Introduction to User Stories, Part 2

CSCI 5828: Foundations of Software Engineering
Lecture 06 — 09/11/2014

Goals

- Continue our introduction to the topic of user stories
    - how do we write them?
    - how do we generate them?
        - user role modeling

Writing Styles

- Writing user stories is like any form of writing
  - there are good ways to do it and there are bad ways to do it
- If you adopt the wrong writing style, you can create horrible user stories
  - If you end up not liking user stories, it might be because of the style you used and not the technique itself!
- This was true of a technique called use cases that are sometimes used in object-oriented analysis and design
  - let's see an example

Bad Use Case

- Access User Profile

    1. If user is on any page other than home page, user clicks on Home link
    2. On home page, user clicks on Account link
    3. System responds by fading in a dialog asking for username and password
    4. User clicks on Name field; User enters name; User hits the tab key
    5. User types password; User hits the enter key
    6. If the name/password combo is correct, system displays profile page
    7. If the name/password combo is wrong, system reveals a hidden HTML element that tells them to try again

Better Use Case

- Access User Profile
  - Preconditions
    - User is viewing home page and is currently unauthenticated
  - Steps

1. User makes request to view their profile
2. User provides credentials
    2.1. Credentials are incorrect
        2.1.1. System informs user; use case ends
3. System displays user profile

Discussion (I)

- The “bad” use case would be horrible to work with on a daily basis
  - It’s brittle
  - It’s unstructured
  - It mixes conditional logic into steps in an awkward way
    - making the success path unclear (imagine if it had more conditionals!)
- The “better” use case provides a description of what functionality is needed without specifying how that functionality should be implemented
  - It clearly specifies preconditions and makes success path clear
  - It is resilient to changes in the actual prototype

Discussion (II)

- When people tell me they “hate” use cases, which type of use case do you think they were working with?

- This underscores the need to understand the writing style that goes into the documents that you create in a software development process

- Chapter 2 of our textbook lays out the writing style for user stories

INVEST

- INVEST is an acronym for the user story writing style adopted by our textbook
- User stories should be
  - Independent
  - Negotiable
  - Valuable to users and customers
  - Estimatable
  - Small
  - Testable

Independent

- User stories should be self contained pieces of functionality
  - They should not be dependent on other user stories
- The main reason is that dependencies introduce problems with planning
  - A high priority story may depend on a low priority story
    - We’re supposed to work on high priority stories first!
    - May require customer education
- Dependencies can also impact our ability to make estimates
  - Will the work on the first story mean that the second story can be completed more quickly?
- Find ways to combine the stories so the result is self contained

Negotiable

- The textual details of a story should not become “set in stone” contracts
  - The story is a reminder to have a conversation about a particular feature
    - You want the feature to be described briefly
      - You can add notes about issues that need to be resolved via conversation
    - Details that become resolved should be turned into tests

Example

- Compare
    - A company can pay for a job posting with a credit card
        - Note: Accept Visa, MasterCard. Consider other cards
- With
    - A company can pay for a job posting with a credit card. Note: Accept Visa, MasterCard, and American Express. Consider Discover. On purchases over $100, ask for CCV on back of card. The system can tell what type of card it is from first two digits of card number. The system can store a card number for future use. Collect the expiration date for the card.
- Writing style matters! All of the details in the second example can change. They clutter the card and obscure the intent. They hide what should be other user stories in among the details.

Valuable

- Harks back to our discussion in Lecture 5 about customer and user
  - There may be a distinction between these roles; if so, you will write user stories that address both roles
    - User: A user can archive closed bug reports.
    - CTO: Updates to the bug tracking system will automatically be detected and applied by all machines used by employees
- The main point is that each user story should describe functionality that the end users and the customer find valuable
  - Avoid stories that are only interesting to engineers on the development team
    - “All connections to the database are through a connection pool”

Estimatable (I)

- Write stores such that they can be given a solid estimate
  - (And, yes, at the beginning all estimates are guesses while the team gets used to the “points system” and tries to determine their “velocity”)
- The main issues that prevent a team from making good estimates are
  - The story requires domain knowledge that the developers do not have
    - Before determining exposure age, an accumulation rate should be calculated on a calibration set
  - The story requires technical skills not known by the developers
    - The amount of new data generated by the system is expected to be terabytes per day; The team has no expertise in “big data”
  - The story is too big
    - The system will process all of the companies financial transactions

Estimatable (II)

- How to address?
  - Lack of Domain Knowledge
    - (Lots of) Conversations with the customer and end users
  - Lack of Technical Knowledge
    - Technology Spikes (Goal is not to develop software; Goal is to learn)
      - Handled by treating the spike as a story; allows integration into the planning process
- Story is too big
  - Conversations with customer/end users to identify smaller stories
    - Story needs to be decomposed until the parts can be assigned solid estimates; problem: don’t go too far in the other direction!

