Agile Requirements with User Stories

Part of the "Intro to Agile" Track

Gerard Meszaros
ClearStream Consulting
gerard@clrstream.com

Overview

- **What's A User Story?**
- **Why Do Things Differently?**
- **How Do We Use User Stories?**
- **Where do Stories Come From?**

clearStream

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**

clearStream

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:

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.

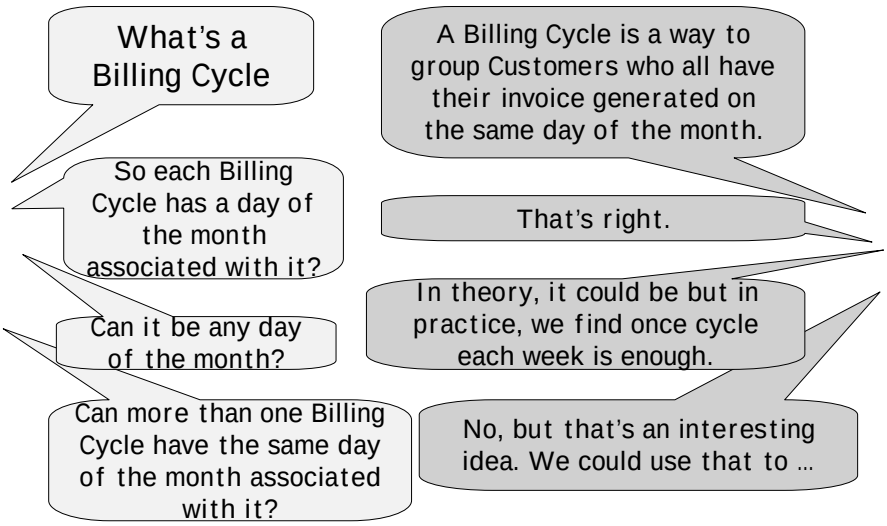
Each story adds value as soon as it is "Done"

Sample Story Card

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

clearStream

Sample Conversation



clearStream

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

Agile Requirements Philosophy

Do as little work as possible without sacrificing quality

Just Enough: Maximize the comprehensiveness of requirement specifications, 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**

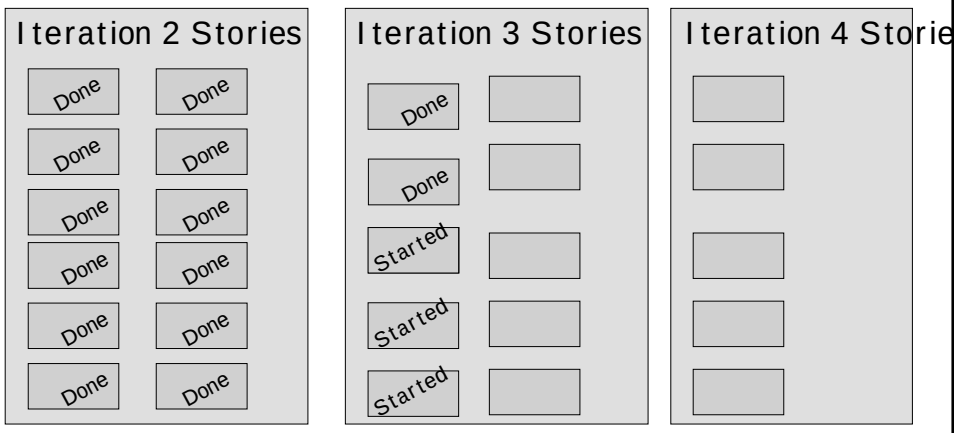
User Stories

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

clearStream

User Stories are a Planning Mechanism

Not a requirements capture mechanism!



clearStream

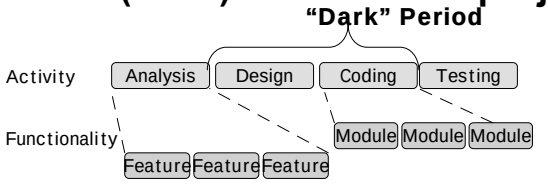
User Stories are a Planning Mechanism

Stories are:

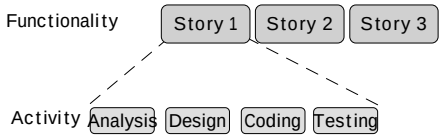
- **A way for Customer to tell Devt what they want**
- **A way to talk about and remember what requirements exist, not the details of the requirements**
- **A way to talk about what work needs to be done and what value it provides.**
- **A way for the customer to drive the project plan and to monitor 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:**



- **Feature Breakdown Structure:**



- **On agile projects, we define the project plan in terms of <u>what will be delivered</u> rather than <u>what work steps will be performed</u>**

