Agile Planning with User Stories

Part of the “Agile Bootcamp” Stage

Gerard Meszaros
Agile2011@gerardm.com

Note to Reviewers

• This slide deck is a starting point and will be evolved extensively if this session is accepted.

• Thanks,

• Gerard
Overview

• What’s A User Story?
• Why Do Things Differently?
• How Do We Use User Stories?
• Where Do Stories Come From?

A Note on Terminology

• “User Story” is the XP term
• Each agile method has it’s own terms
• Equivalent terms exist in most other agile methods
  E.g.
  – Scrum:  Product Backlog Item
  – FDD:  Feature
• All agile methods rely on some sort of feature-based requirements
• I use “User Story” and “Feature” interchangeably
A User Story is ...

... a unit of work to develop functionality that:
- Is very specific (has concrete examples)
- Provides value to the customer
- Can be tested independently of other (later) features
- Can be finished in a single iteration

... consists of 3 major components:

- Story Card
- Conversation
- Story Tests

Sample User Stories

A user can:
- Generate an invoice for a subscription charge.
- Generate an invoice that includes usage-based charges.
- Finalize a customer’s invoice and send it to them.
- Generate invoices for all customers.
- Generate invoices for all customers in a single billing cycle.
- Maintain customer data
- Select customers for invoice finalization using drag & drop.

An invoice cannot be sent to a customer until it has been finalized. An invoice that has been finalized cannot be regenerated or modified in any way.

Each story adds value as soon as it is “Done”
Sample Story Card

A user can generate invoices for all customers in a single Billing Cycle

Sample Conversation

What’s a Billing Cycle

A Billing Cycle is a way to group Customers who all have their invoice generated on the same day of the month.

That’s right.

In theory it could be. But in practice, we find one cycle each week is enough ...

Can more than one Billing Cycle have the same day of the month associated with it?

No, but that’s an interesting idea. We could use that to ...
Sample Story Tests

A user can generate invoices for all customers in a single Billing Cycle.

- **Verify Basic Functionality**
  - Define two Billing Cycles A and B
  - Define 2 customers A1 and A2 and add them to B
  - Define another 2 customers B1 and B2 and add them to B
  - Request invoice generation for all customers in Billing Cycle A
  - Verify invoices were generated for A1 and A2
  - Verify no invoices were generated for B1 and B2.

- **Verify Other Functionality:**
  - Etc.

Overview

- What’s A User Story?
- Why Do Things Differently?
- How Do We Use User Stories?
- Where do Stories Come From?
What Makes a Project Agile?

- **Deliver Value Early**
  - Get ROI earlier; don't “front-end load” the project
- **Deliver Value Often**
  - Many opportunities to learn and change minds
- **Respond Easily to Changes in Requirements & Encourage**
- **Be Able to do this in a Sustainable Fashion**
  - Reduce the inertia caused by hard-to-change artefacts

> “Plans are useless but planning is essential”
> General Dwight D Eisenhower

Agile Requirements Philosophy

Do as little work as possible without sacrificing quality

**Just Enough:** Ensure the requirements get communicated while minimizing the number and formality of artifacts

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

**Just Because:** Balance the above with: risk tolerance level, openness to change, corporate standards, politics, etc.
Sample Traditional Requirement:

2.2.1 Invoicing

A user can generate an invoice (consisting of one or more subscription or usage charges) for one or all customer. The user can select the customers whose invoices are to be generated using a multi-selection list box or using Add/Remove buttons to move the customers from the All Customers pane to the Selected Customers pane. The system should remember the last set of customers for whom an invoice was generated.

An invoice cannot be generated for a customer until the sales manager has approved them. An invoice cannot be generated for a customer until all mandatory data elements have been provided. These include name, contact information (mailing address, phone #), title, and company name. Customers can be created with as little as just a name but they cannot be invoiced.

When the user is satisfied with the invoice for a customer, they may finalize it and then send it to the customer. Once finalized, the invoice cannot be regenerated or modified in any way.

A Customer ...

Sample Requirement As Use Cases

- Manage User
- Manage Customer
- Manage Rates
- Generate Invoices
- View Invoice
- Finalize Invoice
- Send Invoice

Each Use Case has many alternate paths some with higher value than others.

Some paths are understood now while others may require further thinking.

