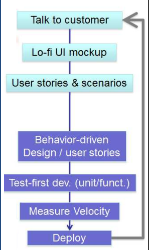
TCSS 360: SOFTWARE DEVELOPMENT AND QUALITY ASSURANCE

User stories, use cases, and requirements analysis

Wes J. Lloyd
Institute of Technology
University of Washington - Tacoma

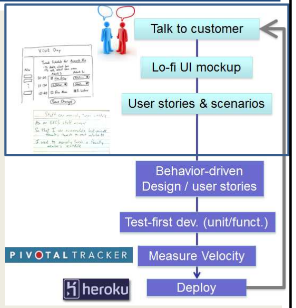


OBJECTIVES

- Behavior drive design
- User stories
- Use cases
- UI sketches, storyboards
- Agile cost estimation

AGILE: BEHAVIOR DRIVEN DESIGN

- Behavior Driven Design (BDD)
- Ask questions about the design of an application before and during development
- Record and continuously refine requirements
- Goal of BDD is validation focused:
  - Ensure we build the right thing



BDD: USER STORIES

- BDD's version of requirements/use cases
- Describe how the application is expected to be used
- Take the place of plan & document's formal requirements and design documents
- Help project stakeholders plan and prioritize development
- User stories must be testable
- Small enough to implement in one agile process iteration (~2 weeks)
- Must have business value

BDD: USER STORIES - 2

- From human computer interaction (HCI) community
- Originally written on index cards
- 1 to 3 sentences
- Capture who, what, why
- Non-technical language
- Can be written by customers (users) or developers
- Cards: easy to rearrange (prioritize), promote brainstorming

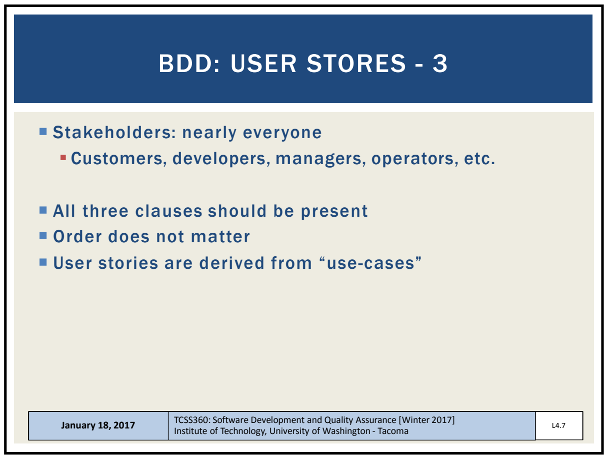
```
Feature name: [descriptive name]
  As a [kind of stakeholder],
  So that [I can achieve some goal],
  I want to [do some task]
```

USER STORIES EXAMPLE

```
Feature name: [descriptive name]
  As a [kind of stakeholder],                who
  So that [I can achieve some goal],         why
  I want to [do some task]                   what
```

```
Feature name: Add a movie to RottenPotatoes
  As a movie_fan,
  So that I can share a movie with other movie_fans,
  I want to add a movie to the RottenPotatoes database
```

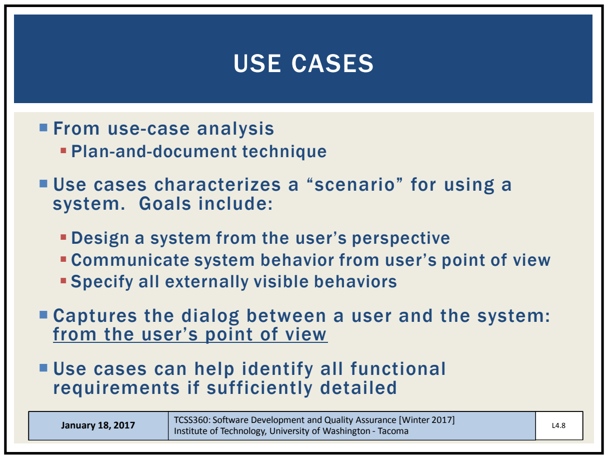
```
Feature: Add movie tickets to shopping cart
  As a patron
  So that I can attend a showing of a movie
  I want to add tickets to my order
```



BDD: USER STORES - 3

- Stakeholders: nearly everyone
  - Customers, developers, managers, operators, etc.
- All three clauses should be present
- Order does not matter
- User stories are derived from "use-cases"

