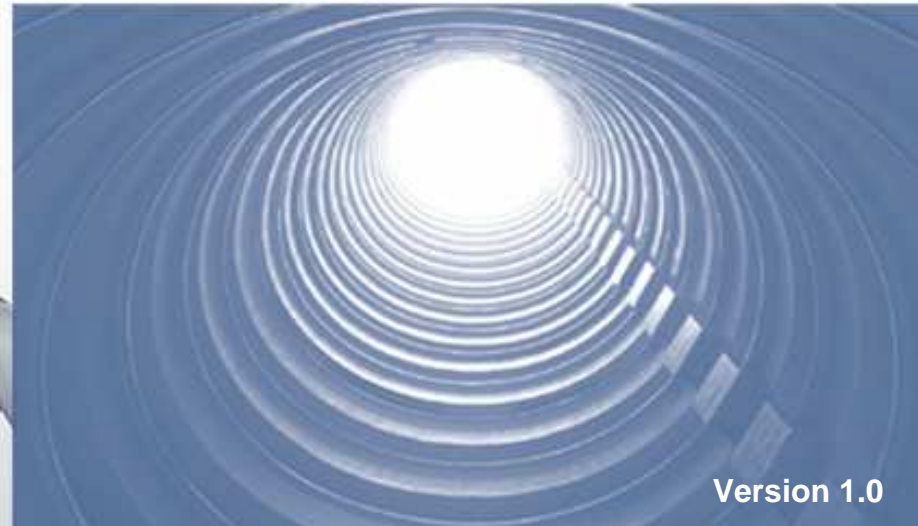




Developing Agile Requirements

User Stories for a Scaled Project

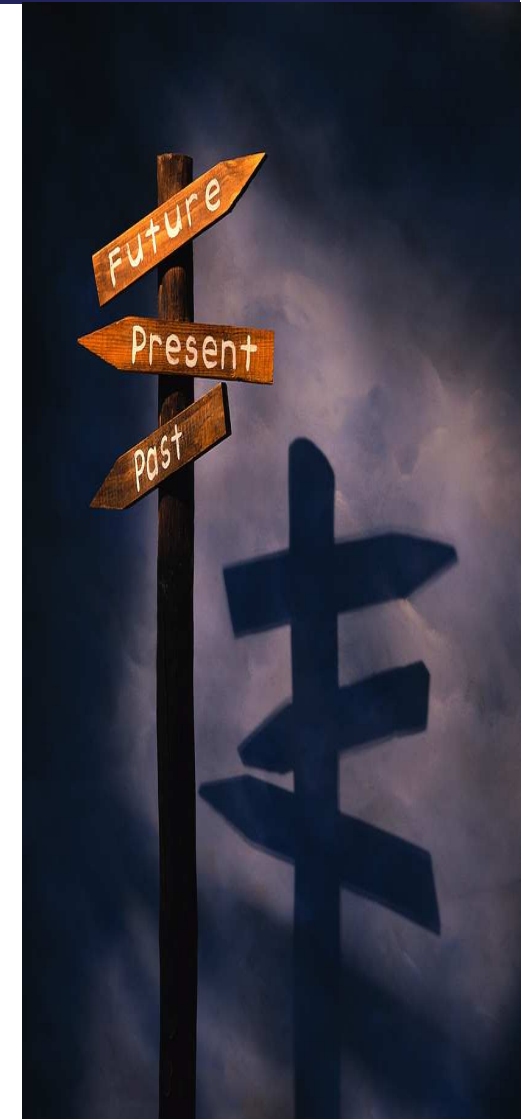
Art Staden (636) 530-7776 www.iconATG.com



Version 1.0

Outline

1. The Agile Philosophy
2. Understanding User Stories
3. Patterns & Anti-Patterns
4. Scaling Up



Agile Principles

❖ Agile methods start with this value statement:

Individuals and interactions	over	processes and tools
Working software	over	comprehensive documentation
Customer collaboration	over	contract negotiation
Responding to change	over	following a plan

That is, while there is value in the items on the **right**,
we value the items on the **left** more.

