From Concept to Product Backlog

What Happens Before Iteration Zero?

Gerard Meszaros
Concept2Backlog@gerardm.com

Maximizing Value

- One way Agile maximizes value is by accommodating change late in the process.
- The latest version of these slides are available at

  http://Concept2Backlog.gerardm.com

Instructor Biography

Gerard Meszaros is an independent consultant specializing is agile development processes. Gerard built his first unit testing framework in 1996 and has been doing agile test-driven development ever since. He is an expert in test automation patterns, agile project management and improving usability practices on agile projects.

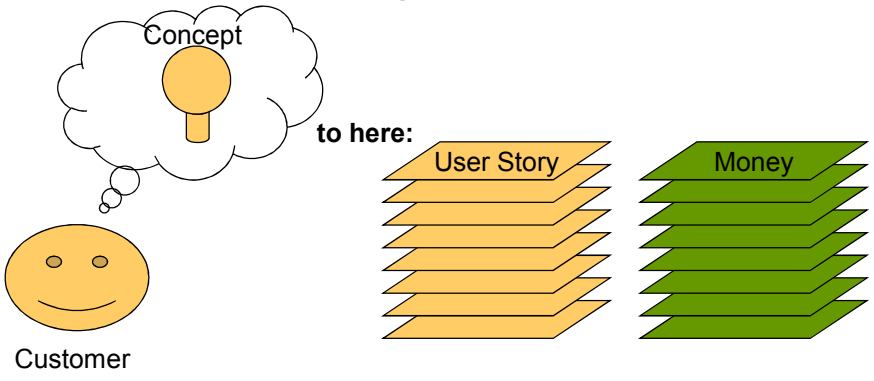
Gerard has applied agile techniques including automated unit and acceptance testing on projects in technologies ranging from Java, Smalltalk and Ruby to PLSQL stored procedures and SAP's ABAP.

He is the author of the Jolt Productivity Award winning book xUnit Test Patterns – Refactoring Test Code and a frequent presenter on agile development at major conferences and user groups.

Gerard Meszaros
Concept2Backlog@gerardm.com

Objectives of Tutorial

- **Understand how to get from here :**



Agenda

- **Motivation**
  - Some Stories
  - Balancing BDUF & YAGNI
- **Product Envisioning**
  - Understand Users & Usage Context
  - Defining the Product
- **Product & Project Planning**
  - Understanding Risk
  - Validating Architecture & Technology
  - Defining Test Automation Strategy
  - Estimating & Release Planning

BDUF: Big Design Up Front
YAGNI: You Ain't Gonna Need It

What's Iteration 0 (Zero)??

- **An iteration dedicated to setting up the development environment before the first "real iteration" is started.**
- **Usually includes:**
  - Installing development tools on workstations
  - Installing team tools on servers
  - Priming the User Story Backlog ready for the Iteration 1 Planning Meeting (IPM1):
- **Optional Secondary Objectives:**
  - Calibrating the development team's velocity
  - Learning any new technologies that will be used

True Story (1)

- Fired up project
- Got developers on board & started Iteration 0
- Worked with customer to define user stories
- Got iteration 1 going but Customer didn't get all the story tests finished
- Quick, need to plan for Iteration 2
- Got iteration 2 going but Customer didn't get all the story tests finished
- Customer never really got caught up on story tests ….
- … even by Iteration 10

Root Cause

- Customer was learning "on the job"
- Customer never had a chance to get ahead of developers …
  - … because they all started at same time

True Story (2)

- Developers were being fed stories each iteration
- Core functionality was XML-based messaging
- Fit tests were built for the messaging functionality
- Subsequent stories kept breaking existing Fit tests due to new logic being introduced

Root Cause

- Messages contain multiple input parameters
- As new ones were introduced, business calculations resulted in different expected outputs
- Eventually addressed through intelligent defaulting of input fields in Fit fixtures
- Could have been avoided by coming up with strategy sooner …
- … but problem couldn't be anticipated because of "Story Blinders"
- A.K.A. "Don't look ahead because YAGNI!"

YAGNI: You Ain't Gonna Need It

True Story (3)

- Developers were being fed stories each iteration
- Support functionality involved web-based data administration & searching of messages
- Acceptance testing of administration UI revealed many show-stopper usability issues
- Work on next release had to be delayed by over a month while issues were resolved.

Root Cause Analysis

- User Interface contained many inconsistencies
- Caused by incremental development of UI without reference to a common UI design
- Caused by “Story Blinders”

What Do These Three Stories Have in Common?

- **Lack of up-front planning led to sub-optimal results…**
- **… that lead to significant rework …**
- **… that could have been avoided with a small amount of up-front planning**

We were missing the "Big Picture"

The Role of Up Front Planning in Agile

Often:

- not addressed explicitly, or
- explicitly discouraged

BDUF is a 4 letter word for many agile teams

- Big Design, Up Front "is a waterfall practice"

The Simplest Thing That Could Possibly Work

- "we can get there faster through refactoring"
- Don't design ahead; YAGNI!

BDUF: Big Design Up Front
YAGNI: You Ain't Gonna Need It

Strengths of Agile

- **Deferring Commitment to Requirements**
  - Giving business the chance to learn before locking in requirements and dates
  - Allowing requirements to emerge late in the project without significant penalty
  - Allows business to manage scope to meet time commitments
- **Deferring Commitment to Cost/Timeframes**
  - Giving development the chance to learn domain before locking in dates
  - Giving development the chance to learn new technology before locking in dates

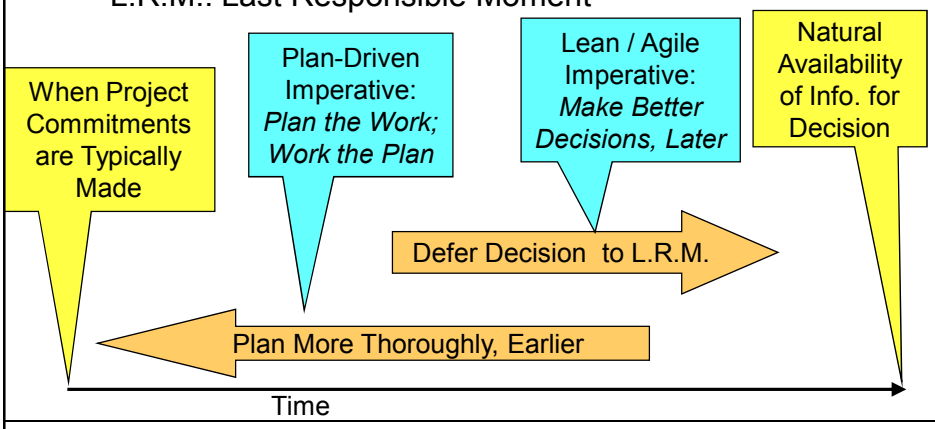
Weaknesses of Agile

- **Dogmatic avoidance of Up Front Planning**
  - Leads to lack of big picture
- **Lack of data for funding – Catch 22**
- **Lack of advance warning for subcontractors**
  - Can result in delays once executing project
- **Perception of lack of a plan to follow**
  - May or may not be accurate