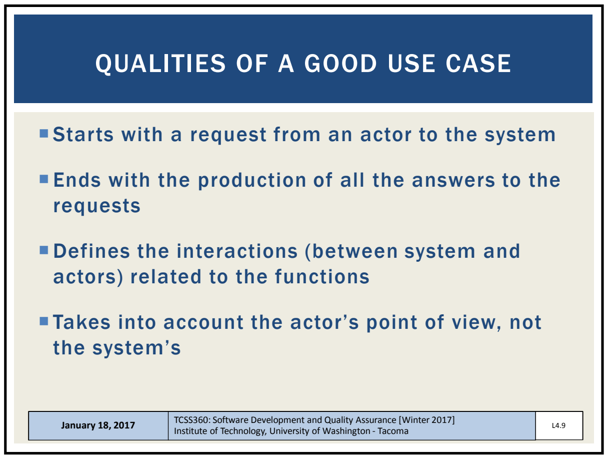
January 18, 2017 | TCSS360: Software Development and Quality Assurance [Winter 2017] Institute of Technology, University of Washington - Tacoma | L4.7



USE CASES

- From use-case analysis
  - Plan-and-document technique
- Use cases characterizes a "scenario" for using a system. Goals include:
  - Design a system from the user's perspective
  - Communicate system behavior from user's point of view
  - Specify all externally visible behaviors
- Captures the dialog between a user and the system: <u>from the user's point of view</u>
- Use cases can help identify all functional requirements if sufficiently detailed

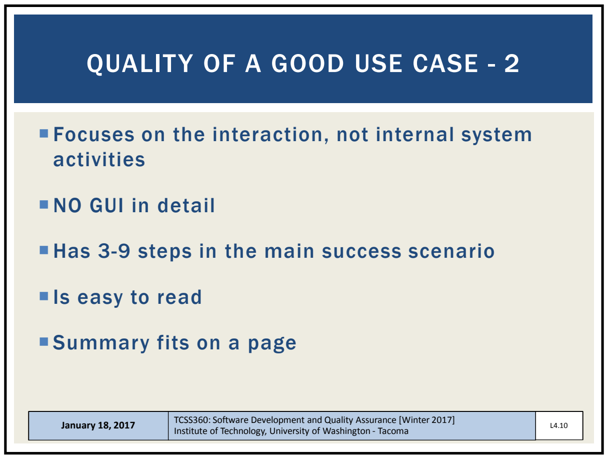
January 18, 2017 | TCSS360: Software Development and Quality Assurance [Winter 2017] Institute of Technology, University of Washington - Tacoma | L4.8



QUALITIES OF A GOOD USE CASE

- Starts with a request from an actor to the system
- Ends with the production of all the answers to the requests
- Defines the interactions (between system and actors) related to the functions
- Takes into account the actor's point of view, not the system's

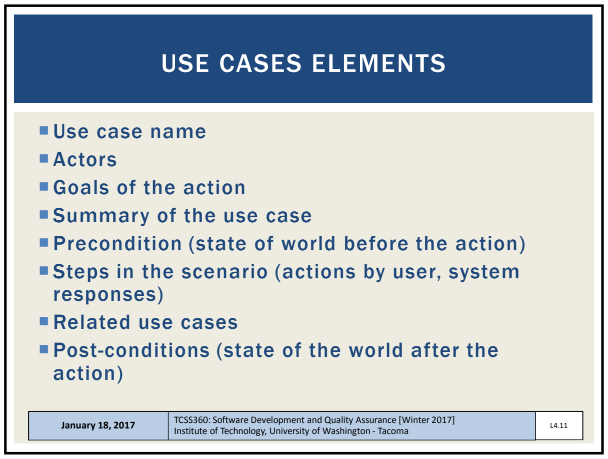
January 18, 2017 | TCSS360: Software Development and Quality Assurance [Winter 2017] Institute of Technology, University of Washington - Tacoma | L4.9



QUALITY OF A GOOD USE CASE - 2

- Focuses on the interaction, not internal system activities
- NO GUI in detail
- Has 3-9 steps in the main success scenario
- Is easy to read
- Summary fits on a page

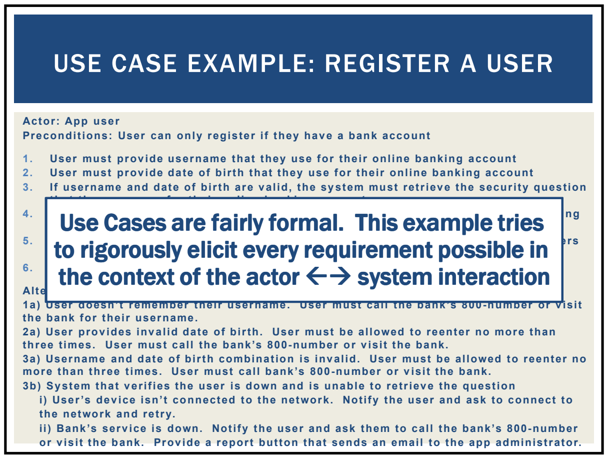
January 18, 2017 | TCSS360: Software Development and Quality Assurance [Winter 2017] Institute of Technology, University of Washington - Tacoma | L4.10



