CSE334
Software Engineering
User Stories

Dr Islam El-Maddah

User stories examples

As a/an	I want to...	so that...
moderator	create a new game by entering a name and an optional description	I can start inviting estimators
moderator	invite estimators by giving them a url where they can access the game	we can start the game
estimator	join a game by entering my name on the page I received the url for	I can participate
moderator	start a round by entering an item in a single multi-line text field	we can estimate it
estimator	see the item we're estimating	I know what I'm giving an estimate for
estimator	see all items we will try to estimate this session	I have a feel for the sizes of the various items
moderator	see all items we try to estimate this session	I can answer questions about the current story such as "does this include ___"
moderator	select an item to be estimated or re-estimated	the team sees that item and can estimate it

User Stories

- 3 aspects
  - Written description of story used for planning/reminding
  - Conversations about the story that serve to fill out details
  - Tests that convey and document details & can be used to determine when a story is complete
- Represent functionality that will be valued by users

Examples of specifying value to users

- Good:
  - "A user can search for jobs"
  - "A company can post new jobs"

Bad:

"The software will be written in C++"

"The program will connect to the database through a connection pool"

Sizing stories

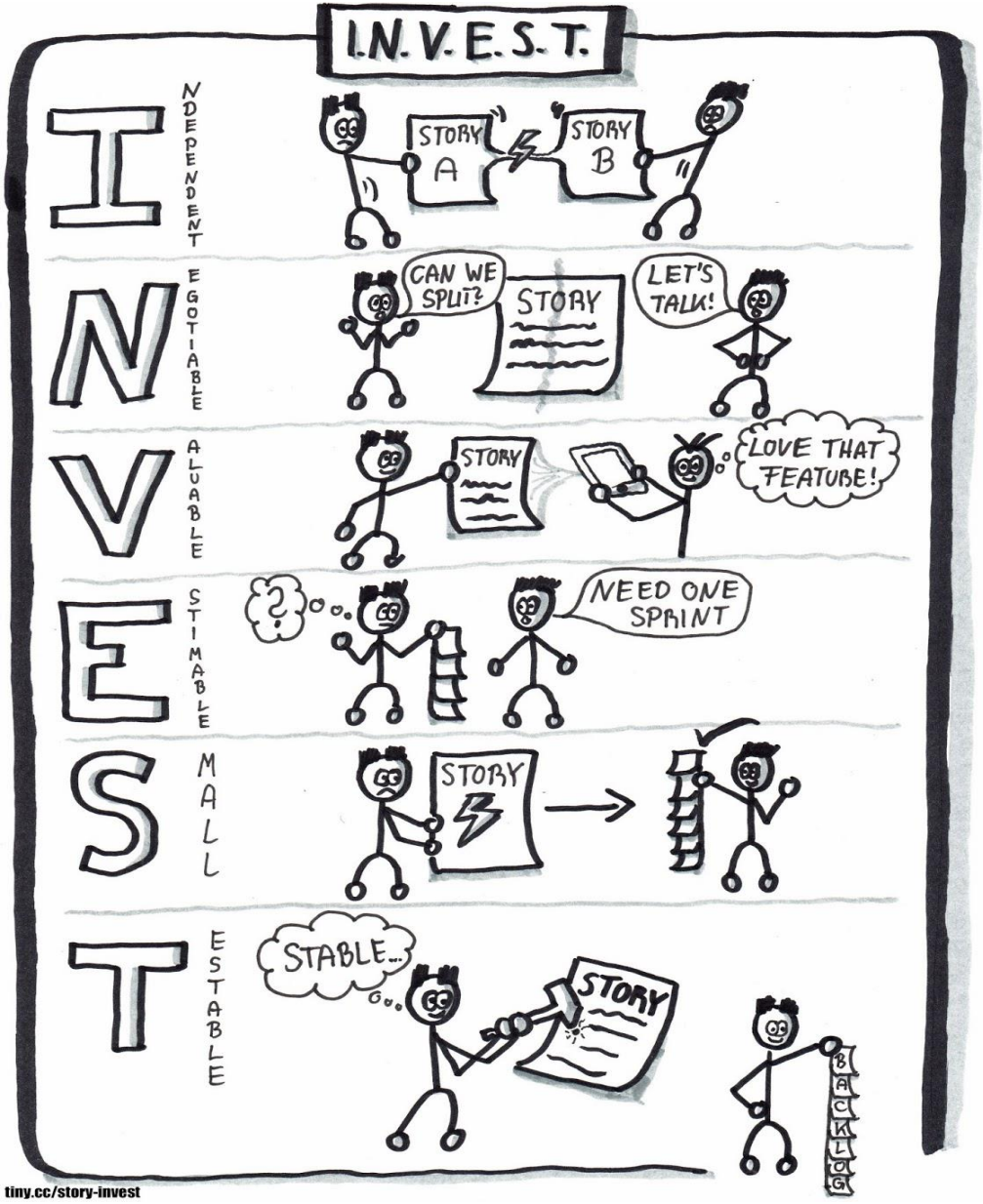
- Too broad = impossible to test/code
- Too narrow = more time spent specifying than implementing