www.agilemanifesto.org

❖ Another Agile principle tells us to delay specifying and designing a thing until ready to implement it

- And to do the simplest implementation now, to meet today's requirements

How Agile Methods Affect Requirements

- ❖ **Less emphasis on documentation, more emphasis on conversation**
- ❖ **Requirements artifacts may not be retained or maintained**
- ❖ **Test cases are used to define the details of requirements or user stories**
- ❖ **All the project requirements are not gathered and documented up front**
- ❖ **Agile methods assume that it is impossible to predict the final desired content at the beginning of the project**

User Stories

- ❖ Agile “requirements” are usually in the form of user stories
- ❖ A user story typically has the form:
 - *As a <user role>, I can <goal>, so that <motivation>*
- ❖ Where
 - *User Role* is the type of user performing the function
 - » Same thing as an Actor in use cases
 - *Goal* is what the user is trying to accomplish in using the system
 - *Motivation* is why the user wants to accomplish it (*optional*)

As a Customer, I can select items to purchase, to obtain goods I'd like to have.

As a Customer, I can pay for the items in my shopping cart with a credit card, to complete the purchase.

Card, Conversation, and Confirmation

- ❖ Stories on cards are just the visible manifestation of user stories
- ❖ User stories are said to be “*a reminder to have a conversation*”
- ❖ User stories have three aspects:
 - **Card**
 - » A written description of the story used for planning and as a reminder for a conversation
 - » Written prior to the iteration in which it is implemented
 - **Conversation**
 - » Conversations between the Product Owner and developers to flesh out the details of the story
 - » Small conversation during iteration planning, more detailed conversation during story design and implementation within the iteration
 - **Confirmation**
 - » Acceptance criteria that documents story details and used to determine when story is complete
 - » Noted on back of story card whenever conversations are held

Acceptance Criteria

- ❖ Requirements details revealed in conversations are captured as acceptance test criteria
 - Traditionally on the back of the user story card
- ❖ Acceptance criteria are then turned into actual tests which are executed to demonstrate a story's implementation is complete
- ❖ Customer writes user stories and document customer's vision and expectations

As a Customer, I can pay for the items in my shopping cart with a credit card, to complete the purchase.

- Test Visa, M/C & Amer. Exp. pass
- Test Diner's Club fails
- Test good, bad, missing card IDs
- Test future and past expiration dates
- Test purchase amount within and above credit limit

INVEST in a Good Story

- ❖ **I**ndependent
 - Write stories that don't depend on other stories
- ❖ **N**egotiable
 - A story is a reminder to a later conversation
 - Details, when revealed, are captured as acceptance criteria
- ❖ **V**aluable
 - Provides business functionality to users, or financial or quality benefit to purchasers ... achieves a business goal
 - Avoid stories that only bring value to developers
- ❖ **E**stimable
 - Needs to be small enough for estimate to be accurate
 - Needs to be specific enough to understand scope
- ❖ **S**mall
 - Should be able to develop in 1/2 of iteration duration or less
- ❖ **T**estable
 - Have to know when it is done

Patterns & Anti-Patterns

- ❖ **User stories are about what a user wants and does**
 - After all, their not called “developer stories”
 - So make them about users
 - Include the user role in the story
- ❖ **User stories should seek to achieve a business goal for a user**
 - A story should be big enough to be meaningful to a user
 - A story should produce a result noticeable by a user
- ❖ **Keep UI out of stories**
 - UI design is just that: design
 - UI design might be mentioned in acceptance criteria, but try not to
- ❖ **Each story should only discuss one user**
- ❖ **Customers, Product Owners, Users, etc. write stories**
 - While developers can contribute, they shouldn't be the author
- ❖ **Leave details for conversations, stories should be brief**

