Welcome to the PMI Houston Conference & Expo and Annual Job Fair 2014

- Please set your cell phone to silent mode
- There will be time at the end of this presentation for you to take a few moments to complete the session survey. We value your feedback which allows us to improve this annual event.
Agenda for Today’s Workshop

• Introduction

• Overview of Agile/Scrum

• From Vision to Acceptance Criteria
  ▪ Modeling Users & Customers
  ▪ Epics, Features & User Stories
  ▪ Elaborating from Vision to Story
  ▪ Acceptance Criteria & Testable Examples

• Q & A
An Introductory Exercise

1. Find a partner.
2. Start telling them about yourself.
3. When they hear something you both have in common, they will say “Me Too!” and find a new partner.
Problem Context: Communication

Business Wants
Development Builds
QA Tests
Striking a Balance

Business

Development
Overview

Agile & Scrum
“Agile” describes a number of related methods.

Scrum is the most popular.

• **Scrum**
  Jeff Sutherland & Ken Schwaber

• **Extreme Programming (XP)**
  Kent Beck, Ward Cunningham, Ron Jeffries

• **Kanban**
  David Anderson

• **Scaled Agile Framework (SAFe)**
  Dean Leffingwell

Source: 2010 State of Agile Development Survey, VersionOne
Dealing with Uncertainty

You don’t need agile if you know what to build, who to build it for, and how to build it.

Use agile when you have uncertainty...

What to build: End Uncertainty
How to build it: Means Uncertainty
Who to build it for: User Uncertainty

Empirical methods monitor progress & direct adaptation

Better Plan

Initial Plan
Ceremonies and Artifacts of Scrum

* Discovery is not explicitly part of the Scrum Framework
Roles

Vision

ROI/Product Backlog

Release Plans

Trade-Offs

Product Owner
Product Backlog

ScrumMaster
Values, Practices, Rules

Process

Retrospective

Blockers

Burndowns

Team
Shippable Functionality

Sprint Planning

Sprint Backlog

Daily Scrum

Deliverables
• **Product Backlog**  
  Prioritized list of all items (PBI) required to launch a successful product

• **Sprint Backlog**  
  Tasks to get committed PBIs to done within Sprint

• **Task Board**  
  Stories and tasks for the Sprint tracked from start to completion

• **Burndown/burn-up Chart**  
  Visual aid for tracking team progress and forecasting expected completion dates
The Agile “Fitness Tracker”

**Vision**

- Drive continuous improvement in a fun way
- Continuous Improvement Practice Marketplace

**Goal / Outcome**

- **Big Chunk (Epic)**
- **Smaller Chunks (Features)**
- **Detailed User Stories**

**As a ScrumMaster I can run a full-team retrospective in order to facilitate improvement**
- Describe sprint briefly
- ScrumMaster can see ratings in real time
- Team members rate outcomes on 1-5 scale

**As a Coach I can review what has worked in order to learn & share**
- Gather pluses/deltas
- Rate results of actions
- Ratings are graphed over time
Vision
Aligning Goals & Constraints
For Target Customers
Who Statement of Need
The Product Name
Is a Category
That Compelling Reason to Buy & Use
Unlike Competition / Alternative
Our Product Differentiator

As described by Geoffrey Moore in *Crossing the Chasm*
(Thanks to Gabrielle Benefield for the reference)
Simulation
Restaurant Finder
Create a Vision Poster for your simulation project with:

1. A product name;
2. A product logo;
3. A product slogan or jingle; and
4. Three (3) compelling reasons to buy your product.
Personas
Customer & User Modeling
Users interact directly with the system
They are important to understand, because:
• Knowledge of current usage patterns helps to design better, more usable systems.
• Unsatisfied users will work around the system, nullifying its advantages and eventually eliminating it.

Customers (sponsors) make buying / adoption decisions
They are also important, because:
• They have their own wish lists that may have little to do with their users’ needs.
• They make the purchasing decisions, so if they aren’t happy, you won’t get in the door.
• **Personas** represent a type of user across usage contexts.
  – One member of our current or desired audience in a tangible, less ambiguous way.
  – Provide a name, a face, and a description giving us a mental model of our users allowing us to emphasize with them and **predict how they will use our software**.