- Aim for test & code cycle of about 4 hours to 2 weeks by one or 2 programmers per story
- Split long stories (“epics”) into smaller pieces
- Rather than specify small details, get those in conversations with customer & annotate story
- Big stories can serve as placeholders for areas of the system that still need to be discussed

Writing Stories

- Customer writes stories
  - Written in language of business to allow prioritization
  - Customer is primary product visionary
- Good stories are
  - Independent
  - Negotiable
  - Valuable to users or customers
  - Estimable
  - Small
  - Testable (INVEST)

- **I** independent
- **N** negotiable
- **V** valuable
- **E** estimable
- **S** small
- **T** testable



Independent

- Stories that depend on other stories are difficult to prioritize and estimate
- Example:
  - A company can pay for a job posting with a Visa card
  - A company can pay for a job posting with a Mastercard
  - A company can pay for a job posting with an American Express card

Negotiable

- Story cards serve as reminders not contracts
- Details need to be fleshed out in conversation
- Story cards should have a phrase or sentence to serve as reminder to have conversation & notes about conversation

Valuable

- Both to people using the software and paying for the software
- Avoid stories valued only by developers (make the benefits to customers/users apparent for these stories)
- Example
  - “All connections to the database are through a connection pool” could be rewritten as “Up to 50 users should be able to use the application with a 5-user database license”

Estimable

- 3 common reasons why story might not be
  - Developers lack domain knowledge
    - Get details from customer
  - Developers lack technical knowledge
    - Perform spike to explore technology
  - Story is too big
    - Split the story into smaller ones

Small

- Easy to use in planning
- Split compound & complex stories
- Combine too small stories

