An Introduction to User Stories for Software Requirements

Mike Cohn
September 21, 2004

Today’s agenda

- What stories are
- What stories are not
- Why user stories
- User role modeling
- INVEST in good stories
Ron Jeffries’ Three Cs

- **Card**
  - Stories are traditionally written on note cards.
  - Cards may be annotated with estimates, notes, etc.

- **Conversation**
  - Details behind the story come out during conversation with customer

- **Confirmation**
  - Acceptance tests confirm the story was coded correctly

Samples – Travel Reservation System

- A user can make a hotel reservation.
- A user can cancel a reservation.
- Users can see photos of the hotels.
- Users can restrict searches so they only see hotels with available rooms.
Where are the details?

- A user can make a hotel reservation.
  - Does she have to enter a credit card?
    - If so, what cards are accepted?
    - Is the charge applied immediately?
  - How can the user search for the hotel?
    - Can she search by city?
    - By quality rating?
    - By price range?
    - By type of room?
  - What information is shown for each room?
  - Can users make special requests, such as for a crib?

Details added in smaller “sub-stories”

- A user can make a hotel reservation.
- A user can search for a hotel. Search fields include city, price range and availability.
- A user can view detailed information about a hotel.
- A room can be reserved with a credit card.
Details added as tests

- Tests are written on the back of a story card
  - Can be used to express additional details and expectations

<table>
<thead>
<tr>
<th>A user can make a hotel reservation.</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Try it with a valid Visa then a valid MasterCard.</td>
</tr>
<tr>
<td>• Enter card numbers that are missing a digit, have an extra digit and have two transposed digits.</td>
</tr>
<tr>
<td>• Try it with a card with a valid number but that has been cancelled.</td>
</tr>
<tr>
<td>• Try it with a card expiration date in the past.</td>
</tr>
</tbody>
</table>

Today’s agenda

- What stories are
- What stories are not
- Why user stories
- User role modeling
- INVEST in good stories
Stories are not IEEE 830 SRS

IEEE 830 Software Requirements Spec (SRS):
- Time-consuming to write and read
- Tedious to read (so readers skip and skim)
- Assumes everything is knowable in advance
- Valuable feedback loops are ignored

- Treats learning as a “change of scope”?

What are we building?

**IEEE Specs**

6. The product shall have a gas engine.
7. The product shall have four wheels.
   1. The product shall have a rubber tire mounted to each wheel.
8. The product shall have a steering wheel.
9. The product shall have a steel body.

Source: Adapted from *The Inmates are Running the Asylum* by Alan Cooper (1999).
What if we had stories instead?

The product makes it easy and fast for the user to mow her lawn.

The user is comfortable while using the product.

The product
Stories are not use cases

Title: Accept reservation for a room.
Primary Actor: Purchaser
Main Success Scenario
- Purchaser submits credit card number, date, and authentication information.
- System validates credit card.
- System charges credit card full amount for all nights of stay.
- Purchaser is given a unique confirmation number.

Extensions
2a Purchaser is given a unique confirmation number.
   2a1 System notifies the user to use a different card.
3a The card is expired.
   3a1 System notifies the user to use a different card.

Differences between use cases and stories

Scope
- Use case is almost always larger.
- Story is closer to a scenario of a use case.

Completeness
- Use cases are complete; stories aren’t.
More differences from use cases

Longevity

- Use cases are permanent artifacts.
- Story cards are torn up.

Purpose

- Use cases document agreement between customers and developers.
- User stories are placeholders for future conversations and are written to facilitate planning.

Today’s agenda

- What stories are
- What stories are not
- Why user stories
- User role modeling
- INVEST in good stories
So, why user stories?

- Shift focus from writing to talking

  If requirements are written down, then
  The user will get what she wants
  At best, she’ll get what was written

- “You built what I asked for, but it’s not what I need.”

Words are imprecise

- Entrée comes with soup or salad and bread.
- (Soup or Salad) and Bread
- (Soup) or (Salad and Bread)
A panda walks into a restaurant…

Eats, shoots, and leaves

Eats shoots and leaves

Actual examples

The user can enter a name. It can be 127 characters.