• **Level of detail**
  – Add just enough detail to aid empathy, more details can be distracting.
  – Lightweight personas will suffice for many.
User Models Summary

- **Use what works** user roles, personas, etc., without getting hung up in vocabulary.
- **Prioritize your user(s)** and prioritize stories for them.
- Post **big charts** (e.g. personas) in team room to aid empathy.
- **Focus testing and evaluation** on the right users, identifying test subjects similar to your models.
- Base models on reality (ethnography / field study):
  - Usability Testing
  - Observation
  - Interviews
  - Data Analysis
  - Feedback Forms
  - Surveys, etc.
Exercise: Create Personas

Who are your most critical personas, or early adopters?

1. **List potential stakeholders** that would represent the Customers & Users of your product.
2. **Prioritize these stakeholders** and pick two to elaborate.
3. **Create at least two personas** by writing brief stories that outline the motivations and goals of these customers or users.
User Stories & the Backlog
Working with Agile “Requirements”
The basic user story template is simplistic, it helps us remember a need while providing context.

**As a** customer who drives
**I want to** find a conveniently located branch
**So that** I can minimize driving time

**User Role, Persona**
**(Who?)**

**Desired Function**
**(What?)**

**End Result**
**(Why?)**

What is not specified?
Key Characteristics

• High-level descriptions of desired functionality and goals

• “Contracts for conversation”, not all-inclusive requirements

• Pulled into the Sprint Backlog from Product Backlog

• Contain Acceptance Criteria to define “Done”

• Vertical slices of the system’s functionality

Work in Agile projects is organized by Units of Value, rather than by Architectural Layer.
What Makes a Good Story?

**Ron Jeffries’ 3 Cs**
- Card
- Conversation
- Confirmation

**Bill Wake’s INVEST**
- Independent
- Negotiable
- Valuable
- Estimable
- Small
- Testable
If all stories are independent, any one can be picked and delivered in isolation.
For large systems this is nearly impossible!
But, minimizing, identifying and prioritizing, dependencies can result in a better backlog.
Which user story must come first?

As prospect
I want to register
So that I can execute electronic transactions

As a user
I want to pay bills online
So that I don’t have to write checks

PO: I want “Pay Bills” now!
User Stories: Invest - Negotiable

- Leaving room for give and take and decide the details when you have more context
  - High priorities stories should be more precisely defined
  - Low priority stories should have more play

**As a driver**
I want to get directions to conveniently located stores
So that I get there quickly

**Acceptance Criteria:**
- Show locations on map
- Show locations on Google Maps

Defer details until you are close to building, in this case update the acceptance criteria
User Stories: Invest - Valuable

The user story must have value to the user and to the business

- **As a user**
  - I want to have my previous orders stored in the database
  - So they will be there permanently

- **As a repeat customer**
  - I want to access old orders
  - So that I can quickly purchase the same products again

- **As a customer**
  - I want 75% off all purchases
  - So I can save money

There is clearly value to the user, but is there value to the business?
User Stories: Invest – Estimable/Small

- If you can’t estimate it, it is either too large, too vague, too risky, or some combination thereof
- Solutions include adding acceptance criteria, splitting the story, or better defining it

**As a customer**
I want a self service center  
So that I can address basic needs 24 by 7 by 365 from my computer

**Acceptance Criteria:**
1. Stop payment on check
2. Find a branch
3. Find an ATM
4. Order new checkbook
5. Get statement < 2 years old

Easier to estimate, perhaps small enough to complete in a few days

**As a customer**
I want to see my canceled checks online  
So that I can confirm transactions

**As a customer**
I want to stop payment on check so that I an prevent a payment made in error

**As a customer**
I want to find an ATM  
So that I can make deposits or with-drawals outside of banking hours

**As a customer**
I want to find a nearby branch  
So that I can conduct business in person
User Stories: Invest - Testable

- You need clarity on the story specific done criteria
- Solutions include adding acceptance criteria or better defining the story

As a registered user
I want a better looking homepage
So that I don’t have to look at something so ugly

Acceptance Criteria:
1. All text is dark color on light background (no more red on black)
2. Only two different fonts used (instead of seven)

DONE

Have to manually test, but it is clear.
Epics, Features, Stories

Product backlog
**Epics** are high-level features or activities that span Sprints, or even Releases.