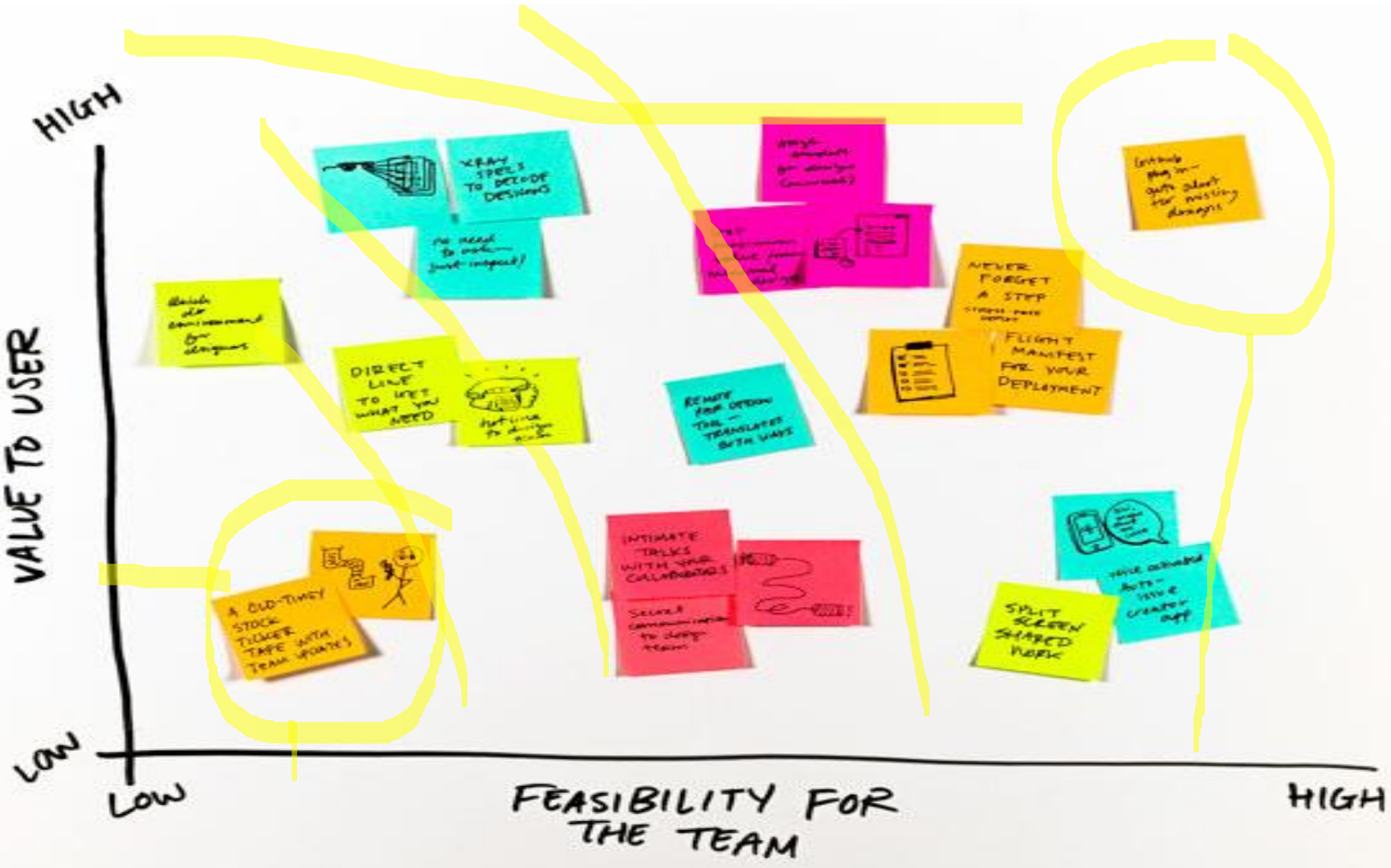
Splitting Stories

- Compound
  - Conversations may reveal multiple stories
  - Split along Create/Update/Delete
  - Split along data boundaries
- Complex
  - Split into investigative and develop the new feature stories (define timebox for investigation)
- Make sure each split-off portion is a good story (INVEST)

Testable

- Can’t tell if story is done without tests
- Aim for most tests to be automatable

Sorting user stories



Story Responsibilities (Developers)

- Help the customers write stories that
  - Are promises to converse rather than detailed specs
  - Have value to the users or the customer
  - Are independent
  - Are testable
  - Are appropriately sized
- Describing the need for technology/infrastructure in terms of value to users or customers
- Have the conversations with the customers

Story Responsibilities (Customers)

- Writing stories that
  - Are promises to converse rather than detailed specs
  - Have value to users or to yourself
  - Are independent
  - Are testable
  - Are appropriately sized
- Have the conversations with the developers

Users

- Can be more than one type of user for system
- Different user types may have different roles and different stories
- Disfavored users (e.g., hackers) may be included as well as favored users
  - Obviously, the value to users piece gets flipped for disfavored users

Role Modeling

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

Roles for Magic DB?