Small

- The amount of functionality in a user story needs to be “just right”
  - Too small and it’s not worth the time it takes to create the story and insert it into the planning process
- Epic stories are too big
  - They often describe the purpose of the entire system
    - They still can be useful if you need to describe a major class of functionality that does not need to be implemented first
- Epics typically fall into two categories
  - Compound stories and Complex stories
    - For the former, you decompose until you can make estimates
    - For the latter, you identify domain spikes that let you learn what you need to make progress

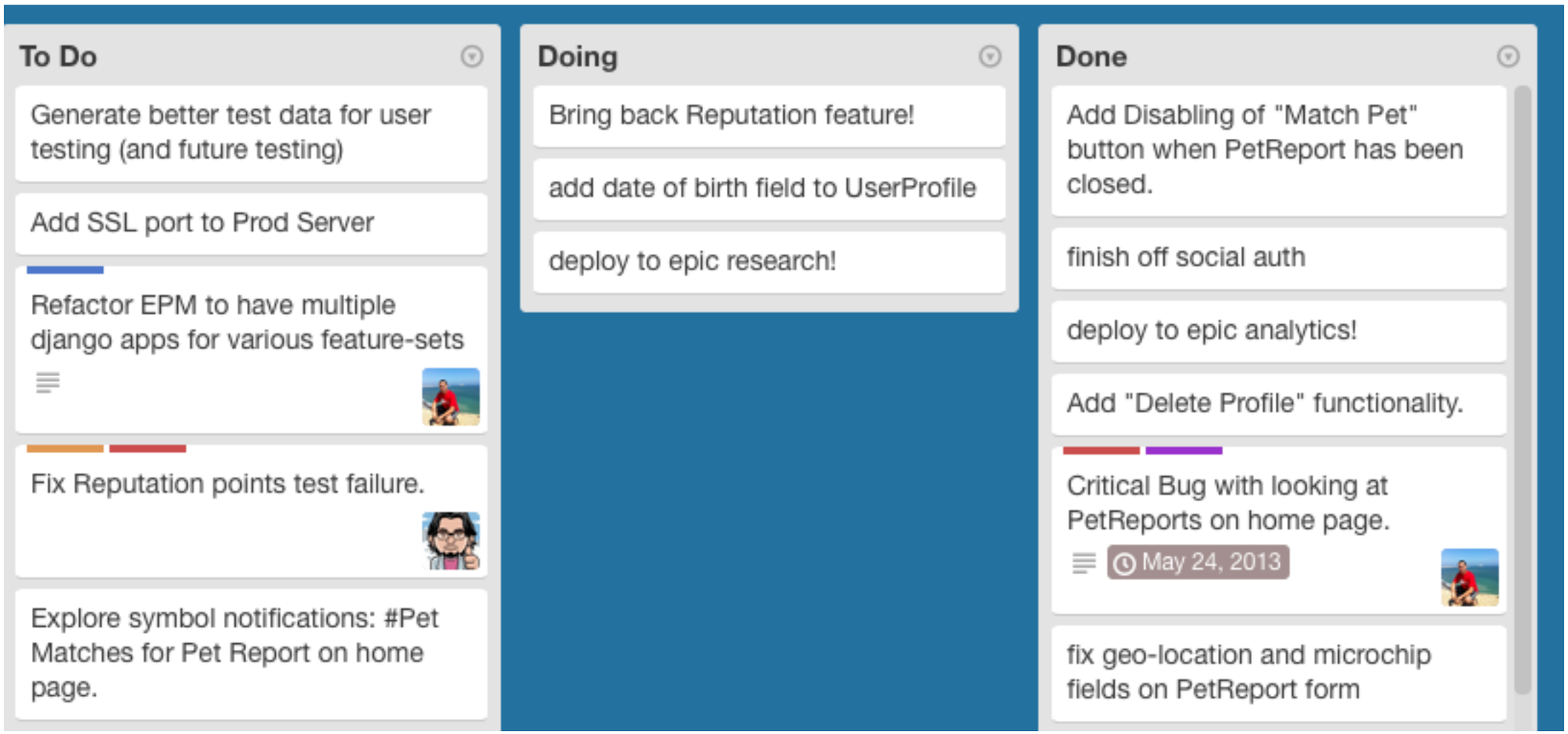
Testable

- We need to generate stories that can be tested
  - The details of a story are documented by tests
  - A story is considered finished when all of its tests finally pass
- A story that is not testable, would wreck our ability to make progress

- These problems sometimes arise with non-functional requirements
  - Avoid ambiguous terms when describing non-functional requirements
    - The system will display all pages quickly vs. The system will display all pages in under a second
  - Avoid non-functional stories that cannot be automated
    - Novice users can use the system without training

Discussion

- These style guidelines are just that: guidelines
- In truth, you can write anything you want for your user story



Here's a Trello board with a mix of stories and tasks

User Role Modeling

- How do we “discover” stories?
  - We, of course, talk with our users and customers
    - Or domain experts and customer team members if the former are not available
- But, an important technique (taken from human-centered computing) is
  - user role modeling and personas
- User role modeling and personas
  - What roles do users adopt when using our system?
  - What types of users do we have? What do they care about (with respect to our system)?