- Add a Customer Center for self service.
- Improve database response time by 50%.

**Logistics**
- The PO works with stakeholders and the Team to create epics that address desired Goals.
- Epics are often defined prior to Release Planning.
- Often months of effort.
Features are tangible expressions of functionality, but still too large to build.

- As a Bank Customer, find a branch so that I can deposit checks.
- As a Shopper, set up a mobile wallet so I can pay for purchases via Near Field Communications.

Logistics
- Created by the PO with input from the team
- Often defined prior to Release Planning
- Decomposed over time to smaller Stories
- Typically weeks of effort
User Stories are ready for the Team to build.

- As a Bank Customer, find a branch near an address so that I can minimize travel.
- As a Shopper, add an account to my mobile wallet so that I can fund it.

Logistics

- Refined in backlog grooming sessions by PO and representatives from Team
- Stories should be well-defined prior to Sprint Planning
- Generally about 1-3 days of effort
Non- Functional Requirements (NFRs)

- **System-wide** nonfunctional requirements may become part of the **Definition of Done**.
- Articulated as tests
- Serve as design constraints

Search response time will not exceed 10 seconds.
All stories will meet Section 508 accessibility guidelines.

- **Story-specific** NFRs are expressed as Acceptance Criteria.
What User Stories are NOT

User Stories $\not\equiv$ Requirements

(User Stories $\Rightarrow$ Requirements)
Requirements, More than Just a Story

Context (Project Vision, Business Case, etc.)

Requirement = User Story + Conversation(s) + Acceptance Criteria + Supporting Information

Common Understanding of a Need
<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
</table>
| **A** | As a prospect  
I want to enter my billing information          | As a prospect  
I want to register  
So I can make purchases online |
| **B** | As a driver  
I want to find the store with the shortest drive time  
So I can get there quickly | As a driver  
I want to find directions to a store on Google Maps  
So I can get there quickly |
| **C** | As a repeat customer  
I want to access old orders  
So that I can rapidly purchase the same products again | As a user  
I want to have my previous orders stored in the database  
So they will be there permanently |
| **D** | As a color blind user  
I want dark text & light background  
So that I can easily read the text | As a user  
I want a nice looking site  
So my aesthetics are satisfied |
Exercise – Write User Stories

Using the Epics we provided for our awesome restaurant finder app:

- Create at least five Sprint-sized User Stories based on these Epics.
- Use the “As a, I can, so that” format for the User Stories.
Release Planning/Roadmapping

Story Maps
User Goals
• Minimize the time needed to access patient records
• Minimize the customer inputs necessary to access patient records

Persona
Night Nurse
Robin
Robin leaves for work at 6pm, after sleeping during the day. She works a 7pm-7am shift in Labor & Delivery, caring for prospective mothers and their babies. Complex computer apps make Robin grumpy.

Features & User Stories
Add comment
Another Example

Release 1
- Limit Search to one field
- Send HTML email
- Open HTML email
- Empty Deleted Items
- View Daily Format
- Create HTML appt
- Propose new time
- Add address data
- Update Address Info
- Delete Contact

Release 2
- Search attachments
- Get address from contacts
- View Weekly Formats
- Get address from contacts
- View Attachments
- Import Contacts
- Export Contacts

Release 3
- Search sub folders
- Send Attachments
- Search Calendar
- Add Attachments
- View Attachments

Thanks to Winnipeg Agilist for this image.
**Value** is influenced by many things:

- **Time sensitivity** – Build features that decay in value over time earlier.
- **Uncertainty & Risk** – Use “spikes” to test market or technical viability for critical, risky features.
- **Size** – All else being equal, do the shortest first.
- **External Dependency** – Third party or support group dependencies can be immutable constraints.
Plan your **first few Releases with a Story Map.**

- **Place Epics** or **High Level Activities** at top, in order of their natural workflow as appropriate.

- **Place User Stories** underneath the Epics that they support, from top to bottom by value.