Can All Be Addressed With Some Planning

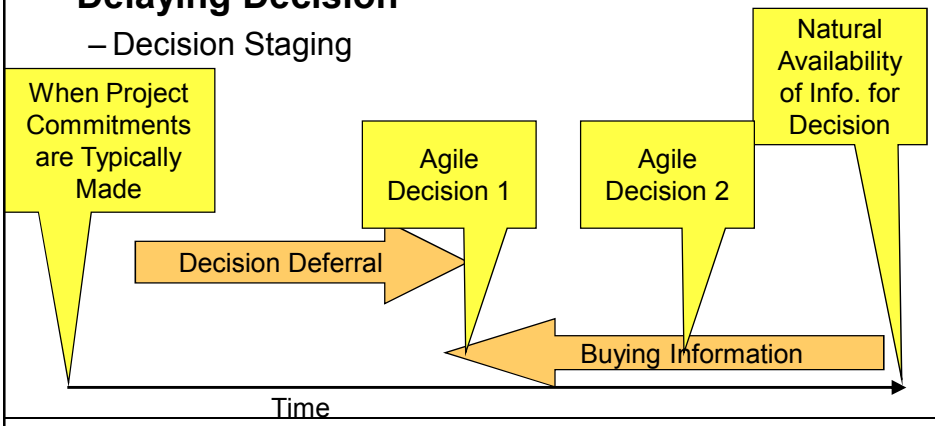
Agile Planning Tension

- **BDUF vs LRM**
  - Big Design Up Front
  - L.R.M.: Last Responsible Moment

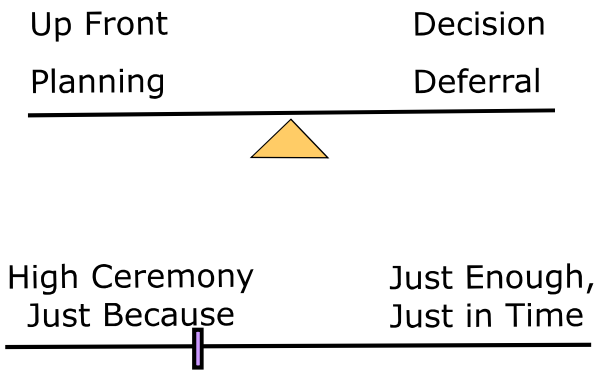


Ways to Get Better Data

- **Prototyping**
  - Buying information through activity
- **Delaying Decision**
  - Decision Staging



Finding the Right Balance

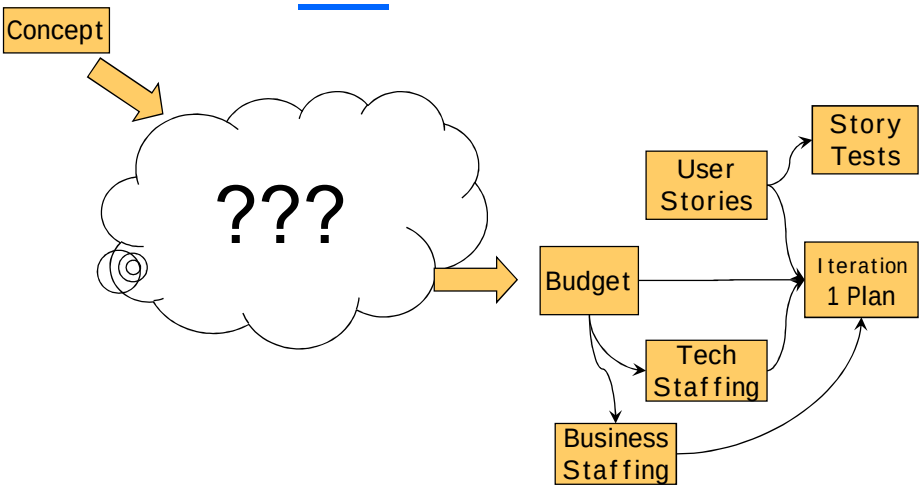


Five Levels of Agile Planning

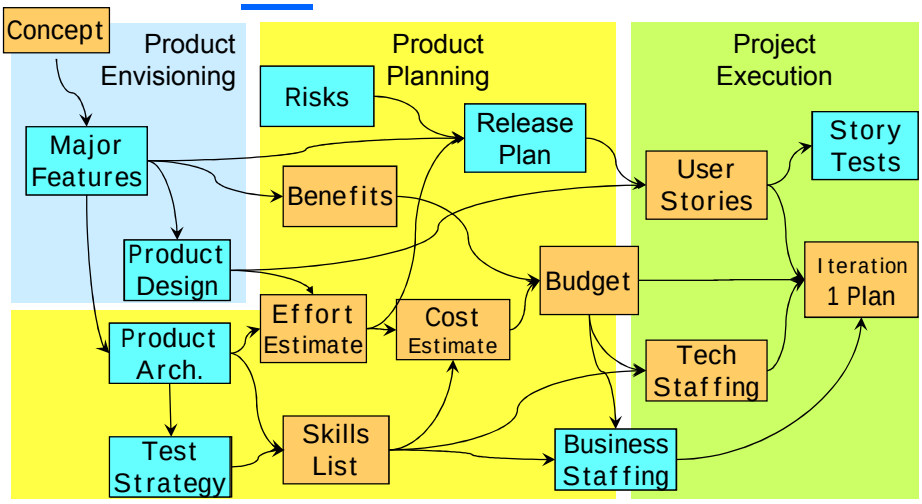
- **Product Vision**
- **Product Roadmap** Might be skipped for smaller products
- **Release Plan**
- **Iteration Plan**
- **Daily Plan**

We need all these plans;
Level of detail increase as horizon shortens

What Do You Need to Get Started



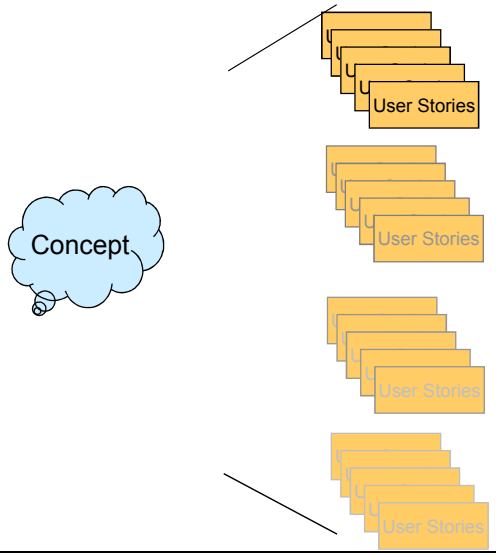
What We Needed to Get Started



An incomplete set; *you* may need other things too!