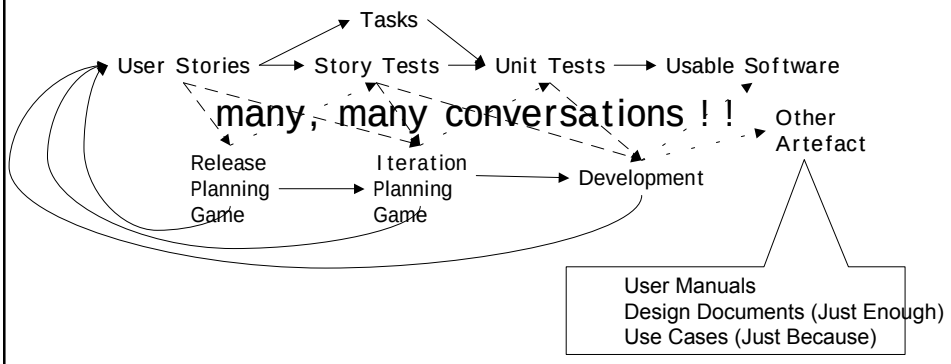
Overview

- **What's A User Story?**
- **Why Do Things Differently?**
- **How Do We Use User Stories?**
- **Where do Stories Come From?**

clearStream

User Stories in Process

User stories are the starting point for developing everything else:



clearStream

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

clearStream

* 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

clearStream

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

clearStream

Where Do Stories Come From?

- **Written by the customer, not the developers**
  - Though development can make suggestions
- **Before/during Planning Game or story workshop**
- **Use Role Modeling to identify stakeholders**
  - E.g. Lightweight Use Case modeling
  - Stakeholders and their goals
- **Different stakeholders should participate**
  - End-users identify End User Stories
  - Operations people identify Operations User Stories
- **Usee StoryOTypes to "right size" the stories**

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

clearStream

Overview

- **What's A User Story?**
- **Why Do Things Differently?**
- **How Do We Use User Stories?**
- **Where do Stories Come From?**

clearStream

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

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

- **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!**

clearStream

Getting Story Granularity Right

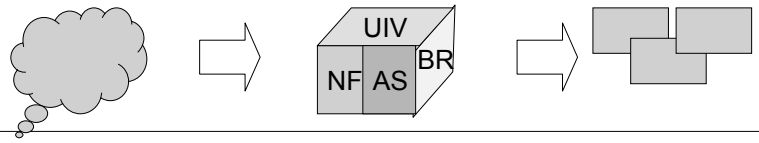
- **Smaller Stories facilitate release/iteration planning**
- **Right-sizing is major challenge for agile teams.**
- **Split a user story into smaller stories when:**
  - Too large to build in one iteration
  - Too large to estimate confidently
  - Various parts have different business value
  - Various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

clearStream

"Right-Sizing" Using Storyotypes

Storyotypes are "Story Stereotypes":

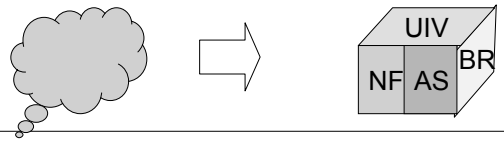
- **Heuristics for decomposing requirements into smaller user stories**
- **A set of guidelines for identifying different kinds of functionality in stories**
- **A set of guidelines for combining story fragments into cohesive stories**



clearStream

Four Common Storyotypes

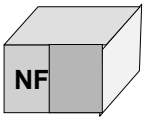
- **New Basic Functionality**
  - Automates some main scenario with minimal user interface
- **Alternate Scenario**
  - Automates alternate scenario (variation of functionality)
- **Business Rule**
  - Defines some rule and provides handling for when rule is broken
- **User Interface Variation**
  -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



clearStream

New Functionality Stories

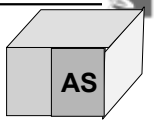
- **Generate a very simple invoice consisting of a single subscription charge for a single customer. (Bootstrap Story)**
  - Simple UI (Enter Customer # and press submit)
  - No data validation or rules enforcement
  - Verify by querying database or printing it out
- **View a customer's (generated) invoice.**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



Some value is provided as soon as these are "Done"

clearStream

Alternate Scenario Stories

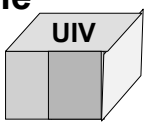


- **Generate an invoice that includes usage-based charges.**
  - The usage data is read in from a flat file and the usage is charged at a rate of $1 per unit of usage.
- **The usage rate can be set via a user interface.**
  - Generate the invoice and view it to verify the rate is being applied correctly.
- **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.

clearStream

UI Variation Stories

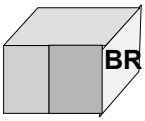
- **Generate (or view) the Invoice for a single customer by clicking on a hyperlink**
- **Generate (or view) the Invoice for a single customer by clicking on a custom icon**
- **Select customers using:**
  - Simple dual list box with add/remove buttons.
  - Drag & Drop into a "Selected Customers" listbox.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



clearStream

Business Rule Stories

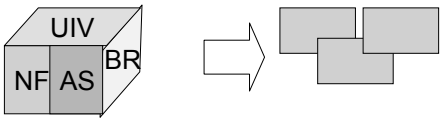
- **An invoice cannot be sent to a customer until it has been finalized.**
- **An invoice that has been finalized cannot be regenerated or modified in any way.**
- **An invoice cannot be generated for a customer until the sales manager has approved them.**
  - This also requires a simple UI to approve 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.