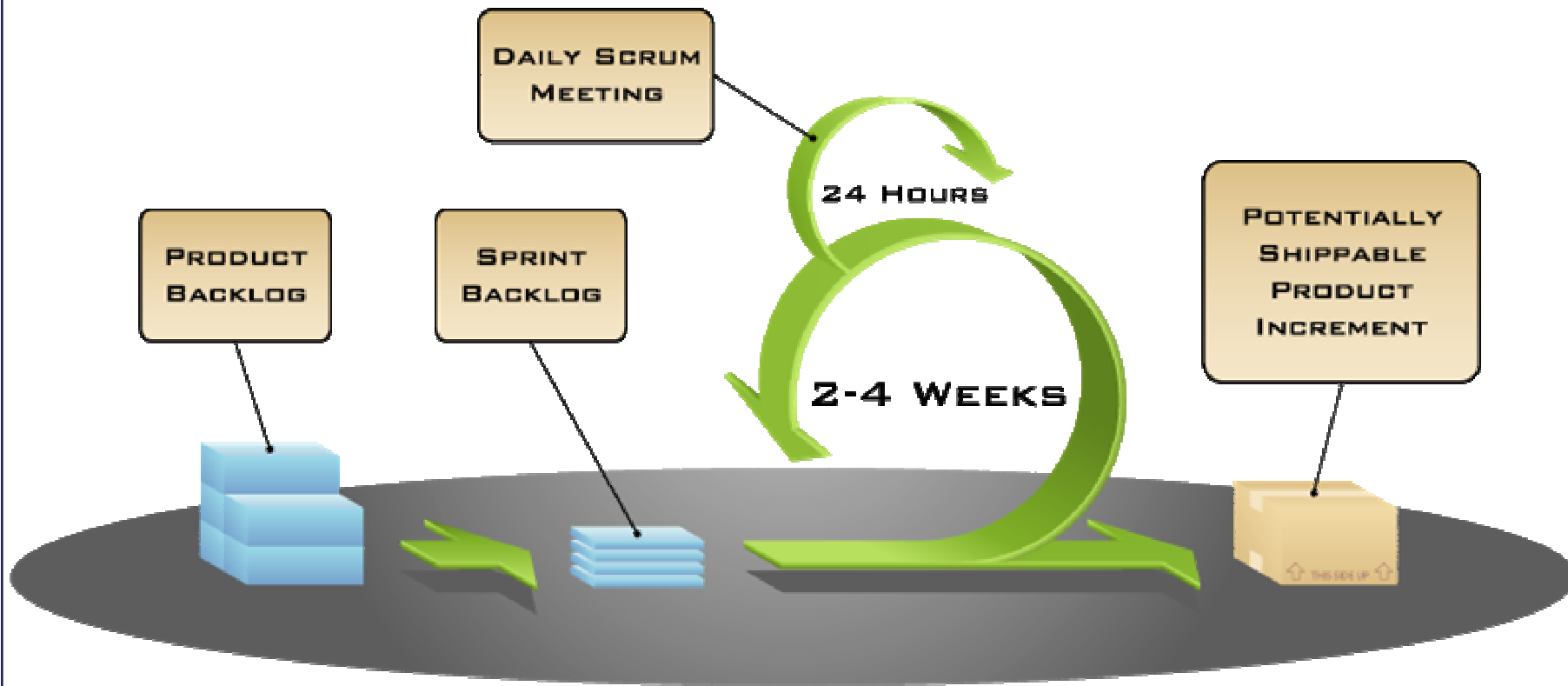
Patterns & Anti-Patterns

- ❖ **Epic stories need to be sliced into implementable stories**
 - Stories need to be implemented in 1/2 an iteration or less
- ❖ **Slice stories vertically**
 - Slice stories to include multiple tiers of the application: UI through to the database and back, for example
 - » Don't write UI only or database only “user” stories
 - » To cut back on “size” of story, limit first story to one UI and Db field, then add remaining fields to a second story
 - Slice stories to include multiple steps in a usage scenario
 - » Prefer not to make each step a story
 - » To cut back on “size” of story, limit how much of each step it implements, then add more stories to handle the additional step scopes
- ❖ **If your project is writing *technical* stories in addition to *functional* stories, you probably aren't writing *user* stories**
 - Architecture and other technical solutions should be built in support of a true user story
 - Each new story should cause project to build out more architecture