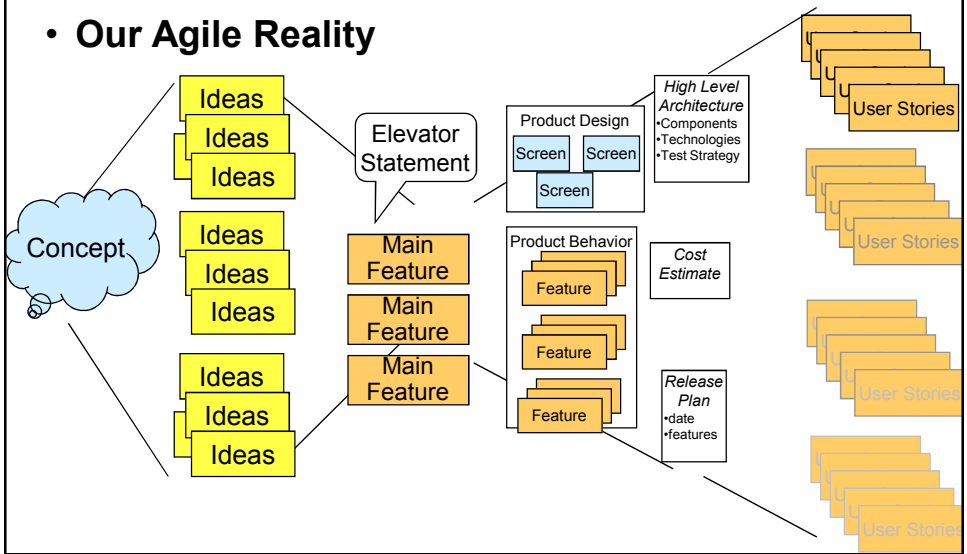
Product Envisioning & Design

- **Agile Myth**



Product Envisioning & Design

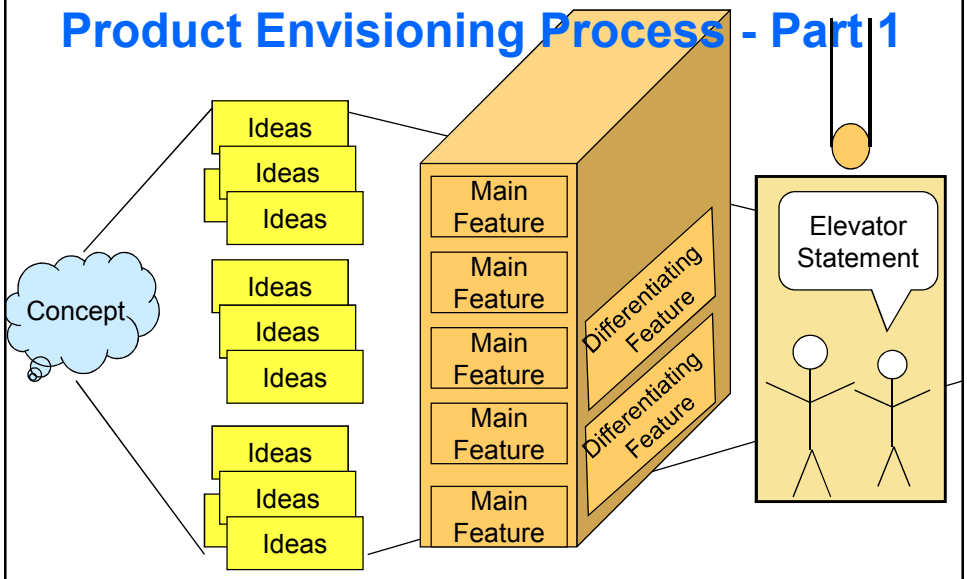
- **Our Agile Reality**



Product Envisioning

- **Goal: Collective understanding of Product**
- **Why: Get everyone working to a common purpose**
  - Get everyone's input into product
  - Get everyone's buy-in
  - Help everyone understand the big picture
- **How: Multi-disciplinary workshops for envisioning product to be built**
  - Brainstorming, listing functionality, users, value
  - Summarize key capabilities & value
- **Several potential outputs:**
  - Elevator Statement
  - "Product Box"
  - Major & Differentiating features list
  - Low-fidelity User Interface Prototype

Product Envisioning Process - Part 1



Product Box

- Product Graphic
- 15-20 main Features
- 3-4 key differentiating Features

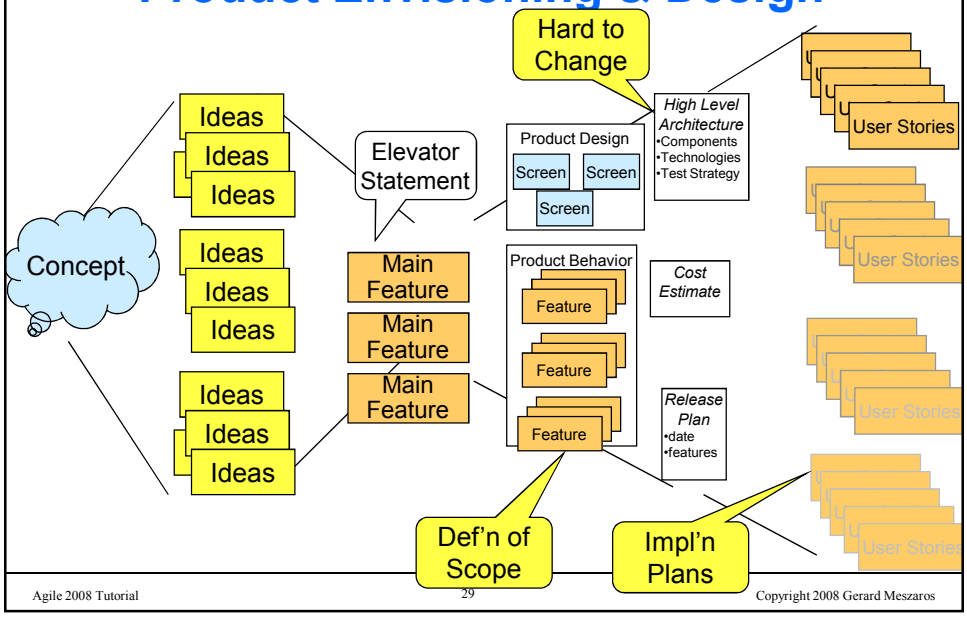
- Helps team focus on what's really important