USE CASES ELEMENTS

- Use case name
- Actors
- Goals of the action
- Summary of the use case
- Precondition (state of world before the action)
- Steps in the scenario (actions by user, system responses)
- Related use cases
- Post-conditions (state of the world after the action)

January 18, 2017 | TCSS360: Software Development and Quality Assurance [Winter 2017] Institute of Technology, University of Washington - Tacoma | L4.11



USE CASE EXAMPLE: REGISTER A USER

Actor: App user
Preconditions: User can only register if they have a bank account

1. User must provide username that they use for their online banking account
2. User must provide date of birth that they use for their online banking account
3. If username and date of birth are valid, the system must retrieve the security question
4. [illegible] ng
5. [illegible] ers
6. [illegible]

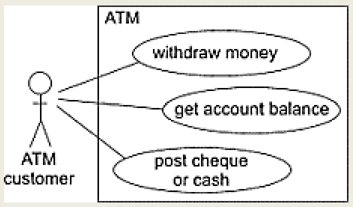
Use Cases are fairly formal. This example tries to rigorously elicit every requirement possible in the context of the actor ←→ system interaction

Alte[illegible]
1a) User doesn't remember their username. User must call the bank's 800-number or visit the bank for their username.
2a) User provides invalid date of birth. User must be allowed to reenter no more than three times. User must call the bank's 800-number or visit the bank.
3a) Username and date of birth combination is invalid. User must be allowed to reenter no more than three times. User must call bank's 800-number or visit the bank.
3b) System that verifies the user is down and is unable to retrieve the question
  i) User's device isn't connected to the network. Notify the user and ask to connect to the network and retry.
  ii) Bank's service is down. Notify the user and ask them to call the bank's 800-number or visit the bank. Provide a report button that sends an email to the app administrator.

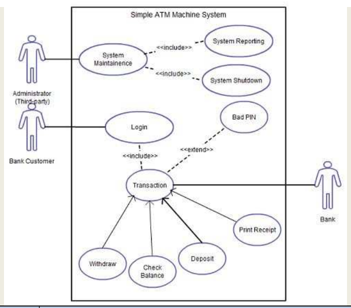
USE CASE DIAGRAMS

- From the Unified Modeling Language (UML)
- Stick figures standing in for actors
- Identifies
  - Different kind of users of a system
  - Different use cases
  - Interactions between one or more users and the system

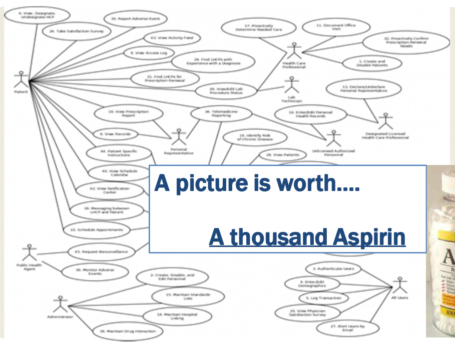
USE CASE DIAGRAM



USE CASE DIAGRAM - 2



USE CASE DIAGRAM - 3



A picture is worth....

A thousand Aspirin

MEASURING PRODUCTIVITY

- Count the number of functions (**function points**) completed per agile iteration (2 weeks)
- Productivity estimate
  - Assume the team can complete this many user stories per iteration
- But do all user stories require the same effort?
- Refinement: rank from 1 to 3 (ordinal measure)
- 1- straightforward, 2- medium, 3- very complex story
- **Ordinal measure:** ensures ranking, but the "ranks" don't carry mathematical meaning

METRICS

- **Ordinal measures**
- Is it meaningful to compute the average effort ranking?
- 1 + 2 + 1 + 3 + 2 + 1 + 1 / 7 =
- What does 1.57 mean?
- What if something is:
  just barely not a 2 (high effort 1) vs. a no effort 1 ?
- What is a 1.3, 2.1, 1.7, 2.7, 3.0?
- How can everyone on the team arrive at the same ranking?
- **Ratio measures:** time, length, temperature
- Developer A completes user stories (1-3) in: 27, 52, 33 minutes
- Average rate of user story completion = 37.33 minutes