Patterns & Anti-Patterns

- ❖ **If it can't fit on a 3x5 card, the story is probably over specified**
- ❖ **Acceptance tests aren't unit tests**
 - **Acceptance tests are black box tests of whether the system meets the customer's needs, vision, expectations**
- ❖ **Acceptance tests should not assure a specific design**
 - **Not: Test the data is stored in a relational database**
 - **Not: Test the XYZ page has five tabs labeled "abc", "pqr", ...**

Scrum Process Flow



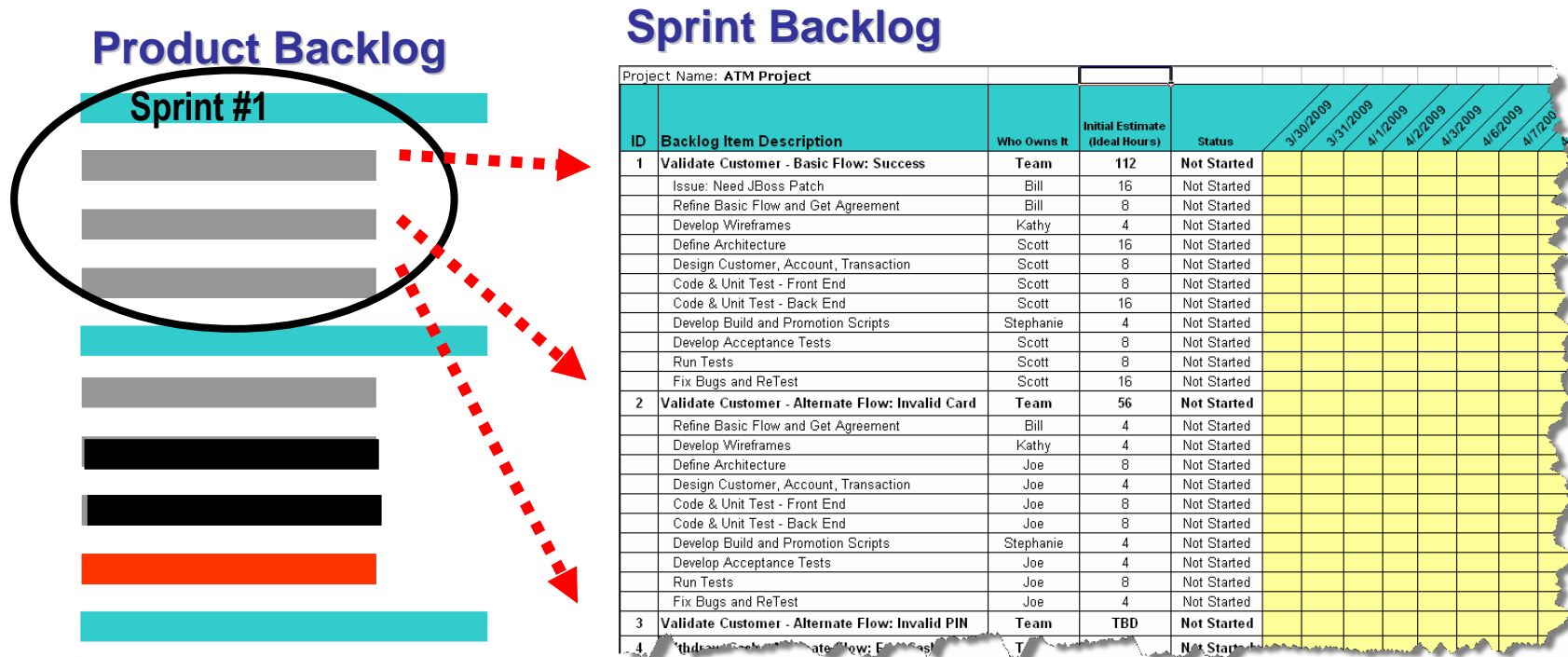
COPYRIGHT © 2005. MOUNTAIN GOAT SOFTWARE