Most Use Cases cannot be tested by themselves

Use Cases are the wrong granularity. Too big yet too small!
Agile Project Qualities

Agile Projects

• Incremental Development
• Time-boxed Iterations
• Maximize value delivered per $
• Prefer direct communication over documentation

User Stories

• Incremental Requirements
• Small enough to fit
• Self-Consistent value
• Story Cards + Conversation + Story Tests

User Stories are a Planning Mechanism

Not a requirements capture mechanism!
User Stories are a Planning Mechanism

Stories are:

• A way for Customer to tell Devt what they want
• A way to talk about and remember what requirements exist,
  – But not the details of the requirements!
• A way to talk about what work needs to be done and what value it provides.
• A way for the customer to drive the project plan and to monitor progress transparently
• A way to decide what is in/out of a scope for a Project/Release/Iteration

User Stories an Alternative to WBS

• Stories are an alternative to Work Breakdown Structure (WBS) of traditional project plans:

  “Dark” Period
  
  Activity
  Analysis Design Coding Testing
  
  Functionality
  Module Module Module
  Feature Feature Feature
  
  • Feature Breakdown Structure:

  Activity
  Analysis Design Coding Testing
  
  Functionality
  Story 1 Story 2 Story 3

• On agile projects, we define the project plan in terms of what will be delivered rather than what work steps will be performed
Overview

• What’s A User Story?
• Why Do Things Differently?
• How Do We Use User Stories?
• Where do Stories Come From?

User Stories in Process

User stories are the starting point for developing everything else:

- User Stories
- Story Tests
- Unit Tests
- Usable Software
- Tasks
- Development
- Other Artefact
- Release Planning Game
- Iteration Planning Game
- Many, many conversations

- User Manuals
- Design Documents (Just Enough)
- Use Cases (Just Because)
Stories & Planning

- Stories are selected for a release or iteration
  - Based on ROI:
    - Business value delivered, and
    - Estimated cost
- Business value can be any of:
  - Reduced effort/cost
  - Improved quality or consistency
  - Reduced risk
  - Increased flexibility
  - Improved usability

Examples of Business Value

- Automate a previously manual process
- Do something with less effort
- Reduce the possibility of human error
- Ensure consistency of process
- Enforce business rules
- Provide alternative ways to do something
- Provide audit trail of business activity
- Better integration between systems
- Keep stakeholders informed of progress

Used for Prioritization during Planning
Stories & Estimation

- Planned functionality is costed per user story
- We estimate per story, not per work type
  - Because that is what the customer understands and values
  - Because stories are what are used for planning
- If estimating is too hard or not possible,
  - split the story into smaller stories that can be estimated more easily
  - or do an experiment if it is a technology or skills issue

Estimating User Stories

- Can use whatever estimating methodology works for you.
- Main criteria:
  - Quick (a few minutes per User Story)
  - Team-based (nobody estimates for someone else)
  - Good enough accuracy (It's an estimate, not a certainty!)
What an Estimate Includes:

• The design effort plus any design artifacts we have agreed to produce:
  – Working code
  – User Interface Design (if needed)
  – User Documentation (if needed)
  – “As built” requirements documentation (e.g. Use Cases)
  – Story Tests (the Confirmation that we’re done)
  – Unit Tests (our safety net for when things change)

Common Issues with Estimating

• Lack of estimating experience
  – Addressed through estimating frequently
  – Practice makes perfect

• Dominant personality skews estimates
  – Development lead estimates;
  – Everyone else concurs

• Insufficient information to estimate with
  – Split stories if too big
  – Defer “mushy” stories or parts thereof
  – “Buy Information” if technology not understood
Planning Poker

- Customer describes User Story
- Team asks questions
- Facilitator asks “Are you ready”
- Team members each select an estimate card
  - ½, 1, 2, 4, 8, 16, 32
  - ½, 1, 2, 3, 5, 8, 13, 21, 34
- Facilitator says “Presto Revealo”
- Team members show their cards
- If everyone agrees, estimate is done
- Otherwise
  - Lowest & Highest estimates are explained, discussed, and process is repeated.

Why This Works:

- The law of large numbers
  - Many small estimates for small features are more accurate than a single large estimate
- The wisdom of crowds
  - Many people each estimating the same feature are less likely to overlook something.
  - Discussion trigger by disagreement is where real improvement of estimates occurs.
**Why This Works (2)**

- **Comparative Estimating**
  - New estimates are based on similar work estimated in previous iterations
    - **Component-based reuse of estimates**
  - Limited range of numbers ensures discussion only occurs when needed
    - 8 vs. 9 is noise; 1 vs 2 is significant

- **Practice Makes Perfect**
  - Estimating iteration after iteration

---

**Overview**

- What’s A User Story?
- Why Do Things Differently?
- How Do We Use User Stories?