How do we identify user roles?

- Most of the roles will be obvious
    - For a job finding website: job seeker and job poster will jump right out
        - Also a default role (at least in web contexts) is: administrator
- But, if you spend time thinking about it, a whole range of behaviors can be identified
    - First Time Job Seeker
    - Geographic Searcher
    - Longtime Monitor
    - Recent Layoff

Role Modeling Steps

- Brainstorm an initial set of roles
- Organize the initial set
- Consolidate roles
- Refine roles

- Then after you have developed a candidate set of roles
  - Generate user stories for each role
- This should provide a wealth of stories to get started planning releases and iterations

Brainstorming Roles

- Takes about 15 minutes
    - Gather the team (developers and customer team), users, other stakeholders
    - Put index cards on a table
    - Have everyone generate roles all at once
        - No turn taking
        - No duplicate elimination
        - No discussion
- Place roles on a table or up on a wall

Organize the Set

- Now, you can look at the generated roles and organize them
  - Similar roles should be placed on top of each other physically showing the degree of overlap
  - Subcategories should be identified and indicated visually
- See Figure 3.1 in textbook for an example

Consolidate Roles

- Eliminate duplicate roles
- Take roles with a high degree of overlap and see if you can generalize or condense into a single role
    - In the textbook, College Grad and First Timer were too similar. They eliminated the former since the latter conveyed the role more generally
- Your users and customers can help in this phase by identifying the most important roles
    - Perhaps one role has a lot of potential subcategories; the customer/user can help to eliminate the ones that are not important (at all) or not important for the first/next release

Refine the Roles

- Provide more information about a role to help with story generation
- Performed via role attributes: a fact or bit of information about the role
- Consider
  - the frequency with which the role will use the system
  - the role's level of experience with the application domain
  - the role's general level of proficiency with computers and software
  - the role's goal for using the software
- Place this information on the index cards as notes

Personas

- Once you have these roles, you can flesh it out by creating a persona
  - A persona is a representation of a user role that provides motivation and other details about people who play this role. Can motivate story creation
  - Consider:
    - User Role
      - **Matcher**: The matcher is a digital volunteer either within the affected community or outside of the area working to reunite pets with their owners.
    - Persona
      - **Charlie is a 4th grade student** whose teacher is interested in assisting victims of a recent earthquake in California. The teacher directs Charlie and his classmates to EPM as a volunteer activity. Charlie enjoys finding pet matches for lost dogs—the activity is like a game to him.

Additional Examples

- The examples at the beginning of chapter 3 are good initial examples of creating personas
  - Max has a job but always keeps an eye out for a better one
  - Allan will take any job that lets him move to the beach
  - Scott likes his job but realizes he needs to move on to develop professionally
  - Kindra needs to find a job in the Southwest to stay near her family
  - Delaney works in human resources and needs to process resumes quickly
- Anyone of these could be fleshed out with additional details after being assigned to one of the candidate roles

Extreme Characters

- The book reviews a technique called Extreme Characters
  - basically identifying users who might use the system in extreme ways
    - The power web user who has 1000s of browser tabs open (and expects the system to remember them from session to session)
    - The shared calendar used by two professionals with kids
    - The music application used by an audiophile with 10,000s of albums
- This is not a widely used technique but it can be useful to think about power users in your requirements and design work

Summary

- Good writing, in general, adopts good writing styles
  - Writing user stories well is no different
  - The INVEST acronym identifies characteristics of good, usable user stories
  - Independent, Negotiable, Valuable, Estimatable, Small, & Testable
- User Role Modeling can help you identify user stories you might otherwise have missed
  - Who uses your system and why?
- Personas can flesh out the details of a user role and make it easier to identify stories
- While user stories come and go, user roles and personas might be “up on the wall” for the entire development life cycle

Coming Up Next

- Lecture 7: Concurrency: Threads and Locks, Part One
- Lecture 8: Concurrency: Threads and Locks, Part Two
    - Might also include an “Intro to Clojure” to prepare for Chapter 3 in the concurrency textbook
- Homework 3 and Quiz 2 coming soon!