Elevator Statement

- For (target customers)
- who are dissatisfied with (the current market alternatives),
- our product is a (new product category)
- that provides (key problem-solving capability).
- Unlike (the product alternative)
- we have assembled (key "whole product" features for our specific application)

Crossing the Chasm - Marketing and Selling High-Tech Products to Mainstream Customers.
By Geoffrey A. Moore - See P154

Product Envisioning & Design



Product Design

Not Software Design!

Purpose:

- **Provide the "big picture" of the product**
  - Major UI navigation paths – for UI intensive apps
  - Major data model – for data intensive apps
  - Major messaging protocols –for messaging apps

Techniques:

- Use Case Modeling
- Information & Data Modeling
- Interaction & Graphic Design
- Protocol Design

Traditional Approach

User/Usage Centred Design

- **Ethnographic studies**
- **User Roles or Personas**
- **Detailed Task Analysis**
- **Usability labs**

Not done that often
Very heavy weight when done

Use Case Modeling

- **Use Case Model (complete)**
- **Use Case documents (detailed)**

High Ceremony
Just Because

Just Enough,
Just in Time

Agile Product Design

- **Lightweight User Task Modeling**
  - Actors & Goals list or Essential Use Case Model
- **Paper Prototyping**
  - To define what the product looks like
- **Wizard of Oz Testing**
  - To get feedback on that design

This moves product from
"Just OK"
to
"Love this Product!"

High Ceremony
Just Because

Just Enough,
Just in Time

GGM40

Essential Use Case Model

Alternative: Actor & Goals List

High Ceremony
Just Because

Just Enough,
Just in Time

Actors & Goals List

Actor	*Goal*
Account Manager	• Prepare quotes for customers • Request prices from business partners • Challenge existing rates • Request new/missing rates
Price Analyst	• Prepare quotes for customers • Request prices from business partners • Maintain rates • Respond to Rate Challenges
Contract Administrator	• Publish new rates

Alternative: Use Case Model

High Ceremony
Just Because

Just Enough,
Just in Time

Slide 33

GGM40 Replace with PR3 model
Gerard Meszaros, 29/06/2008

Personas

- **A handy way to keep users in front of mind;**
- **Caricatures of kinds of users, not roles they play**

Crusty Cal:

- 25 years with company
- Barely sufficient PC skills
  - E-mail, work apps
- Not keen on learning
- Will only do what is taught

Keener Kelly:

- 2 years out of college
- Life centers around PC
  - Facebook, 2nd Life
  - Loves keyboard shortcuts
- Keen to learn new things
- Willing to experiment

Posted on wall & Asked Frequently:
How would Crusty Cal react to this?

Paper Prototyping

- **Low fidelity screen mock-ups**
  - Rougher is better; says "we're open to input"
- **A tangible representation of what app is about**
  - Helps communicate it to all project stakeholders
- **Get feedback from users to arrive at a better design**
  - Do this as cheaply as possible …
  - before we've invested too much in the design

Refs: About Face by Alan Cooper
Paper Prototyping by Carolyn Snyder
Agile Usability by Jeff Patton

High Ceremony
Just Because

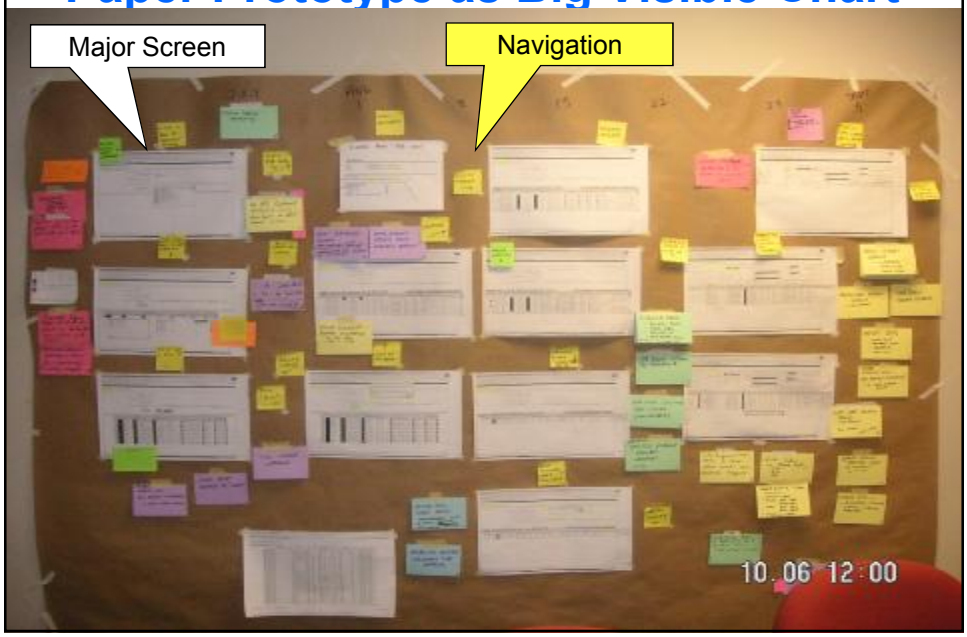
Just Enough,
Just in Time

From Concept to Product Backlog

Paper Prototypers in Action!