- **Group the stories** for your first Release (MVP) and subsequent Releases, describing the targeted benefits of each at a Roadmap level.
Product Backlog Management

Refining, Accepting & Testing
Adding User Stories
• Anyone can suggest backlog items
• Product Owner prioritizes them

Estimating & Elaborating
• High-priority items are estimated and described most precisely, since they will be worked on first
• Low-priority items are generally estimated and described coarsely

Prioritizing
• Ordering is driven mainly by business value and risk reduction.

## Product Backlog Essentials

<table>
<thead>
<tr>
<th>#</th>
<th>Backlog Item</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Create login screen</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Allow user to browse recently viewed items</td>
<td>8</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>Add personalization</td>
<td>30 (or so)</td>
</tr>
</tbody>
</table>

**High priority** items are better defined

**Low priority** items are often “epics”
Progressive Elaboration

### Ideation
- Market Trends
- Prototypes
- Focus Groups
- User Experience
- Basic Workflows
- Vision
- Business Outcomes
- Release Timing and Goals
- Product Architecture
- Epics and Features

### Maturation
- User Story Decomposition
- User Story Maturation
- Acceptance Criteria
- Test Cases
- Dependencies
- Story Mapping
- Prioritization
- Epic Estimation
- Backlog Development

### Execution
- Sprint Planning
- Sprint Estimation
- Daily Standups
- Software Development
- Testing
- Burndowns
- Documentation
- Product Demos
- Retrospectives

For each phase:
- **3-4 Sprints ahead or more**
  - Marketing/Sales, Product Management, Product Owners, Architects
- **~1-2 Sprints ahead**
  - Product Owners, Architects, Dev Leads, QA Leads, UX/Analysts
- **Current Sprint**
  - Leads, UX/Analysts, Dev Team Members
Maturation of a User Story

User Stories
Created during upfront and ongoing Discovery

As a user
I want to create an account
So that I can shop online

Priority - 1 (High)  Estimate - 5 Points

Acceptance Criteria
Sprint Pre-Planning (Backlog Grooming)

• Phone # in US format, contains no alpha characters, minimum 10 digits
• First name, Last name and email address required
• Email specified in standard format

Testable Examples (ATDD)
Sprint Pre-Planning (Backlog Grooming)

<table>
<thead>
<tr>
<th>Name</th>
<th>Phone</th>
<th>Email</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Smith</td>
<td>215-555-1212</td>
<td><a href="mailto:jsmith@ls.com">jsmith@ls.com</a></td>
<td>TRUE</td>
</tr>
<tr>
<td>Smith</td>
<td>215-555-1212</td>
<td><a href="mailto:jsmith@ls.com">jsmith@ls.com</a></td>
<td>FALSE</td>
</tr>
<tr>
<td>John</td>
<td>215-555-1212</td>
<td><a href="mailto:jsmith@ls.com">jsmith@ls.com</a></td>
<td>FALSE</td>
</tr>
<tr>
<td></td>
<td>215-555-1212</td>
<td><a href="mailto:jsmith@ls.com">jsmith@ls.com</a></td>
<td>FALSE</td>
</tr>
<tr>
<td>John Smith</td>
<td><a href="mailto:jsmith@ls.com">jsmith@ls.com</a></td>
<td>TRUE</td>
<td></td>
</tr>
<tr>
<td>Smith</td>
<td><a href="mailto:jsmith@ls.com">jsmith@ls.com</a></td>
<td>FALSE</td>
<td></td>
</tr>
</tbody>
</table>

Sprint Tasks
Sprint Planning

• Design UI Mock Up
• Write online help text
• Develop CSS/HTML
• Develop validation criteria
• Create database tables
• Code test fixtures
• Code & Test
The tactical act of getting a story ready is often performed as a **two sprint look-a-head** by an amigos team.

<table>
<thead>
<tr>
<th>Sprint n</th>
<th>Sprint n + 1</th>
<th>Sprint n + 2</th>
</tr>
</thead>
</table>
| Select User Story 999 for Sprint n + 2  
Re-estimate it, sharpen story & acceptance criteria | Create testable example and other supporting material for 999  
Get sign off from external parties | Develop User Story 999 |

The PO and 3+ Amigos look-a-head and select story 999 for inclusion for Sprint n + 2. They do cleanup on the story.

