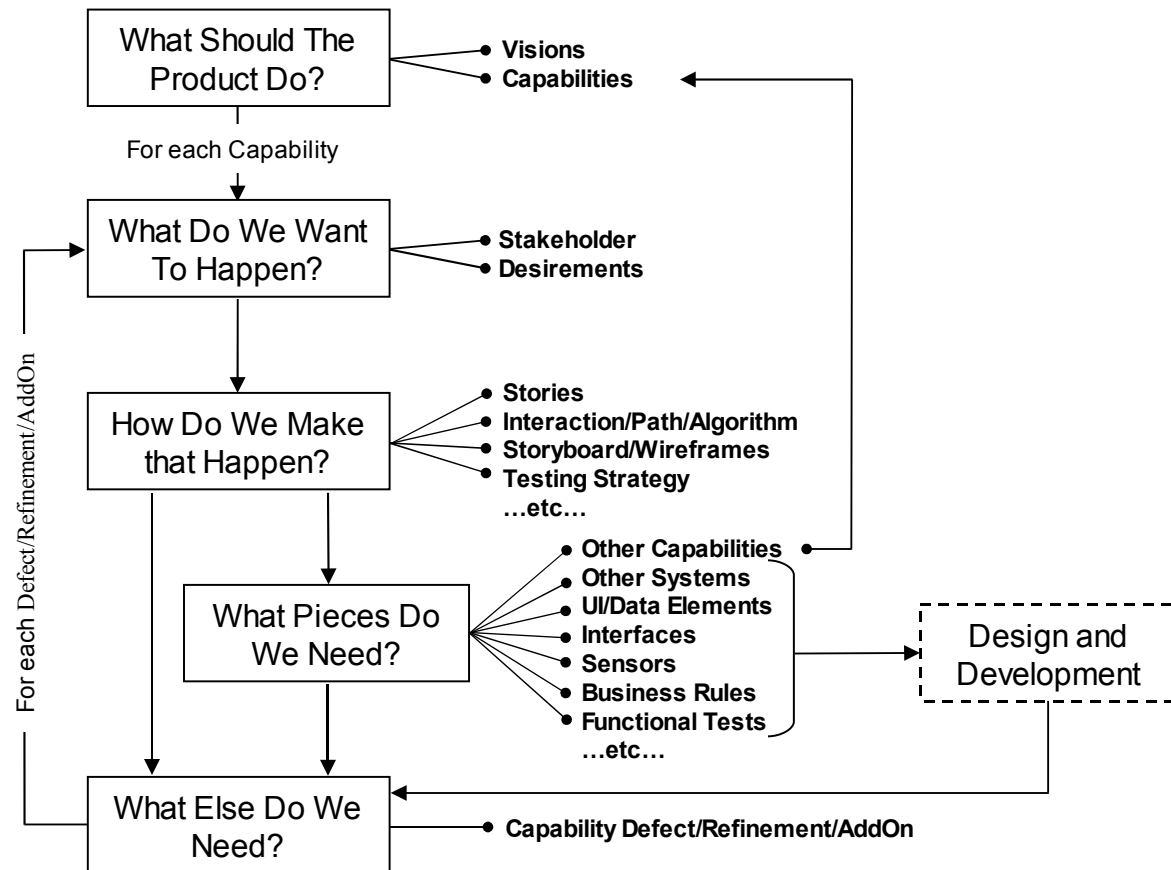
- We have several reasons for analysis in an agile project:
 - Analysis that determines what capabilities are needed
 - Analysis that produces stories that implement capabilities
 - Analysis before a story, getting smart on the story in order to be able to help the developers
 - Analysis within a story, helping the developers work on in the story
- All but the last of these can be called Analysis Stories, these last are tasks within a Dev story

Common Script for Analysis Stories

- Many analysis stories follow the same script, from the analyst's perspective. The analyst drives, and...
 - First, have a goal – determines what StoryoType you are doing
 - Do your homework, figure out the right questions to ask (work with your own team, study other software, requirements documents, etc)
 - Figure out who the right SMEs to talk to are (and who you want to invite from your team), and invite them to a meeting. Provide an agenda based on the story goals
 - Have the meeting, make sure you have a facilitator and a scribe (could both be the analyst, but that is hard, better to have the analyst facilitate, and somebody from you team scribe)
 - Summarize the results, and help your team consume them

Capability-Based Development

- Here's a diagram of Capability-Based Development, showing development steps... mostly analysis, actually...



StoryoTypes with this Basic Script Are:

- Review the System
 - Goals: new/improved business-based capabilities for system, release constraints/strategies
 - SMEs: Business stakeholders, product owner
- Determine Stakeholder Needs/Wants
 - Goal: Specific desires involving a particular capability
 - SMEs: Capability Stakeholders
- Determine Backbone Story
 - Goal: Agreement on a Capability Backbone story
 - SMEs: Capability Stakeholders, Architect

Continued...

- Decompose Backbone Story
 - Goal: Decompose the Backbone story into right-sized stories
 - SMEs: Capability Stakeholders, Dev Team
- Review Capability
 - Goal: Find new stories for a given Capability
 - SMEs: Capability Stakeholders, Dev Team
- Get Smart on Story
 - Goal: to help analyst (and architect) get in tune with stakeholder needs so they can act independently to help team
 - SMEs: Capability Stakeholders, Architect

Combining Analysis Storytypes

- Analysis StoryTypes concentrate on one thing at a time
- They are often combined into actual stories because
 - They are progressive, one often leads to another, with a different focus
 - The right people are available to work on one after another (the same people are the SMEs for different storytypes)
- In the preceding pages, we see how we can combine storytypes if we have the time and the right people in the room – just have a good agenda...

Other Analysis StoryoTypes

- Mine the Change Requests
 - Goal: find new stories
 - Method: review change requests, combine, rework, evaluate, etc, to get new stories
 - Players: Stakeholders, Analyst, Dev Team
- Testing as Analysis
 - Goal: find new stories
 - Method: ad hoc testing to determine defects/gaps in system, document them as stories
- Usability Inspection as Analysis
 - Goal: to find new stories for Capability
 - Observe ‘users’ try out the capability, determine defects/gaps, document them as stories

Questions So Far?



Infrastructure/Environment, Business Support, and Other StoryoTypes



Short and Sweet

- These StoryoTypes are relatively straightforward, so I'll just list them by category
- They exist because:
 - They provide value to the business, project, or product, if not to the user
 - We have to do these things, so must plan for them – so there are stories
 - There are common patterns across projects and organizations – so there are storyotypes

Infrastructure/Environment

- “Set Up” Storyotype – Makes our team able to do work better and easier
 - Set up CM Environment, Development Environment, Test Lab, Etc
- Team Training – courses, internal training, etc
- Add a Tool – “Install FITnesse”
- Improve XXX – “New build process”

Business Support

- Attend a Trade Show
- Write a Paper for a Conference
- Do a Demo
- Train Users
- Prepare Marketing Materials

Other StoryoTypes

- Validation
 - Product Testing (stress, interoperability, security, etc)
 - Usability Testing (in the lab)
- Quality
 - Refactoring
 - Rewrites
 - Extract Framework
 - Add Tests or Documentation to Legacy

Any final Questions?



Thank You Very Much!