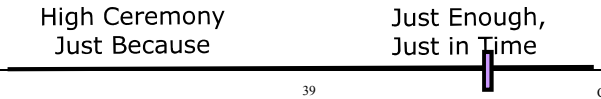
From Concept to Product Backlog

Paper Prototype as Big Visible Chart



Wizard of Oz Testing

- **Purpose:**
  - Validate the Product Design
  - Find design defects quickly
  - Make design decisions based on data, not speculation or opinion
- **How:**
  - Test early versions of the design with users thru simulated execution of paper prototypes
  - Gather data on alternative design options



Wizard of Oz Testing *

- **Roles:**
  - 1 Test Facilitator (business SME)
  - 2 people playing computer, coprocessor & HELP system (anyone)
  - 2-3 observers (developers + business)
  - 1 or 2 test subjects
- **Preparation:**
  - Task descriptions – what the user will attempt to do
  - Paper Prototypes – what the user will use to do it
- **Test Sessions:**
  - Typically about 1 hour each
  - Tested with pairs of end users (co-discovery)

* = Slide may be skipped during presentation

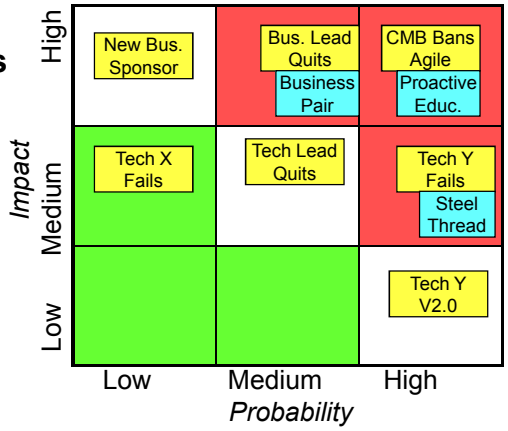
Wizard of Oz Testing

Test Session Workflow *

1. **Facilitator describes testing process**
   1. How the user and computer will interact
      - **By pointing or writing**
   2. The role of the observers
      - **"Please speak your thoughts so they can record them"**
   3. How the user can ask for help
      - **"Point to the HELP button in the top right corner"**
2. **Facilitator provides user(s) with task to do**
3. **User attempts to do task using "application"**
   - By "clicking" and "typing" in "application"
4. **Observers record any problems encountered**
5. **Facilitator debriefs the user**
   - Was there anything that particularly confused you?

Light Weight Risk Assessment

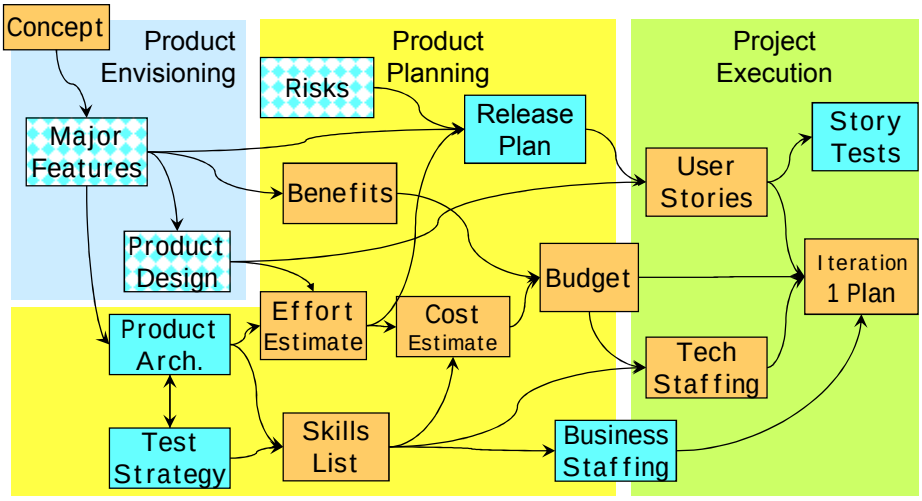
- **Cardstorm potential events & write on cards**
  - Discuss likelihood & impact
- **Brainstorm mitigation strategy for Red Risks**
  - Monitor White Risks for change in probability or impact
- **Repeat every few months or when something significant changes**



High Ceremony Just Because

Just Enough, Just in Time

What We Need to Get Started



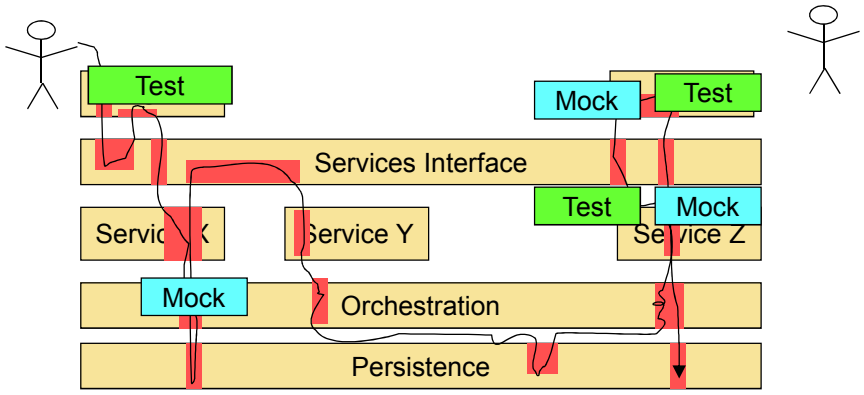
An incomplete set; *you* may need other things too!

Define Product Architecture

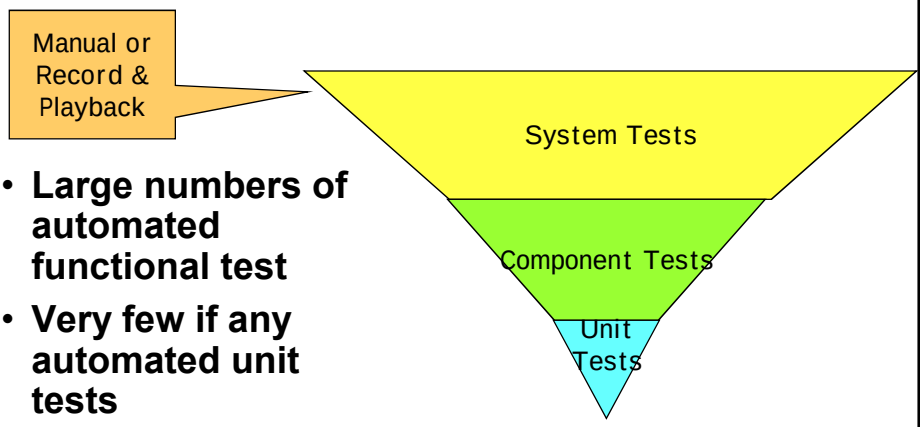
- **Goal: Understand the key components and technologies that will comprise the finished product.**
- **Why: Reduce Technology & Schedule Risk**
  - Avoid high-cost technical changes
- **Activities:**
  - Propose architecture
  - Evaluate technologies
  - Build "Steel Thread" (executable skeleton)