Scales of Measurement

What is a *nominal scale?*	A scale that categorizes items
What is an *ordinal scale?*	A scale that categorizes and rank orders items
What is an *interval scale?*	A scale that categorizes and rank orders items, and has equal intervals
What is a *ratio scale?*	A scale that categorizes and rank orders items, has equal intervals, and a zero that means the absence or none of the thing being measured

METRICS - 2

- Velocity
  - Average number of function points per agile iteration a team completes
- Backlog
  - Collection of user stories not yet completed during an iteration
- Velocity gives all stakeholders an idea how many agile iterations a set of features will require to complete
- Helps establish reasonable expectations
  - Stop promising everything in 2 weeks…

SMART USER STORIES

- **S**pecific, **M**easurable, **A**chievable, **R**elevant, and **T**ime-boxed
- **Specific** vs. vague features
  - Vague feature: User can search for a movie
  - Specific feature: User can search for a movie by title
- **Measurable**
  - A specific and measurable feature will be testable
  - Are the following features testable?

SMART USER STORIES - 2

- Testable = Specific + Measurable
  - Feature: RottenPotatoes should have a good response time
  - Is this testable?
  - How would you test it?
  - Feature: When adding a movie to RottenPotatoes, 99% of new movies added should be accessible to everyone within 3 seconds
  - Is this testable?
  - How would you test it?

SMART USER STORIES - 3

- **Achievable**
  - User stories should be scoped so that they are achievable in one agile iteration
  - If team completes less than 1 user story per iteration, user stories need to be decomposed into smaller stories
- **Relevant**
  - User stories must have business value to one or more stakeholders
  - A feature has business relevance if you can drill down and answer at least 5 "Why" questions

SMART USER STORIES - 4

- Relevance example
  - Feature: When adding a movie, the user will be able to post a link to the their new movie review to their facebook wall.
  1. **Why add the facebook feature? More people will go with friends and enjoy the movie.**
  2. **Why does it matter if people enjoy the movie? We will sell more tickets**
  3. **Why sell more tickets? Because the theater will make more money**
  4. **Why make more money? So the theater doesn't go out of business**
  5. **Why does it matter that the theater is in business? If not, I have no job**
  - If five WHY questions can't be answered, it may not be important

SMART USER STORIES - 5

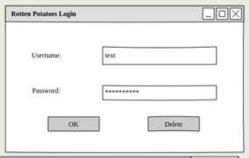
- **Timeboxed**: User stories have time limits, when exceeded:
  - Give up
  - Divide story into smaller user stories
  - Reschedule remaining functionality using a new estimate
  - Reduce scope: ask customer to identify highest value parts of story, than can be done quickly
- Motivation
  - Extremely easy to underestimate length of a software project
  - Without careful accounting entire project could be late and fail
  - When exceeding a story time budget, refactoring the user story helps continually refine the project scope and maintain "tractability"

UI SKETCHES AND STORYBOARDS

- BDD: In addition to user stories need to capture what the UI looks like:

```
Feature: Enable a user to login to RottenPotatoes
  As a user
  So that I can interact with RottenPotatoes as userid
  I want to login by specifying my password and userid
```

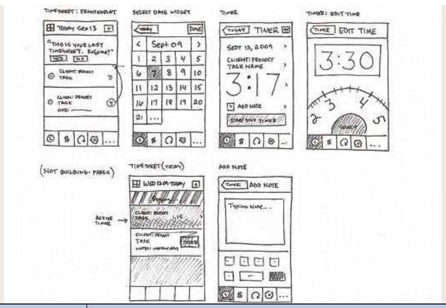
- UI equivalent of index cards
- Engage nontechnical stakeholders
  - Encourage them to sketch
- Make sketches for all user stories
- Helps specify the details
- Easier to get sketches right than code



UI SKETCHES AND STORYBOARDS - 2

- "Pencil" is quite nice for UI mockups and diagrams
  - Though perhaps it is almost too much effort
  - For those who don't like to draw by hand
  - http://pencil.evolus.vn/
- Storyboarding: Tree or graph of screens driven by different user choices
  - Name from "storyboarding" for movie development
  - Try to capture user interactions with:
    - Pages or sections of pages
    - Forms and buttons
    - Popups