- Must the user enter a name?
- Can it be other than 127 chars?

The system should prominently display a warning message whenever the user enters invalid data.

- What does should mean?
- What does prominently display mean?
- Is invalid data defined elsewhere?
Another real example

“I handed in a script last year and the studio didn’t change one word.”

“The word they didn’t change was on page 87.”

~Steve Martin

Additional reasons

- Stories are comprehensible
  - Developers and customers understand them
  - People are better able to remember events if they are organized into stories†
- Stories are the right size for planning
- Support and encourage iterative development
  - Can easily start with epics and disaggregate closer to development time

†Bower, Black, and Turner. 1979. Scripts in Memory for Text.
Yet more reasons

- Stories support opportunistic development
  - We design solutions by moving opportunistically between top-down and bottom-up approaches†

- Stories support participatory design
  - Participatory design
    - The users of the system become part of the team designing the behavior of the system
  - Empirical design
    - Designers of the new system make decisions by studying prospective users in typical situations


Today’s agenda

- What stories are
- What stories are not
- Why user stories
- User role modeling
- INVEST in good stories
“The User”

- Many projects mistakenly assume there’s only one user:
  - “The user”
- Write all stories from one user’s perspective
- Assume all users have the same goals
- Leads to missing stories

Travel Site—Who’s the user?

- **Mary**
  - Frequent flier who never knows where she’ll be
- **Jim**
  - Frequent flier who flies every week but always to the same place
- **Howard**
  - Mary’s assistant; books her reservations
- **Laura**
  - Wants to schedule her family’s annual vacation
- **Dominic**
  - Hotel chain Vice President; wants to monitor reservations
User roles

- Broaden the scope from looking at one user
- Allows users to vary by
  - What they use the software for
  - How they use the software
  - Background
  - Familiarity with the software / computers
- Used extensively in usage-centered design
- Definition
  - A user role is a collection of defining attributes that characterize a population of users and their intended interactions with the system.

Common attributes

- **Mary**: Frequent flier who never knows where she’ll be
- **Jim**: Frequent flier who always flies to the same place
- **Laura**: Wants to schedule her family’s annual vacation
- **Dominic**: Hotel chain Vice President; wants to monitor reservations
- **Howard**: Mary’s assistant; books her reservations
- **Insider**: Repeat Traveler
- **Scheduler**: Infrequent Vacation Planner

Source: Software for Use by Constantine and Lockwood (1999).
User role modeling

**Identify attributes that distinguish one user role from another**

- How often the software will be used
- Level of domain expertise
- General level of computer proficiency
- Level of proficiency with this software
- General goals for using the software

Document the user role

**User Role: Infrequent Vacation Planner**

Not particularly computer-savvy but quite adept at using the web. Will use the software infrequently but intensely (perhaps 5 hours to research and plan a trip). Values richness of experience (lots of content) over speed. But, software must be easy to learn and also easily recalled months later.
Advantages of using roles

- Users become tangible
- Start thinking of software as solving needs of real people.

- Avoid saying “the user”
- Instead we talk about “a frequent flier” or “a repeat traveler”

- Incorporate roles into stories
- “As a <role>, I want <story> so that <benefit>.

Ask users the right questions

- “Would you like it in a browser?”
- “Of course, now that you mention it!”

- A problem
  - The question is closed
    - {Yes | No}
We can do better

“What would you think of having this app in a browser rather than as a native Windows application even if it means reduced performance, a poorer overall user experience, and less interactivity?”

- It’s open
  - Full range of answers
- But it has too much context

The best way to ask

“What would you be willing to give up in order to have it in a browser?”

- We want to ask questions that are
  - Open-ended
  - Context-free
Today’s agenda

- What stories are
- What stories are not
- Why user stories
- User role modeling
- INVEST in good stories

What makes a good story?