The Product Backlog

Project Name: ATM Project			Product Owner: Pat O'Reilly				
ID	Backlog Item Description	Source	Story Point	Bus Priority	Release	Sprint	
1	Validate Customer - Basic Flow: Success	Product Owner	3	H	1	1	
2	Validate Customer - Alternate Flow: Invalid Card	Product Owner	5	H	1	1	
3	Validate Customer - Alternate Flow: Invalid PIN	Product Owner	5	H	1	1	
4	Withdraw Cash - Alternate Flow: Fast Cash	Product Owner	3	H	1	2	
5	Build audit trail of transactions according to corporate policy (POL #1227 Rev 3)	IT standard	8	M	1	1	
6	Generic error message when ATM is down (e.g., no cash in machine; one or more component - like card reader - isn't responding)	Product Owner	5	M	1	1	
7	Risk: Poor Usability (plan & run quick usability test)	Team	8	M	1	1	
8	Withdraw Cash - Alternate Flow: Request Exceeds Limits	Product Owner	3	M	1	2	
9	Withdraw Cash - Basic Flow: Specify Amount	Product Owner	3	H	1	2	
10	Withdraw Cash - Alternate Flow: Apply Fees	Product Owner	3	H	1	2	
11	Risk: servers can't handle transaction load (plan for load tests; run in next sprint)	Team		H	1	2	
12	See 10 for risk mitigation for financial transactions	IT		H	1	2	

From Product Backlog to Sprint Backlog

- ❖ The tasks required to meet the sprint's goal
- ❖ Define detailed tasks for each selected product backlog item
 - A task is 1 – 16 hours of effort, estimated by whoever does the work



The Sprint Backlog

Project Name: ATM Project															
ID	Backlog Item Description	Who Owns It	Initial Estimate (Ideal Hours)	Status	3/30/2009	3/31/2009	4/1/2009	4/2/2009	4/3/2009	4/6/2009	4/7/2009	4/8/2009	4/9/2009	4/13/2009	4/14/2009
1	Validate Customer - Basic Flow: Success	Team	112	Not Started											
	Issue: Need JBoss Patch	Bill	16	Not Started											
	Refine Basic Flow and Get Agreement	Bill	8	Not Started											
	Develop Wireframes	Kathy	4	Not Started											
	Define Architecture	Scott	16	Not Started											
	Design Customer, Account, Transaction	Scott	8	Not Started											
	Code & Unit Test - Front End	Scott	8	Not Started											
	Code & Unit Test - Back End	Scott	16	Not Started											
	Develop Build and Promotion Scripts	Stephanie	4	Not Started											
	Develop Acceptance Tests	Scott	8	Not Started											
	Run Tests	Scott	8	Not Started											
	Fix Bugs and ReTest	Scott	16	Not Started											
2	Validate Customer - Alternate Flow: Invalid Card	Team	56	Not Started											
	Refine Basic Flow and Get Agreement	Bill	4	Not Started											
	Develop Wireframes	Kathy	4	Not Started											
	Define Architecture	Joe	8	Not Started											
	Design Customer, Account, Transaction	Joe	4	Not Started											
	Code & Unit Test - Front End	Joe	8	Not Started											
	Code & Unit Test - Back End	Joe	8	Not Started											
	Develop Build and Promotion Scripts	Stephanie	4	Not Started											
	Develop Acceptance Tests	Joe	4	Not Started											
	Run Tests	Joe	8	Not Started											
	Fix Bugs and ReTest	Joe	4	Not Started											
3	Validate Customer - Alternate Flow: Invalid PIN	Team	TBD	Not Started											
4	Withdraw Cash - Alternate Flow: Error Cash	T	TBD	Not Started											

Agile Requirements Philosophy

Just Enough: Maximize the comprehensiveness of requirements specifications, while minimizing the number and formality of artifacts

Just in Time: Reduce the potential for waste in the form of unused, or outdated requirements