STORYBOARD EXAMPLE: MOBILE TIMESHEET APP – ADD RECORD

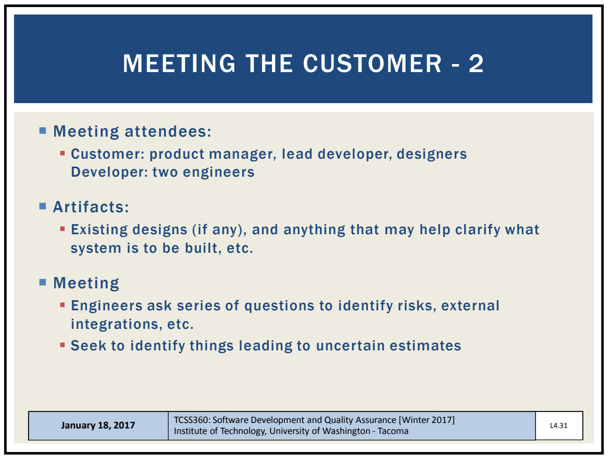


UI SKETCHES AND STORYBOARDS - 3

- Completed UI sketches and storyboards can then be implemented using HTML/CSS if a WEB GUI
- Sketches provide UI definition to speed the GUI implementation
- UI developer doesn't have to create the UI design
  - Just implement the UI from sketches / storyboards stakeholders have already provided
- TRUE/FALSE:
- The purpose of UI sketches/storyboards is to debug the UI before it is programmed?

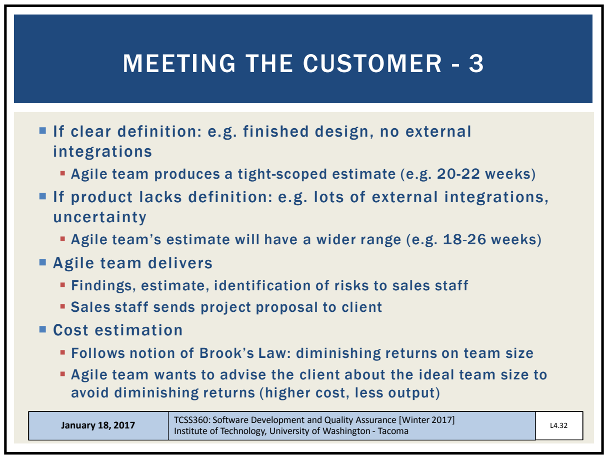
PREPARING TO MEET THE CUSTOMER

- Agile: customer collaboration over contract negotiation
- Never commit to delivering features X, Y, Z, by date D
- Initial contact: 30 to 60 minute phone call
- Explain the agile process
- Agile works on a time and materials basis, not a fixed bid basis
- Agile team asks customer for high level description of the system to be built
- If agile team and customer are a good fit, typically a 90-minute in-person scoping meeting is scheduled



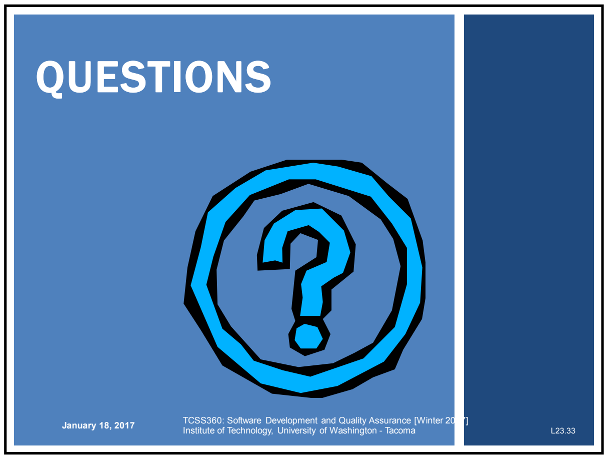
MEETING THE CUSTOMER - 2

- Meeting attendees:
  - Customer: product manager, lead developer, designers
    Developer: two engineers
- Artifacts:
  - Existing designs (if any), and anything that may help clarify what system is to be built, etc.
- Meeting
  - Engineers ask series of questions to identify risks, external integrations, etc.
  - Seek to identify things leading to uncertain estimates



MEETING THE CUSTOMER - 3

- If clear definition: e.g. finished design, no external integrations
  - Agile team produces a tight-scoped estimate (e.g. 20-22 weeks)
- If product lacks definition: e.g. lots of external integrations, uncertainty
  - Agile team's estimate will have a wider range (e.g. 18-26 weeks)
- Agile team delivers
  - Findings, estimate, identification of risks to sales staff
  - Sales staff sends project proposal to client
- Cost estimation
  - Follows notion of Brook's Law: diminishing returns on team size
  - Agile team wants to advise the client about the ideal team size to avoid diminishing returns (higher cost, less output)



QUESTIONS