INVEST

- Independent
- Negotiable
- Valuable
- Estimatable
- Small
- Testable

Thanks to Bill Wake for the acronym. See www.xp123.com.
Independent

- Avoid introducing dependencies
  - Leads to difficulty prioritizing and planning

- A company can pay for a job posting with a Visa card.
- A company can pay for a job posting with an AmEx card.
- A company can pay for a job posting with a MasterCard.

- The first of these stories will take 3 days to develop
  - It doesn’t matter which is first
  - The others will take 1 day

Making stories independent

- Combine the stories
  - A customer can pay with a credit card.

- Split across a different dimension
  - A customer can pay with one type of credit card.
  - A customer can pay with two other types of credit cards.

- Write two estimates and move on
  - 3 days if first; 1 otherwise
Negotiable

- Stories are not
  - Written contracts
  - Requirements the software must fulfill
- Do not need to include all details
- Too many details give the impressions of
  - false precision or completeness
  - that there’s no need to talk further
- Need some flexibility so that we can adjust how much of the story gets implemented
  - If the card is contract then it needs to be estimated like a contract

Valuable

- Stories must be valuable to either:
  - Users
    - A user can search for a job by title and salary range.
  - Purchasers
    - Throughout the project, the development team will produce documentation suitable for an ISO 9001 audit.
    - The development team will produce the software in accordance with CMM level 3.
    - All configuration information is read from a central location.
Stories valued by developers

- Should be rewritten to show the benefit
  - All connections to the database are through a connection pool.
  - Up to 50 users should be able to use the application with a five-user database license.
  - All error handling and logging is done through a set of common classes.
  - All errors are presented to the user and logged in a consistent manner.

Estimatable

- Because stories are used in planning
- A story may not be estimatable if:
  - Developers lack domain knowledge
    - New users are given a diabetic screening.
  - Developers lack technical knowledge
    - A user can select to see all text on the site in a larger font.
  - The story is too big
    - A user can find a job.
Small

- Large stories (epics) are
  - hard to estimate
  - hard to plan
    - They don’t fit well into single iterations
- Compound story
  - An epic that comprises multiple shorter stories
- Complex story
  - A story that is inherently large and cannot easily be disaggregated into constituent stories

Compound stories

- Often hide a great number of assumptions

A user can post her resume.

- A resume includes separate sections for education, prior jobs, salary history, publications, etc.
- Users can mark resumes as inactive
- Users can have multiple resumes
- Users can edit resumes
- Users can delete resumes
Splitting a compound story

Split along operational boundaries (CRUD)

- A user can create resumes, which include education, prior jobs, salary history, publications, presentations, community service, and an objective.
- A user can edit a resume.
- A user can delete a resume.
- A user can have multiple resumes.
- A user can activate and inactivate resumes.

Splitting a compound story, cont.

Split along data boundaries

- A user can add and edit educational information on a resume.
- A user can add and edit prior jobs on a resume.
- A user can add and edit salary history on a resume.
- A user can delete a resume.
- A user can have multiple resumes.
- A user can activate and inactivate resumes.
Testable

- Tests demonstrate that a story meets the customer’s expectations
- Strive for 90+% automation

A user must find the software easy to use.

A novice user is able to complete common workflows without training.

A user must never have to wait long for a screen to appear.

New screens appear within 2 seconds in 95% of all cases.

Additional guidelines for good stories

- Start with goals
- Slice the cake
- Write closed stories
- Put constraints on cards
- Size the story to the horizon
- Keep the UI out as long as possible
- Some things aren’t stories
- Include user roles in the stories
- Write for one user
- Write in active voice

Don’t forget the purpose
For more on user stories

- www.userstories.com
- groups.yahoo.com/group/userstories

Where to go next?

- **Agile Planning**
  - groups.yahoo.com/agileplanning
  - www.mountaingoatsoftware.com/agileplanning

- **Agile in General**
  - www.agilealliance.com

- **Scrum**
  - www.mountaingoatsoftware.com/scrum
My contact information

- Email: mike@userstories.com, mike.cohn@computer.org