The 3+ Amigos further support the story and the PO gets appropriate sign offs.

Story 999 makes it’s way into a sprint and it is built.
1. To Demonstrate Progress: UX with no validation or save
2. CRUD: Create, report, update & delete (e.g. split of manage)
3. Basic to Advanced: Sort by one field (name), sort by any one field (name, date, etc.), sort by a combination of fields
4. Use Case scenarios: Happy path, alternates, exceptions
5. Workflow steps: Find book, see details, purchase
6. Importance: Credit card, split across cards, automatic billing
6. UI complexity: Manual coordinates, interactive web map
7. Spike and Build: Research credit card processing, implement
<table>
<thead>
<tr>
<th></th>
<th>As a ...</th>
<th>I want to manage my widgets</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>As a report viewer</td>
<td>I want to filter my report by any combination of columns</td>
</tr>
<tr>
<td>2</td>
<td>As a possible room renter</td>
<td>I want to find and book a room</td>
</tr>
<tr>
<td>3</td>
<td>As a customer</td>
<td>I want to pay electronically</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>
| **5** | As a low budget vacation traveler  
I want to find flights using a range of dates |   |
| **6** | As a credit card purchaser  
I want to pay by Amex, MasterCard, Visa or Discover |   |
| **7** | As a frequent user  
I want to personalize my experience |   |
3+ Amigos (BA, QA, Dev) can serve as a readiness team to look ahead and ensures we are:

1. Ready to prioritize
2. Ready to estimate / right size
3. Ready to build
**When is a Story “Ready”?**

**Definition of “Ready”**

- Choose the **few items** that your team finds most useful in Sprint Planning.
- Confident and **quick Sprint Planning** and **smooth Sprints** that produce polished results are your goals.

<table>
<thead>
<tr>
<th>✔️ Interaction Diagrams</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔️ Prototypes</td>
</tr>
<tr>
<td>✔️ Wireframes</td>
</tr>
<tr>
<td>✔️ Sample Data</td>
</tr>
<tr>
<td>✔️ Testable Examples</td>
</tr>
<tr>
<td>✔️ Acceptance Criteria</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>✔️ Small Enough</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔️ Agreement from other teams</td>
</tr>
<tr>
<td>✔️ Approvals (Compliance, Security, Brand Mgmt, etc.)</td>
</tr>
<tr>
<td>✔️ Dependency List</td>
</tr>
<tr>
<td>✔️ Stakeholder signoff</td>
</tr>
</tbody>
</table>
When is a Story “Done”?

Definition of “Done”

- A **shared definition** and compact between Teams and Stakeholders
- Denotes what stories require to be **accepted**
- Ideally represents “**potentially releasable**” or even released state

- ✔ Acceptance Criteria are met
- ✔ Cleared by QA
- ✔ Accepted by Tactical PO
- ✔ Accepted by Strategic PO
- ❑ Live for A/B Testing
- ❑ Final Deployment
- ❑ Training Script

- ✔ Pair reviewed
- ❑ Peer Reviewed
- ✔ Integrated
- ❑ Lightweight usability tested
- ✔ Automated testing in place
- ✔ User documentation created
- ✔ Ops documentation created
“Done Done” at Release Level

What are the minimum criteria for each Release?

- Testing/quality targets
- Performance targets
- Operational (e.g. sales/marketing) deployment goals
- Required documentation & artifacts
- Regulatory compliance targets

Example Release Criteria

- System response on all level 1 functions within 5 seconds
- No Severity 1-3 bugs in Firefox 2+, Chrome, IE 7+ or Safari 3+
- No Severity 1 or 2 bugs found during final month
- Full compliance with accessibility guidelines in Section 508
As a user
I can cancel a registration
So that I don’t have to pay

- Premium member can cancel the same day without a fee
- Non-premium member is charged 50% of first day for a same-day cancellation
- Email confirmation is sent to members primary and secondary email addresses
- Hotel is notified of any cancellation
Modern Agile Acceptance Model

- Conditions of Satisfaction – Broad Terms
- Acceptance Criteria – Further Refined
- Examples – Actual scenarios or data
- Executable Examples – Ready to automate
Testing a **Registration Function**

- What constraints should we impose?
- Business stakeholders and PO agree that passwords should not be easy to crack.