Validate Architecture via Steel Thread

- **Implement just enough of the architecture to prove it hangs together; most logic can be hard-coded**



Test Automation Pyramid

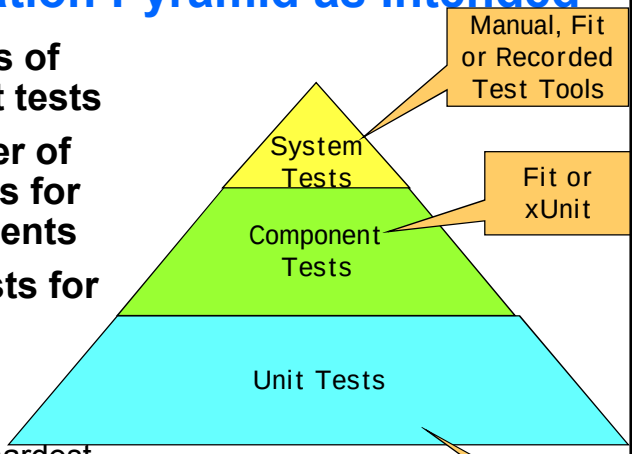


- **Large numbers of automated functional test**
- **Very few if any automated unit tests**

If you don't think about Test Strategy, Test Automation Will be Difficult and Expensive

Test Automation Pyramid as Intended

- **Large numbers of very small unit tests**
- **Smaller number of functional tests for major components**
- **Even fewer tests for the entire application & workflow**
  - These are the hardest, most expensive to automate



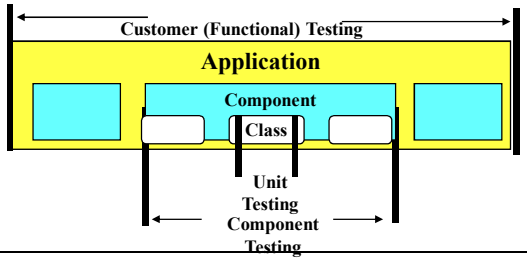
Tests Should be Complementary

Define Test Strategy

- **What kind of testing is required?**
  - Unit
  - Component?
  - Functional?
  - Workflow?
- **Which tests need to be automated?**
  - Which tests need to be run often?
  - Which tests will take most effort to run?
- **What test automation challenges are there?**
  - Legacy systems?
  - Binary data?
  - Time/date-based logic?

Why We Need Multiple Kinds of Tests *

- **Functional Tests will tell us which Features of the product doesn't work**
- **Component tests will tell us which component is at fault. Also test complex business logic directly.**
- **Unit tests tell us exactly which class/method is broken**



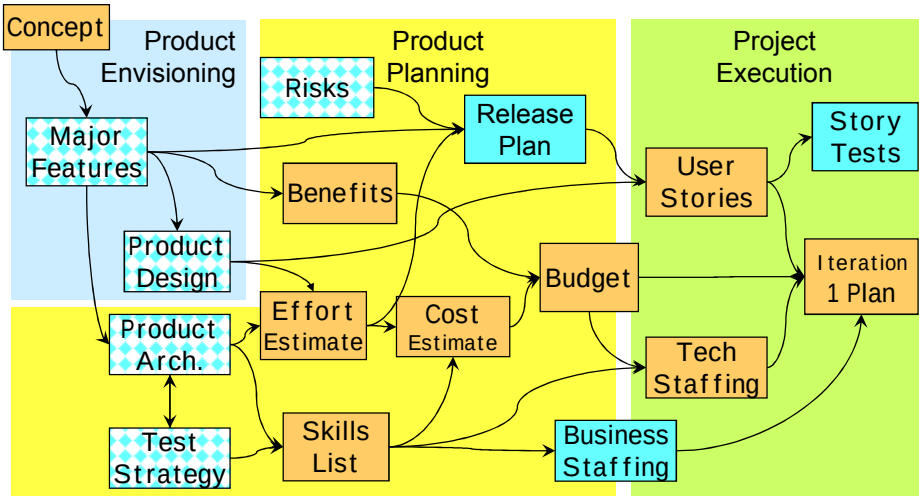
Unit Tests also let us test code we cannot hit in functional tests

Define Testability Requirements *

How does system support test automation?

- **Stub-able interfaces to systems**
  - Control inputs from, monitor outputs to other systems
- **Stub-able data abstraction layer**
  - Control data inputs easily during tests
- **Date/Time control interface**
  - Simulate passage of time
- **Complex business rules & calculations in testable components**
  - Business "Unit Tests"

What We Need to Get Started



An incomplete set; *you* may need other things too!

Release Planning

- **Conceptual Integrity of Release**
  - Set of related functionality for a specific set of users
  - Add more kinds of functionality, for other kinds of users in subsequent releases
- **Full spectrum of priorities**
  - Must haves, Important, Would be nice
  - Allows "wiggle room" for managing scope later
- **Estimate Features, Not User Stories**

- **What not to do:**
  - Put all Priority 1 Features into Release 1
  - Removes ability of agile to manage scope to fit budget

Estimating for Release Planning

- **Typically won't have all the User Stories defined (you don't need them yet – YAGNI!)**
- **Estimate & sum the feature complexity and guess the feature point velocity to find fit**
  - Yes, you'll guess wrong!
- **Or do more detailed estimating on subset of features**
  - See "Agile Estimating and Planning" by Mike Cohn

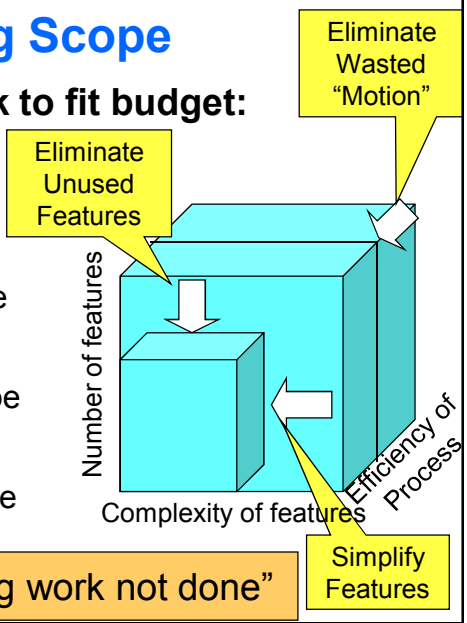
But Always, Always:

