Chapter 1: An Overview

Composed of three aspects:

1. Written description of the story used for planning and as a reminder
2. Conversations about the story that serve to flesh out the details of the story
3. Tests that convey and document details and that can be used to determine when a story is complete

Card, Conversation, Confirmation
Airline Registration System

FEATURE: User wishes to book a flight

USER STORIES:

User needs to book a one way flight

User needs to specify the number of passengers for this booking

User needs to indicate departure airport
User needs to indicate date and time of departure

User needs to indicate destination airport
User needs to know date and time of arrival

User needs to book a roundtrip flight

User needs to indicate date and time of return departure
User needs to know date and time of return arrival
Hypothetical: *BigMoneyJobs* job posting & search website

**Story card 1.1**

An initial user story written on a note card

Sample stories:

- A user can post her resume to the website.
- A user can search for jobs.
- A user can limit who can see her resume.

User stories represent functionality valued by users!

Not good examples:

- The software will be written in C++.
- The program will connect to the database through a connection pool.
The Story details…

Start with a blank webpage… identify the tasks needed to search for a job.

List the unanswered questions about the search:

• What values can users search on?
• Does the user have to be a member of the site?
• Can search parameters be saved?
• What information is displayed for matching jobs?

These details can be expressed as additional stories

Better to have more stories than to have stories that are too large
Two large “epic” stories

1. A user can search for a job
2. A company can post job openings

Good to have stories requiring functionality that can be designed, coded and tested between a half day and two weeks by one or a pair of programmers.

Splitting “epic” stores (example – splitting 1. above):

1.1 A user can search for jobs by attributes like location, salary range, job title, company name, and date the job was posted
1.2 A user can view information about each job that is matched by a search
1.3 A user can view detailed information about a company that has posted a job
Discussing the details of each story

• The **conversation** is “essential”

• **Product Owner** (customer/user) discusses with developers
  
  The intent is not to develop a formal “contract”!

• **Acceptance criteria**
  
  Agreement on the story is documented by tests that demonstrate that the story has been implemented correctly
Each Story… “How long does it have to be?”

- Must define the expectations of the users
- Using paper note cards… list the expectations on the back
  
1.3  A user can view detailed information about a company that has posted a job

Acceptance tests (reminders about how to test the story)
  
Try it with an empty job description
Try it with a really long job description
Try it with a missing salary
Try it with a six-digit salary

(Reminders about how to test are written on the back of the story card)

Note:

Initially, meant to be short and incomplete
Tests can be added and removed later
The Customer Team

On and **ideal** project:

One person with unlimited understanding or knowledge, prioritizes work for the developers, answers their questions

That person would writes all the stories and uses the software when it is finished

Realistically:

Whomever understand what is required to ensure that the software will meet the needs of its intended users
What is the process like?

Product Owner

• Involved throughout the duration of the project
• Expected to be actively involved in writing and approving user stories
• Included are as many user types as possible

For a travel reservation website, include are:
– frequent flyers
– vacation planners
– etc.
Why the **Product Owner** must write the stories…

**Two reasons:**

1. Each story must be written in the language of the business, not in technical jargon… and can be prioritized the stories for inclusion into iterations and releases

2. The **Product Owner** serves as the primary product visionary…

   They are best at describing the expected behavior of the product
Iteration (Sprint) Planning

Product Owner and developers… collaborate

- Iteration length is decided upon… and used for the duration of the project
- At the end of each iteration, developers deliver go-live, fully usable code for some subset of the application

Product Owner:

- Is involved at the start of each Sprint … and at the end of each Sprint
- Specifies acceptance criteria
- Is responsible for ensuring that the project is constantly moving toward delivery of the desired product
more on the process

**Velocity**

Once the iteration length is selected developers (Scrum Team) estimate how much work will be done per Iteration (Sprint)

The initial estimate!