- Where do Stories Come From?
**Where Do Stories Come From?**

- **Written by the customer, not the developers**
  - Though development can make suggestions
- **Before/during Planning Game or story workshop**
- **Based on UI Prototypes or “Story Board”**
  - Iteration 0 Activities
- **Use StoryOTypes to “right size” the stories**
  - Small enough to build in a few days
  - Large enough to provide value

---

**Don’t Throw the Baby Out With The Bath Water!**
Use Whatever Techniques Work For You

---

**No BDUF**

- **eXtreme Programming eschews “Big Design, Up Front” (BDUF)**
  - Because “Ya Ain’t Gonna Need It” (YAGNI)
- **Favours “Emergent Design”**
  - An attempt to avoid over-engineering and “Front-end loading” the project with activities that
    - Take a lot of time
    - Don’t deliver value to the customer
  - Supported by frequent refactoring
- **Often taken as an permission to not do any planning**
Issues with “No BDUF”

• Teams function better when they have a common goal
  – Putting on the no BDUF blinkers obscures the goal

• Hard to estimate “whole product” cost
  – Need some idea of what “whole product” is

• Refactoring User Interfaces is problematic
  – Can be harder than refactoring business logic
  – Users don’t like frequent changes to their application

Reality

• Very few projects start with
  – “Here’s a bunch of developers; just start coding”

• Typical process:
  – Business sponsor defines product vision
  – Gets buy-in from various stakeholders
  – Gets some kind of rough costing from IT
  – Gets approval for funding

• This typically takes months!

Implication: Lot’s of Time to Define Product and Populate Story Backlog before 1st Iteration
Priming the Backlog – Iteration 0

Goals:
• Define “Minimum Marketable Product”
  – List of Major Feature Areas
  – Overall User Interface & Navigation Structure
• Define First Cut at Release Plan
  – What Major Feature Areas in which Release
  – What parts of the UI in which Release

Means:
• Product Envisioning Workshop
• Product Design Workshop
• Release Planning Workshop

Product Envisioning Workshop

• Can Use Role Modeling to identify stakeholders
  – Stakeholders and their goals
    • E.g. Personas per Alan Cooper
    • E.g. User Roles per Larry Constantine
    • E.g. Stakeholders per Alistair Cockburn
  – User Task Model
    • E.g. User Tasks per Alan Cooper
    • E.g. Essential Use Cases per Larry Constantine
    • E.g. Lightweight Use Case modeling per Alistair Cockburn
• Different stakeholders should participate
  – End-users identify End User Stories
  – Operations people identify Operations User Stories
  – Augmented by usability analysts & facilitator
Product Design Workshops

Goal: Come up with overall Product Design
   – Not Software Design/Architecture!
• Overall Look & Feel
• UI Navigation Architecture
   – “Story Board” for most common tasks
• Common Screen Layout
   – Wire Frames
• Paper Prototyping of Major Screens
   – Show how major use cases will function

Paper Prototypers in Action!

• picture of Janice and Gerard build paper prototype with stuff spread out on table between them
Concern: Planning in a Vacuum

- The longer we work in a vacuum, the more we are deluding ourselves.
- Agile emphasises rapid feedback; what’s our feedback during planning?

Wizard of Oz Testing

- A way to get rapid feedback on UI designs
- Typical Test Session:
  - Tested with pairs of real end users (co-discovery)
  - Perform 3 or 4 tasks defined by business SME
  - 3 Test sessions of 1 hour each
- Mechanics of Test Session
  - 2 developers playing computer & coprocessor
  - 2-3 observers (developers + business)
  - Facilitator to interact with test subjects
Wizard of Oz Testing

- Picture of user writing on paper prototype while computer gets ready to lay down next sheet. Observers in the background taking notes.

Release Planning Workshop

Goal: Decide what gets done in which Release

- Break down Major Feature Areas into Stories
- Ensure everything on UI has a User Story
- Decide what’s in 1st Release and what’s not
  - Detailed Scoping exercise
  - Define “Minimum Marketable Product”
- Define dates of 1st Release
- Estimate Cost of 1st Release
  - Requires first cut at Story estimates
  - Ideally involve the developers at this point
Agile Planning with User Stories

Recall: **Story Cards are**

- A “Promise for a later conversation”
- The traditional “low tech, high touch” way of capturing User Stories
- Highly participative:
  - Easily moved around, organized by team

- Physical cards are optional!
  - Distributed teams have trouble using story cards
  - Many teams use online tools such as XPlanner
- Concept of planning token is not optional!

An invoice cannot be generated for a customer until the sales manager has approved them.