Attributes worth considering when refining roles

- Frequency with which user will use software
- User’s level of expertise with domain
- User’s general level of proficiency with computers and software
- User’s level of proficiency with this software
- User’s general goal for using software

Additional user modeling

- Identify personas
  - Should be described sufficiently so everyone on team feels like they know this “person”
  - Choose personas that truly represent user population
- Extreme characters
  - Make a PDA for
    - Drug dealer
    - The Pope
    - 20-year old woman juggling multiple boyfriends

User Modeling Responsibilities (Developers)

- Participate in process of identifying user roles and personas
- Understanding each user role & how they differ
- Thinking about how different user roles might prefer their software to behave
- Identifying & describing user roles doesn’t go beyond role of tool in the process

User Modeling Responsibilities (Customers)

- Looking broadly across space of possible users & identifying appropriate roles
- Participating in process of identifying roles and personas
- Ensuring software does not focus inappropriately on a subset of users
- Ensuring that each story can be associated with at least one user role or persona
- Thinking about how each user role may prefer their software to behave
- Identifying & describing user roles doesn’t go beyond role of tool in process

Getting stories

- Trawl for stories (rather than elicit)
  - Different size nets
  - Fish grow and die
  - Won’t capture all fish by trawling
  - Skill plays a factor
- Techniques
  - User interviews
  - Questionnaires
  - Observation
  - Story-writing workshops

Story Gathering Responsibilities

- Understand & use multiple techniques
- Knowing how to make use of open-ended and context-free questions
- Customer-specific
  - Write as many user stories as early as possible
  - Understanding options in communicating with users
  - Scheduling & running story-writing workshop
  - Ensuring all user roles represented when trawling

Acceptance Tests

- Use tests to track details
- Write the tests before coding
- Acceptance tests are ideally specified by the customer
- Don’t replace unit tests

Acceptance Test Responsibilities (Developers)

- Automating the execution of acceptance tests
- Thinking about additional acceptance tests when starting development
- Unit testing

Acceptance Test Responsibilities (Customers)

- Writing acceptance tests
- Executing acceptance tests

Guidelines for Good Stories

- Start with goal stories
- Write closed stories (stories that have a definite end point)
  - "A recruiter can review resumes from applicants to one of her ads" instead of "A recruiter can manage the ads she has placed"
- Put constraints on the system on cards & implement automated tests for them

Guidelines II

- Size your story appropriately for the time frame it may be implemented in
- Keep the UI out as long as possible
- Don’t rely solely on stories if somethings are better expressed in other ways
- Include user roles in stories rather than saying “user”
- Write for a single user (“A Job Seeker” not “Job Seekers”
- Use Active Voice

Son of Guidelines

- Don’t number the story cards
- Don’t forget the purpose of using story cards

Using Stories

- Assign points to stories based on difficulty of coding
- For each release, customer prioritizes stories
- Developers determine velocity (# of points per release cycle) for previous cycle and plan to implement the highest priority stories up to that number of points for the release

User Stories are Not

- Requirements documents
- Use Cases
- Scenarios

Why do user stories?

- Emphasize verbal communication
- Comprehensible by everyone
- Right size for planning
- Work for iterative development
- Encourage deferring detail
- Support opportunistic development
- Encourage participatory design
- Build up tacit knowledge

Why not user stories?

- May be difficult to understand relations between stories
- Not suitable for requirements traceability (if required by process)
- May not scale well to large teams

Story Smells

- Stories are too small
- Interdependent stories
- Goldplating
- Too Many details
- Including UI detail too soon
- Thinking too far ahead
- Splitting too many stories
- Customer has trouble prioritizing
- Customer won’t write and prioritize the stories