Used to make a *rough* sketch of the work in each Iteration and how many Iterations will be needed

**Release plan**

Stories are sorted into piles, each representing an iteration

The sum of the estimates for each story add-up to the estimated velocity

Highest priority stories go “first”

Prior to the start of each iteration, the Product Owner can make “mid-course” corrections
Planning Releases and Iterations

**Product Owner** prioritizes the *stories* based on:

- Desirability of the feature to a broad base of users
- Desirability of the feature to a small number of important users
- Cohesiveness of the story in relation to the other stories in the release

**Product Owner** considers priorities of developers

- Technical risk of certain stories
- Complementary nature of other stories

**Product Owner** prioritizes the stories to maximize the value to be delivered
Stories and Story Points

**Story points** are a unit of measure for expressing an estimate of the overall effort that will be required to fully implement a **Product Backlog** item or any other piece of work.

When we estimate with **story points**, we assign a point value to each **story**.

Story points are an arbitrary measure used by Scrum teams ... used to measure the effort required to implement a story.

In simple terms its a number that tells the team how hard the story is.

Hard could be related to complexity, Unknowns and effort.
Iteration assignments: stories and their “costs”

<table>
<thead>
<tr>
<th>Story</th>
<th>Story Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Story A</td>
<td>3</td>
</tr>
<tr>
<td>Story B</td>
<td>5</td>
</tr>
<tr>
<td>Story C</td>
<td>5</td>
</tr>
<tr>
<td>Story D</td>
<td>3</td>
</tr>
<tr>
<td>Story E</td>
<td>1</td>
</tr>
<tr>
<td>Story F</td>
<td>8</td>
</tr>
<tr>
<td>Story G</td>
<td>5</td>
</tr>
<tr>
<td>Story H</td>
<td>5</td>
</tr>
<tr>
<td>Story I</td>
<td>5</td>
</tr>
<tr>
<td>Story J</td>
<td>2</td>
</tr>
</tbody>
</table>

**Planned Iterations**

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Stories</th>
<th>Story Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration 1</td>
<td>A, B, C</td>
<td>13</td>
</tr>
<tr>
<td>Iteration 2</td>
<td>D, E, F</td>
<td>12</td>
</tr>
<tr>
<td>Iteration 3</td>
<td>G, H, J</td>
<td>12</td>
</tr>
<tr>
<td>Iteration 4</td>
<td>I</td>
<td>5</td>
</tr>
</tbody>
</table>

**Story I** (5 points) split into

**Story Y** (3 points) and **Story Z** (2 points)

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Stories</th>
<th>Story Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration 1</td>
<td>A, B, C</td>
<td>13</td>
</tr>
<tr>
<td>Iteration 2</td>
<td>D, E, F</td>
<td>12</td>
</tr>
<tr>
<td>Iteration 3</td>
<td>G, H, Y</td>
<td>12</td>
</tr>
<tr>
<td>Iteration 4</td>
<td>J, Z</td>
<td>5</td>
</tr>
</tbody>
</table>
What are Acceptance Tests?

Written early in the iteration… early communication of customer team’s assumptions & expectations to developers

**Sample story:** “A user can pay for the items in her shopping cart with a credit card.”

**Simple tests** (written on back of story card)

1. Test with Visa, MasterCard and American Express (pass)
2. Test with Diner’s Club (fail)
3. Test with a Visa debit card (pass)
4. Test with gook, bad and missing card ID numbers (back of card)
5. Test with expired cards
6. Test with different purchase amounts (include one over the card limit)
Why change from a Requirement’s Document or Use Cases?

**Users Stories:**

– Emphasize verbal rather than written communication
– Are comprehensible by the developers
– Are the right size for planning
– Work fit for iterative development
– Free of technical jargon (comprehensible to both developers and the *customer team*);
– Encourage deferring detail until you have the best understanding about what you really need

“Each user story (*clearly*) represents a discrete piece of functionality that clearly represents what a user would be able to do…”