Agile Planning with User Stories

**Getting Story Granularity Right**

- Smaller Stories facilitate release/iteration planning
- Right-sizing is major challenge for agile teams
- Split a user story into smaller stories when:
  - Too large to build in one iteration
  - Too large to estimate confidently
  - Various parts have different business value
  - Various parts have different likelihood of change
“Right-Sizing” Using StoryOTypes

StoryOTypes are “Story Stereotypes”:
• Heuristics for decomposing requirements into smaller user stories
• A set of guidelines for identifying different kinds of functionality in stories
• A set of guidelines for combining story fragments into cohesive stories

Four Common Storyotypes

• New Basic Functionality
  – Automates some main scenario with minimal user interface
• Alternate Scenario
  – Automates alternate scenario (variation of functionality)
• Business Rule
  – Defines some rule and provides handling for when rule is broken
• User Interface Variation
  – Changes the user interface in some way (usually makes it fancier, but could just provide a different way to do the same thing)
New Functionality Stories

- Generate a very simple invoice consisting of a single subscription charge for a single customer. (Bootstrap Story)
  - Simple UI (Enter Customer # and press submit)
  - No data validation or rules enforcement
  - Verify by querying database or printing it out
- View a customer’s (generated) invoice.
- Finalize a customer’s invoice and send it to them.
- Create/View customer data

Some value is provided as soon as these are “Done”

Alternate Scenario Stories

- Generate an invoice that includes usage-based charges.
  - The usage data is read in from a flat file and the usage is charged at a rate of $1 per unit of usage.
- The usage rate can be set via a user interface.
  - Generate the invoice and view it to verify the rate is being applied correctly.
- Generate invoices for all customers.
- Select a set of customers whose invoices are to be generated.
  - Ticking check boxes and pressing Submit
- Remember the last set of customers for whom an invoice was generated.
  - Show the selection screen and have them already selected.
UI Variation Stories

- Generate (or view) the Invoice for a single customer by clicking on a hyperlink
- Generate (or view) the Invoice for a single customer by clicking on a custom icon
- Select customers using:
  - Simple dual list box with add/remove buttons.
  - Drag & Drop into a “Selected Customers” listbox.
  - Add Customers to a “Shopping Cart”

Business Rule Stories

- An invoice cannot be sent to a customer until it has been finalized.
- An invoice that has been finalized cannot be regenerated or modified in any way.
- An invoice cannot be generated for a customer until the sales manager has approved them.
  - This also requires a simple UI to approve the customer (probably described in the Maintain Customer use case.)
- An invoice cannot be generated for a customer until all mandatory data elements have been provided.
  - These include name, contact information (mailing address, phone #), title, and company name. Customers can be created with as little as just a name but they cannot be invoiced.
- Only the sales manager can approve the customer.
  - This implies some kind of login capability so that the system can be aware of who is using the system. Authentication (that is, security) could be another story.
Now What?

- Keep These as Our Stories or Combine Them?
- Depends on Size, Priority and Stability
- Need to combine them if they cannot be tested separately.
- May choose to combine them if they are too small to bother managing separately, and
  - Same degree of importance, confidence, stability

Conclusions

- User Stories are a way to enumerate requirements
  - Right size for planning incremental dev't
- User Stories are the result of Planning
  - They don't just appear out of thin air!
- Every story has ROI:
  - Business value (Return)
  - Estimated cost (Investment)
- Story Cards are just a planning token; the real requirements come later
  - In conversations with the customer,
  - Captured as Story Tests
Further Reading - Agile

- **Agile Requirements**
  - Tailoring the Functional Requirements Specification Process to Improve Agility. Tutorial by Jennitta Andrea, Gerard Meszaros
  - Slides available by request.

- **Managing the Bootstrap story in an XP Project**

- **Storyotypes - Using Stereotypes to Split Bloated XP Stories**
  - Paper by Gerard Meszaros presented at XP/Agile Universe 2004
  - [www.clrstream.com/downloads](http://www.clrstream.com/downloads)

- **Structuring Use Cases with Goals**
  - [http://alistair.cockburn.us/crystal/articles/sucwg/structuringucswithgoals.htm](http://alistair.cockburn.us/crystal/articles/sucwg/structuringucswithgoals.htm)

- **Use Cases 10 Years Later**
  - [http://alistair.cockburn.us/crystal/articles/uctyl/usecasesetenyearslater.htm](http://alistair.cockburn.us/crystal/articles/uctyl/usecasesetenyearslater.htm)

- **User Stories Applied**
  - Book by Mike Cohn published by Addison Wesley 2004

---

Further Reading - UxD

- **About Face 3rd Edition**
  - The Essentials of Interaction Design
  - by Alan Cooper, April 2007

- **Software for Use**
  - A Practical Guide to the Models and Methods of Usage-Centered Design
  - Larry Constantine, April 1999

- **Paper Prototyping**
  - The Fast and Easy Way to Design and Refine User Interfaces
  - Carolyn Snyder, April 2003
Questions?

Agile Planning with User Stories

Gerard Meszaros
ClearStream Consulting
CalgaryTechFest@gerardm.com