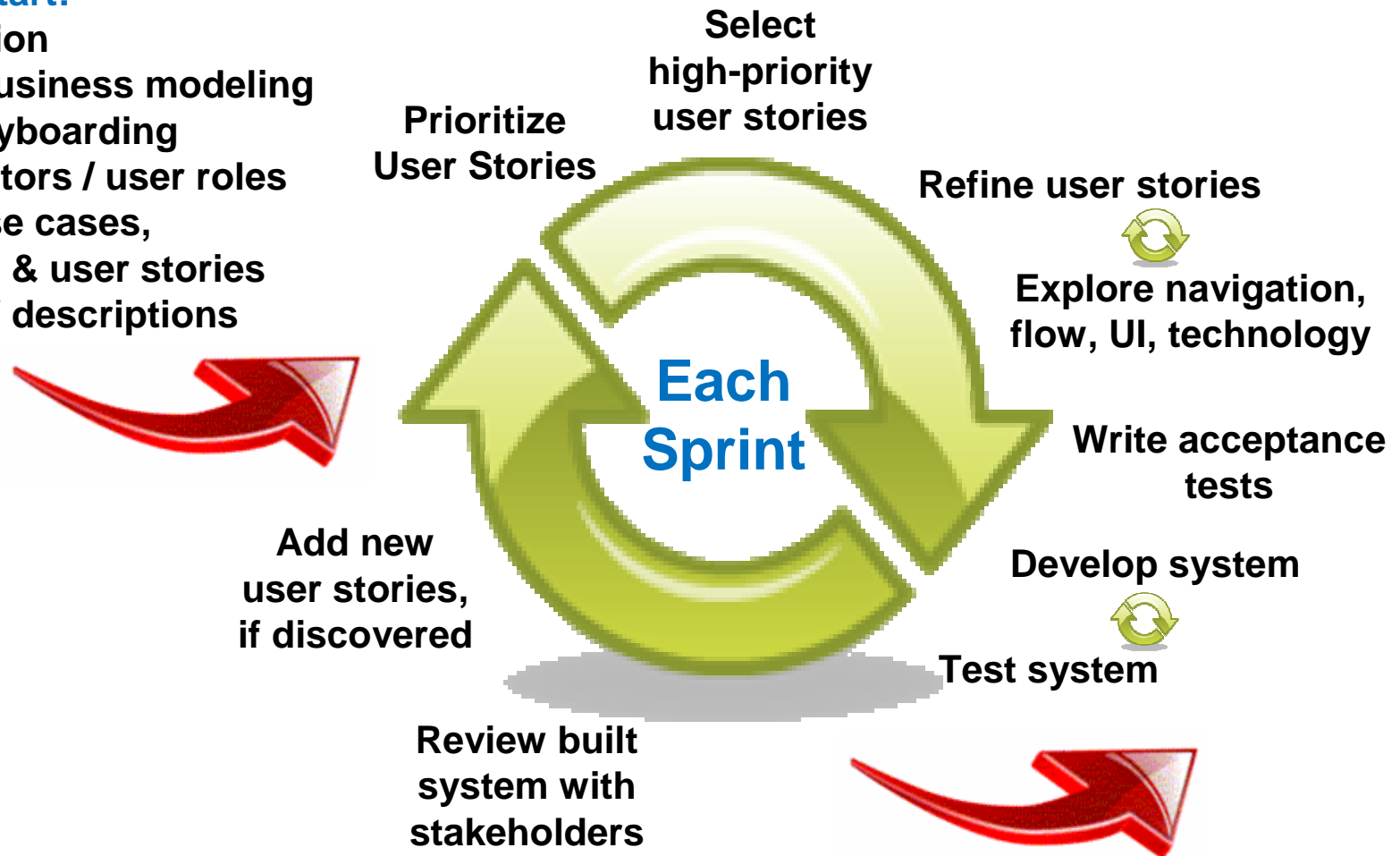
Just Because: Balance the above with your risk tolerance level, openness to change, corporate standards and policies, etc.