---

**Please fill in to register.**

<table>
<thead>
<tr>
<th>Email Address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

[Register]
Acceptance Criteria, the Details

The PO works with testers and developers from the team, business stakeholders, users or SMEs to come up with this definition of “not easy to crack”:

- Must be at least 8 characters and no more than 12
- Must contain only alphanumerics and the period
- Must contain at least one digit
- Must contain at least one alpha character

Please fill in to register.

<table>
<thead>
<tr>
<th>Email Address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>????</td>
</tr>
</tbody>
</table>
**Acceptance – Examples Written**

- Examples, Making it *Real*
- Actual examples are the ultimate way to communicate requirements

<table>
<thead>
<tr>
<th>Password</th>
<th>Expected</th>
<th>Expected Message</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>abc123</td>
<td>Invalid</td>
<td>You password must be at least 8 characters long, and no more than 12 characters long.</td>
<td></td>
</tr>
<tr>
<td>abcdefghiu</td>
<td>Invalid</td>
<td>Your password must contain at least one character and one number.</td>
<td></td>
</tr>
<tr>
<td>1aaaaaaaaaa</td>
<td>Valid</td>
<td></td>
<td><em>Why valid?</em></td>
</tr>
<tr>
<td>ajx972dab</td>
<td>Valid</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Team Members with testing and development expertise take the lead on creating examples, with the Product Owner verifying
- Based on these examples, Acceptance Criteria can be refined
- These criteria act as good starting test cases too
• Executable Example, Making it *Repeatable*
• Examples that can be executed are the final step

Given the “Unregistered User” user has navigated to the “register” page

When entering “newuser” in the “Username” field
And entering “abc123” in the “Password” field
And entering “abc123” in the “Confirm Password” field
And pressing the “Register” button

Then the text “Thank you for Registering” should appear on page
And the URL should end with “use/accountPage”
A practice in which the whole team collaboratively discusses acceptance criteria, with examples, and then distills them into a set of concrete acceptance tests before development begins.

- Elisabeth Hendrickson
1. Elicit details from the business stakeholder(s) about their expectations

2. Distill acceptance criteria into automatable tests expressed in a natural language

3. Wire the tests to SUT with “glue” code (as part of implementation)
Automated Acceptance Tests

Tool
(Cucumber, SpecFlow, FitNesse)

Specification expressed in common language

“Glue” code that ties specification to SUT
ATDD Cycle

Acceptance Test Driven Development (ATDD) Cycle

Discuss

Distill

Demo

Develop

ATDD cycle model developed by James Shore with changes suggested by Grigori Melnick, Brian Marick, and Elisabeth Hendrickson.
Exercise – Write acceptance tests

Create **testable specifications**

Using the User Stories you defined

- **Write acceptance criteria** for each story
- **Create at least two scenarios** to act as examples and tests, using this format:

**Scenario: Recent Account Activity**

*Given* I am a registered user “Jsmith”  
And I am logged in with password “xyx123”  
And I have had account activity in the last 45 days  

*When* I click the “Recent Activity” button  
*Then* I should see the “Account Activity” Page  
And I should see a list of my activity over the last 45 days
Common pitfalls with writing user stories

- Forgetting about the User
- Too much Detail
- Lack of Conversation
- No Acceptance Criteria
Contact Information

Speakers: Susana Esparza & Raj Indugula
Company: LitheSpeed
Website: www.lithespeed.com
Phone: 703.745.9125
E-mail: susana.esparza@lithespeed.com
raj.indugula@lithespeed.com

Thank You
Thank you for attending this session. We hope you found this presentation added value to your knowledge of Project Management.

- Take a few moments to complete the Session Survey. We appreciate and value your feedback.
- Hand in your completed survey to **Registration**, you will receive a free raffle ticket for one of the drawings to be held in the Vendor Expo (see Conference Program Guide for details).