- **estimate Complexity, then**
- **derive Effort**

Managing Scope

Ways to reduce total work to fit budget:

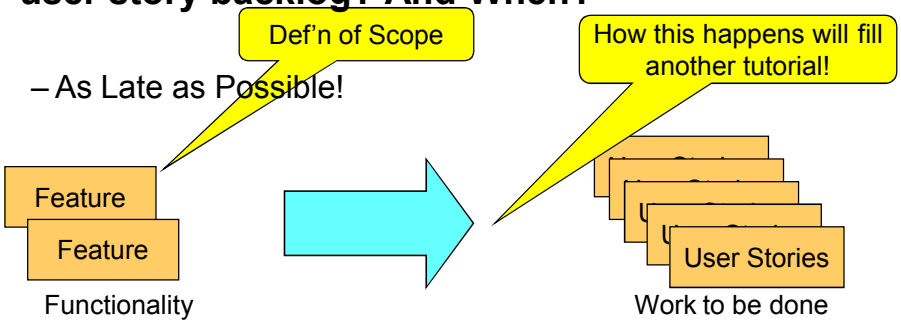
- **More efficient process**
  - Increase team velocity
  - by spending less time
  - doing delivering same value
- **Fewer features**
  - Remove features from scope
- **Simplify features**
  - Lower cost with similar value



"Agile is the art of maximizing work not done"

Defining Product Backlog

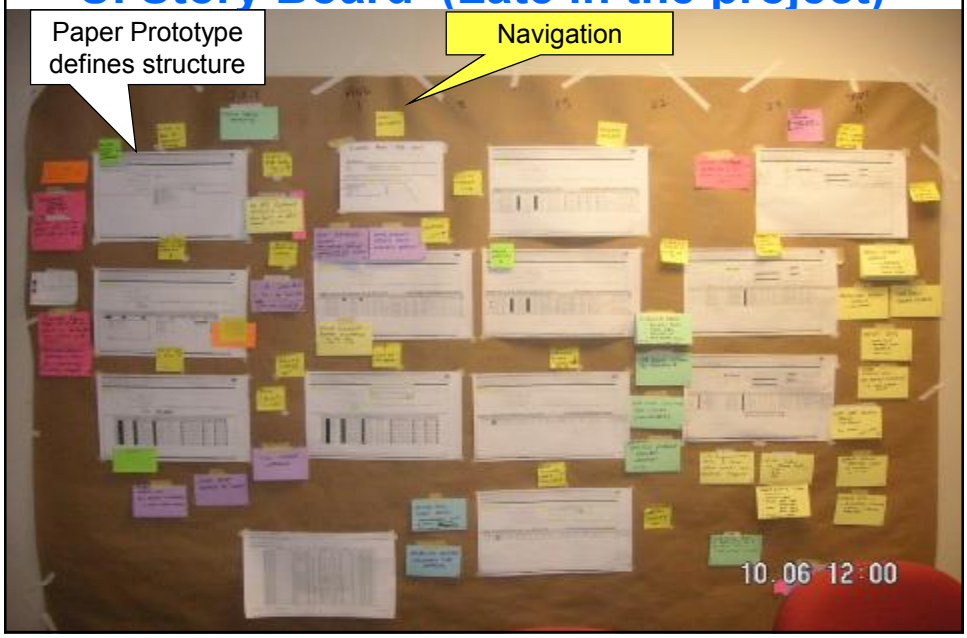
- **How do we go from major features to detailed user story backlog? And When?**
  - As Late as Possible!



- **How do we know we have "all the stories"?**
  - Can annotate the "big picture" with the detailed backlog

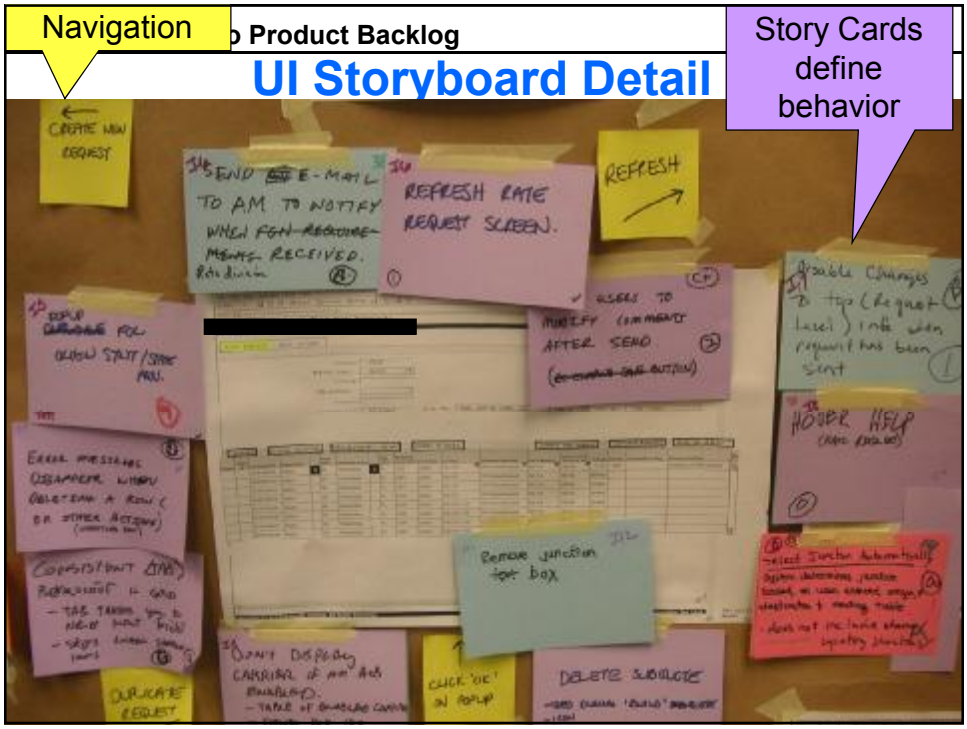
From Concept to Product Backlog

UI Story Board (Late in the project)



From Concept to Product Backlog

UI Storyboard Detail



Clarifying Requirements via Story Tests

- **Show what "done looks like"**
  - Stephen Covey's 2nd "Habit": "**Begin with the End In Mind**"

"*The Seven Habits of Highly Effective People*" by Stephen R. Covey published by Free Press