[Gerard Meszaros, *Agile Requirements with User Stories*]

The Scaled Requirements Process

At Project Start:

- Define vision
- Perform business modeling
- Do UI storyboarding
- Identify actors / user roles
- Identify use cases, scenarios, & user stories
- Write brief descriptions



The Scaled Requirements Process

❖ Develop Product Backlog

» *(usually achievable in 1 - 3 days, depending on project size)*

1. Develop use case model

- » Identify use case actors / user story roles
- » Identify uses of system (use cases)
- » Draw use case diagram

2. Write use case summaries

- » Brief description, Goal, Actors, Scenarios

3. Identify user stories within use cases

- » Map scenarios or portions of scenarios to user stories

4. Write user stories within use cases

- » User story goal and motivation - very brief

5. Prioritize use cases & user stories and create Product Backlog

6. Estimate top user stories in Product Backlog to determine content of next Sprint Backlog

- » Identify story acceptance criteria if clearer scope definition is needed

The Scaled Requirements Process

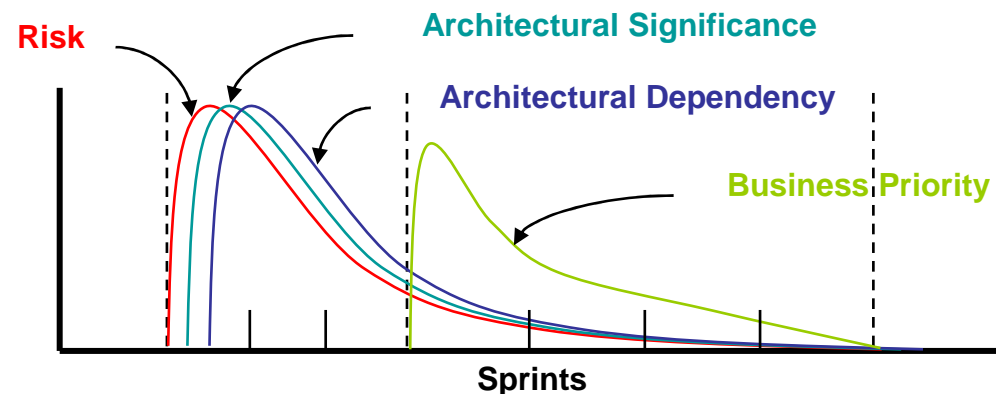
❖ Do a Sprint

1. Hold conversations with Product Owner (stakeholder representative) to negotiate details of chosen user stories
2. Discuss and document acceptance criteria for user stories
3. Optionally, explore UI navigation and content with UI storyboards
4. Optionally, explore scenario flow using flow charts / activity diagrams
5. Define non-functional requirements
6. Develop tests from acceptance criteria and good testing practices
7. Develop system to perform user stories
8. Test system
9. Add any new user stories found during conversations to use cases and Product Backlog
 - » Conversations will reveal alternatives and exceptions
 - » These become new user stories
10. Plan next Sprint

❖ Do next Sprint

Managing the Requirements Elements Used In Product Backlog

- ❖ Requirements elements used in Product Backlog
 - User Stories, User Cases, Flows of Events,
 - Non-functional User Stories, Constraints
- ❖ Scrum places responsibility for prioritizing Product Backlog solely in the hands of the Product Owner
- ❖ Other methods depend on collaboration between Product Owner, Team Lead, and Lead Architect with priority determined by
 - Risk
 - Architectural significance
 - Architectural dependencies
 - Business priorities
 - Leveling of resource loading



Risk-to-Use Case Matrix

<i>Risk Matrix</i>		User Story Identifier													
Priority	Risk Title	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	First Time Using Of Web Server Technology	X	X	X		X				X	X		X		
2	XML Interface to Banking Services			X					X						
3	Dependency on ACME Tax Wizard Enhancements		X												
4	Use of Java by Untrained Staff	X	X	X	X	X	X			X	X		X		X
5	Upgrade to Version 7.0 of DBMS			X	X			X							
6	Etc. ...														

Architecture-to-Use Case Matrix

Architecture Matrix		User Story Identifier													
Priority	Architectural Component	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Web Server	X		X		X				X	X		X		
2	Banking Services			X					X						
3	ACME Tax Wizard		X												
4	User Interface	X		X		X	X			X	X		X		
5	Security Subsystem	X		X				X			X				
6	Etc. ...														

Contact IconATG to Learn Agile Best Practices

IconATG appreciates this opportunity to present to you.
If you'd like this presentation on-site at your company, please contact us:

❖ IconATG www.iconATG.com

❖ (636) 530-7776

❖ info@iconatg.com

❖ Thank you!