User Stories

Writing Good Stories

Writing Good Stories

- Good stories should follow the INVEST acronym:

Independent

Negotiable

Valuable

Estimatable

Small

Testable

Writing Good Stories: Independent

- Dependencies between stories should be avoided
  - Dependencies lead to prioritization and planning problems
  - Make estimation more difficult
  - Need to have the flexibility to easily move stories around in the release schedule, if needed

- Solutions:
  - Combine dependent stories into a larger, independent story
  - Find a different way of splitting the stories

Writing Good Stories: Independent

- Dependent stories:

> The customer should be able to pay with Visa.
> The customer should be able to pay with MasterCard.
> The customer should be able to pay with American Express.

  - The first credit card will take 3 days to support, but each other card after that will only take 1 day. Which story should receive the 3 day estimate?

- A better split:

> The customer cap pay with one type of credit card.
> The customer can pay with two additional types of credit cards.

Writing Good Stories: Negotiable

- Stories should be negotiable – they are not written contracts or requirements

- Stories are reminders to have a conversation
    - Do not need to include all relevant details
    - We negotiate the details in a conversation between the customer team and the development team
    - If we do know some details when the story is written, we can annotate the card with those details

Writing Good Stories: Negotiable

- Right amount of information to the developer and customer who will discuss the story:

> A company can pay for a job posting with a credit card.
>
> Note: Accept Visa, MasterCard, AmEx. Consider Discover.

- Too many missing details for developers to view the story as definitive

Writing Good Stories: Negotiable

- Too much detail:

> A company can pay for a job posting with a credit card.
>
> Note: Accept Visa, MasterCard, and AmEx. Consider Discover. On purchases over $100, ask for cardID number from back of card. The system can tell what type of card it is from the first two digits of the card number. The system can store a card number for future use. Collect the expiration month and date of the card.

- Can lead to mistaken belief that the story card reflects all the details and that there's no further need to discuss the story with the customer

Writing Good Stories: Valuable

- User stories should be valuable to the users and customers

- Avoid stories valued only by developers
    - Should be written so that the benefits to the customers / users are apparent
    - This allows customers to intelligently prioritize the stories

Writing Good Stories: Valuable

- Poor examples:

> All connections to the database are through a connection pool.
> All error handling and logging is done through a set of common classes.

- Better:

> Up to fifty users should be able to use the application with a
> 5-user database license.
> All errors are presented to the user and logged in a consistent manner.

Writing Good Stories: Estimatable

- Developers need to be able to estimate the size of a story

- Three reasons why a story might not be estimatable:
  1. Developers lack domain knowledge
  2. Developers lack technical knowledge
  3. The story is too big

Writing Good Stories: Estimatable

1. Developers lack domain knowledge
   - If developers do not understand a story, they should discuss it with the customer who wrote the story
   - Not necessary to understand all details but developers should have a general understanding of the story

Writing Good Stories: Estimatable

2.   Developers lack technical knowledge
   - Send one or more developers on a spike: a brief experiment to learn about an area of the application
   - Developers learn just enough that they can estimate the task
   - The spike itself is given a defined maximum amount of time (a timebox), allowing the developers to estimate the spike
   - Unestimatable stories turn into 2 stories: the spike and then the actual story

Writing Good Stories: Estimatable

3.   The story is too big
    - Example: *A Job Seeker can find a job*
    - Developers will need to disaggregate it into smaller, constituent stories

- Stories that are too large are called epics

- Despite being too large to estimate, epics are useful as placeholders – reminders about big parts of the system that must be discussed
    - This allows us to make a conscious decision to temporarily gloss over large parts of a system
    - Can be assigned a large, pulled-from-thin-air estimate

Writing Good Stories: Small

- Size matters – stories should be small, but not too small

- Stories that are too large or small make planning difficult

Writing Good Stories: Testable

- Stories must be written so as to be testable

- Successfully passing its tests proves a story has been successfully developed

- If the story cannot be tested, how will the developers know when they have finished implementing it?

- Untestable stories commonly appear with nonfunctional requirements:

> A user must find the software easy to use.
> A user must never have to wait long for any screen to appear.

Writing Good Stories: Testable

- Tests should be automated as much as possible. Strive for 99% automation – not 10%

- Things change quickly:
    - Code that worked yesterday may be broken today
    - Automated tests detect this quickly

- Some stories cannot be tested automatically, however:

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

- This can be tested, but not readily automated

User Stories: Summary

- Emphasize verbal rather than written communication
  - Remind us to have a conversation
- Comprehensible by both customers and developers
  - Written in the language of the domain
- The right size for planning
  - Not too large, not too small
- Work for iterative development
  - As long as we follow INVEST, can easily move stories between iterations
- Encourage deferring detail
  - We may not even need a story – don't waste time detailing it up front

User Stories: Some Caveats

- Scaling up
  - How does this scale to very large projects with geographically distributed teams, especially if we are using small physical cards for our stories?

- Vague, informal, incomplete
  - As user stories are intended to start conversations, they are informal, open to interpretation, and short on details; they are definitely not appropriate for a formal agreement or contract

- Lack of non-functional requirements
  - Since performance or non-functional requirements are most often not included in user stories (testability), there is a risk of them being overlooked