clearStream

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



clearStream

Conclusions

- **User Stories are a way to enumerate requirements**
- **Right size for planning incr. development**
  - of functionally & documentation
- **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 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**
  - Paper by Jennitta Andrea presented at XP2001.
- **Storyotypes - Using Stereotypes to Split Bloated XP Stories**
  - Paper by Gerard Meszaros presented at XP/Agile Universe 2004
  - www.clrstream.com/downloads
- **Structuring Use Cases with Goals**
  - http://alistair.cockburn.us/crystal/articles/sucwg/structuringucswithgoals.htm
- **Use Cases 10 Years Later**
  - http://alistair.cockburn.us/crystal/articles/uctyl/usecasestenyearslater.htm
- **User Stories Applied**
  - Book by Mike Cohn published by Addison Wesley 2004

Time for Questions?

Introduction to User Stories

Gerard Meszaros
ClearStream Consulting
gerard@clrstream.com

clearStream

Additional Material

clearStream

Agile Requirements Techniques

This technique…	reduces wasted effort related to ...
Reduce number	• Telling someone what they don't need to know. • Saying the same thing multiple times.
Reduce detail	• Telling someone what they already know.
Reduce formality	• Spending more time than is necessary to get the message across.
Reduce Handoffs	• Loss of tacit knowledge caused by "throwing over the wall"
Start later	• Getting caught in 'change churn', and 'analysis paralysis'
Retire earlier	• Maintaining artefacts that are no longer needed.
Deliver incrementally	• Getting feedback too late to accommodate change

Reproduced courtesy of Jennitta Andrea and Gerard Meszaros

Advanced Story Techniques

- **Bootstrap Story**
  - A special case
  - Hardest one to plan, estimate
  - Chapters are a way to break a large bootstrap story into smaller chunks
- **Planning multi-release projects**
  - Just Enough look-ahead to do budgeting and feasibility
  - Use “SuperStories” as placeholders for more detailed (and too numerous User Stories)

Combining stories

Combine several stories into one when:

- **Too small to bother managing separately**
  - E.g. MicroFeatures, or small bugs
- **Cannot be tested separately**
  - E.g. Cannot verify x without building y
- **Do not provide value individually**
  - E.g. Data entry provides no value without logic that uses it

More on Story-O-Types

New Functionality

- **Goal:**
  - Automate a task or process in a particular situation
- **Impact:**
  - 1 or more new Transactions or Use Cases
  - Define Main Course per Transactions or Use Case
  - Often requires Business Logic & Presentation Layer logic
- **Examples:**
  - Basic Change Order - Create and View for Simple Product
  - Basic Change Order - Update for Simple Product
- **Notes:**
  - May include the UI to access the functionality, but only one kind of UI (typically, the simplest of several

Variation on Functionality

- **Goal:**
  - Increase # of situations supported by task or process automation
- **Impact:**
  - 1 or more existing Transaction or Use Cases
    - » **May add an alternate Course per Use Case to handle the variation**
    - » **May require changes to the UI to support new functionality**
- **Examples:**
  - Various kinds of payment
  - Change Order - Extra steps for certain kinds of products
- **Notes:**
  - May include the UI to access the extended functionality, but only the same kind of UI as the existing functionality

Business Rule

- **Goal:**
  - Reduce the need for users to enforce the rules of the business
- **Impact:**
  - Affects 1 or more Transactions or Use Cases that change affected Business Object(s)
    - » **may add new Use Case preconditions, verification steps (existing courses), and alternate course for verification failure**
  - May affect Presentation Layer (for Input Validation AKA System Edit Checks)
- **Examples:**
  - Feature-Feature dependency or incompatabilty
  - Prevent unsupported Scenarios
  - Enforce assumptions

User Interface Variation

- **Goal:**
  - Make it easier for users to do an existing task
- **Impact:**
  - Will change Presentation Layer logic
  - Should not change Transaction or Use Case logic
- **Examples:**
  - Given an existing story to:
    - » **Add Item to Invoice by selecting from list**
  - The following could each be a separate UI story:
    - » **Add Item to Invoice via Drag&Drop**
    - » **Create Invoice via HTML (Web Browser)**
    - » **Create Invoice via IVR (Interactive Voice Response)**
    - » **Create Invoice via WAP (Wireless Access Protocol)**

How Can <u>You</u> Apply Them?

- **This "Starter Set" of Storyotypes Came From IS Domain; May Apply to Your Domain, Too.**
- **Look for Common Types (Patterns) of Functionality and Make up Your Own Storyotypes. E.g. Do something to :**
  - One Customer,
  - All Customers,
  - Selected Customers

Use Cases vs User Stories

Sample Story

- **Any Product in the Product Catalog can be ordered via the e-store. Some of the more expensive Products can be purchased using several payments. These payments can either be billed later or set up via preauthorized bank or credit card withdrawals.**
- **When an Admin user defines a product, they can enable the recurring payment feature if the price is over $50.**

clearStream

Stories vs Use Cases