- **Enumerate all the capabilities that must be supported**
  - Identify the success paths
  - Identify all the failure & error scenarios
- **Provide detailed examples of input data and responses**

Tests/Examples defined by business resources

Fit Example: Calculation Test

Using a Row Fixture:

PayrolFixtures.WeeklyCompensation			
Standard Hours	**Holiday Hours**	**Hourly Wage**	**Pay( )**
40	0	10	$400
40	0	20	$800
41	0	20	$830
40	1	20	$840 *expected* $800 *actual*
41	1	20	$870 *expected* $830 *actual*

Note: Fixtures can have side effects. Pay() could add these rows into a database table

Fit Example: Multi-Step Test

- **Load data into table using Column Fixture:**

PayrolFixtures.EmployeeTableLoad			
Number	Name	Hours	Add()
10001	Gerard Meszaros	45	OK
10002	John Smith	40	OK
10003	Jane Doe	420	OK *expected* Too Big *actual*

- **Exercise system using Action Fixture**

PayrolFixtures.RunPayrollActionFixture			
RunJob	Payroll		

- **Verify database contents using Row Fixture**

PayrolFixtures.EmployeeCompensationThisWeek			
Number	Name	Hours	Pay
10001	Gerard Meszaros	45	$950
10002	John Smith	42 *expected* 40 *actual*	$860 *expected* $800 *actual*
10003 *surplus*	Jane Doe	20	$800

Business / Customer Staffing

- **Important to get the right business resources for the Customer Team:**
  - Strong business leadership & vision
  - Understands needs of users or can find out quickly
  - Willingness to learn new ways of working
  - Can plan ahead; not just react
  - collaborative, decisive, conceptual thinker

Wrong resources can:

- Slow down the project
- Lead the team in the wrong direction

Business should provide a Star, not a Lemon!

Business / Customer Staffing

- **Core business team needs to be staffed in time to:**
  - participate in the envisioning
  - be trained on how to do the "customer" job
  - start defining features, user stories and story tests
- **Can augment with additional "do-ers" later**
  - but need to take the time to bring them up to speed on the vision

Technical Staffing

- **Need Senior Developer Early in Process**
  - Estimate Features
  - Define Architecture
  - Validate Architecture
- **Full Staffing Should Wait for Budget Approval**
  - but need to take the time to bring them up to speed on the vision

Onboarding Business & Technical Staff

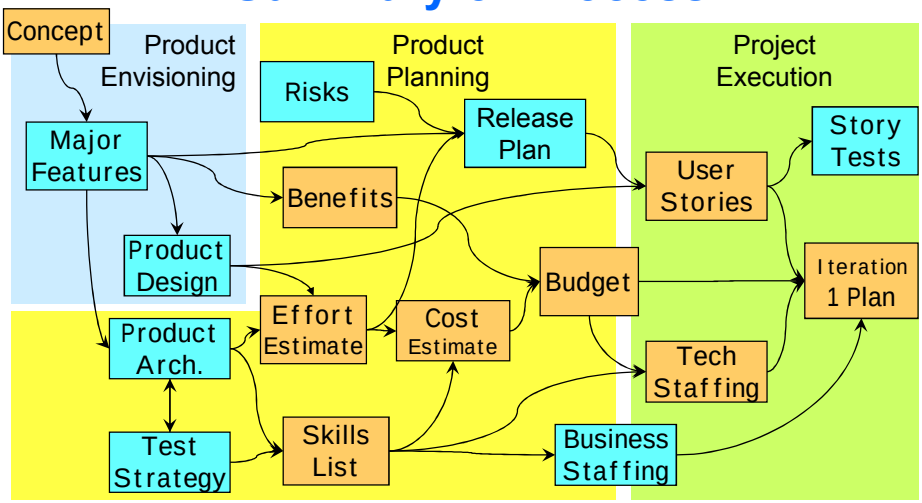
- **Project Background & Vision**
  - Product Concept (the Product Box)
  - Value Proposition (the Elevator Statement)
  - Release Plans
  - Features or Users Stories
- **Process Background & Norms**
  - Agile team practices like iterations and STDD
  - Usability practices
  - Technical practices like TDD, CI & Design for Testability

And Finally, On to Iteration 0

- **Should have entire team on hand**
  - Business resources
  - Technical resources
  - With everyone suitably “on-boarded”
- **Should have “Bootstrap Story” ready to go**
  - Ready to estimate
  - Story tests
  - Additional stories in case team gets done early

Or Maybe Straight to Iteration 1?

Summary of Process



An incomplete set; *you* may need other things too!

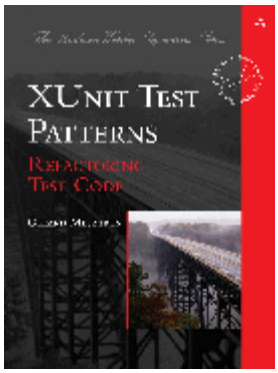
Thank You!

Gerard Meszaros
Agile2008@gerardm.com

latest slides: Concept2Backlog.gerardm.com
http://www.xunitpatterns.com
http://blog.xunitpatterns.com
http://www.gerardmeszaros.com

Call me when you:

- **Want to transition to agile or lean**
- **Want to do agile/lean better**
- **Want to improve usability of your applications**
- **Want to teach developers how to unit test**
- **Want help with test automation strategy**



Jolt Productivity Award winner - Technical Books

Useful References – Agile Books

- **Agile Project Management**
  - Jim Highsmith
- **Fit for Developing Software**
  - Rick Mugridge & Ward Cunningham
- **User Stories Applied**
  - Mike Cohn
- **Effective Use Cases**
  - Alistair Cockburn

Useful References – Usability Books

- **Design of Everyday Things**
  - Dan Norman
- **Design for Use**
  - Larry Constantine & Lucy Lockwood
- **About Face**
  - Alan Cooper
- **Agile Usability**
  - Jeff Patton
- **Paper Prototyping**
  - Carolyn Snyder

Useful References – My Papers

- **Adding Usability Testing to an Agile Project**
  - Experience Report @ Agile 2006
- **Agile ERP**
  - Experience Report @ Agile 2007
- **Using Storyotypes to Split Bloated XP Stories**
  - Experience Report @ Agile Universe 2004

Common Mistakes

- **Not enough up-front planning**
  - "Flying by seat-of-the-pants", "flying blind"
- **Too much up-front planning**
  - "Analysis Paralysis"
  - Detail design
- **Choosing release contents based on Priority**
  - No room for managing scope since everything in release is Priority 1

