Requirements and Specification

October 7, 2020

Byung-Gon Chun

(Credit: Many Slides from UCB CS169
taught by Armando Fox, David Patterson, and George Necula)

Requirements Engineering

- The most important single part of building a software system is deciding **what to build**
  - Cripples the process if done wrong
  - Costly to rectify later

- Must be adapted to the software process
  - Elaborate requirements/specs for plan and document
  - Partial set of user stories for iterative processes

Determining Stakeholders and Needs

- Must determine stakeholders
  - Anyone who benefits from the system developed
  - E.g., who's client and who's user ?
- Try to understand what their needs are
- Reconcile different needs/points of view

Techniques

- Interviewing
- User stories
- Strawmen
- Prototyping
- Pretotyping

Interviewing

- One path is obvious
  - Sit down with client/user and ask questions
  - Listen to what they say, and what they don’t say
- A less obvious path
  - Master-apprentice relationship
  - Have them teach you what they do
  - Go to workplace and watch them do the task
- In all types of interviews, get details
  - Ask for copies of reports, logs, emails on process
  - These may support, fill in, or contradict what the user said

Disadvantages of Talking

- Interviews are useful, but users/clients may
  - Not have the vocabulary to tell you what they need
  - Not know enough about computer science to understand what is possible or impossible
- Good idea to gather requirements in other ways, too

User Stories

- Describe usage scenarios of software
  - Title, short description in a certain format

- Each user story has acceptance tests
  - Concrete **instance(s)** of the story
  - Will tell you when the customer thinks story is done

User Story (Style #1)

- Feature
- Scenario
- Acceptance tests (list **one or more acceptance tests with concrete values for the parameters)**

User Story Example

- Feature: Delete account
- Scenario: ..., confirmation dialog box, ...
- Acceptance test(s)

User Story (Style #2)

- Feature name
- Actor(s)
- Preconditions (in what system state is this user story applicable)
- Triggers (what initiates the story)
- Scenario(s)
- Postconditions (what is the system state after the story is done)
- Exceptions
- Acceptance tests (list **one or more acceptance tests with concrete values for the parameters, and concrete assertions that you will make to verify the postconditions**).

User Story Example

- Feature: Create Tour
- Actors: Traveler or Guide User
- Precondition: The traveler or the guide has to be a registered user and logged in
- Trigger: User clicks on the “Create Tour” button
- Scenario:
  1. The page displays a map with all the map markers that the route that the user has created
  2. On the left you can find buttons to access all the information of the tour such as: Name of the tour, begin and end times of the tour, description of the tour, description of the places of route, dates in which the tour is available, transportation means.
- Exceptions: The user does not fill out all fields to create the Tour
- Acceptance Test:

  Given the user has fill out all the required fields to create the tour

  When the user clicks on the “Create Tour” Button

  Then the user should see “Your tour has been created”

User Story (Style #3: Connextra Format)

Feature: [name]

As a [kind of actor (who)]

I want to [do some task (what)]

So that [I can achieve some goal (why)]

Feature: Add a movie to MovieRank

As a movie fan

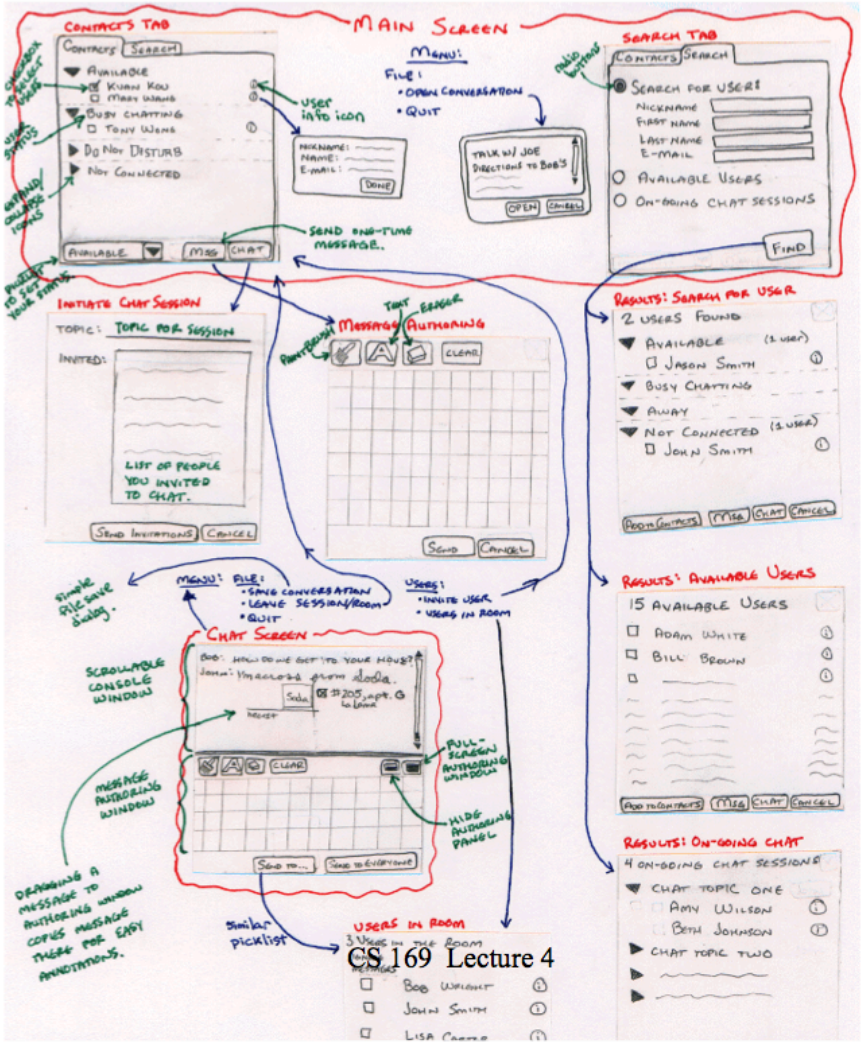
I want to add a movie to MovieRank service

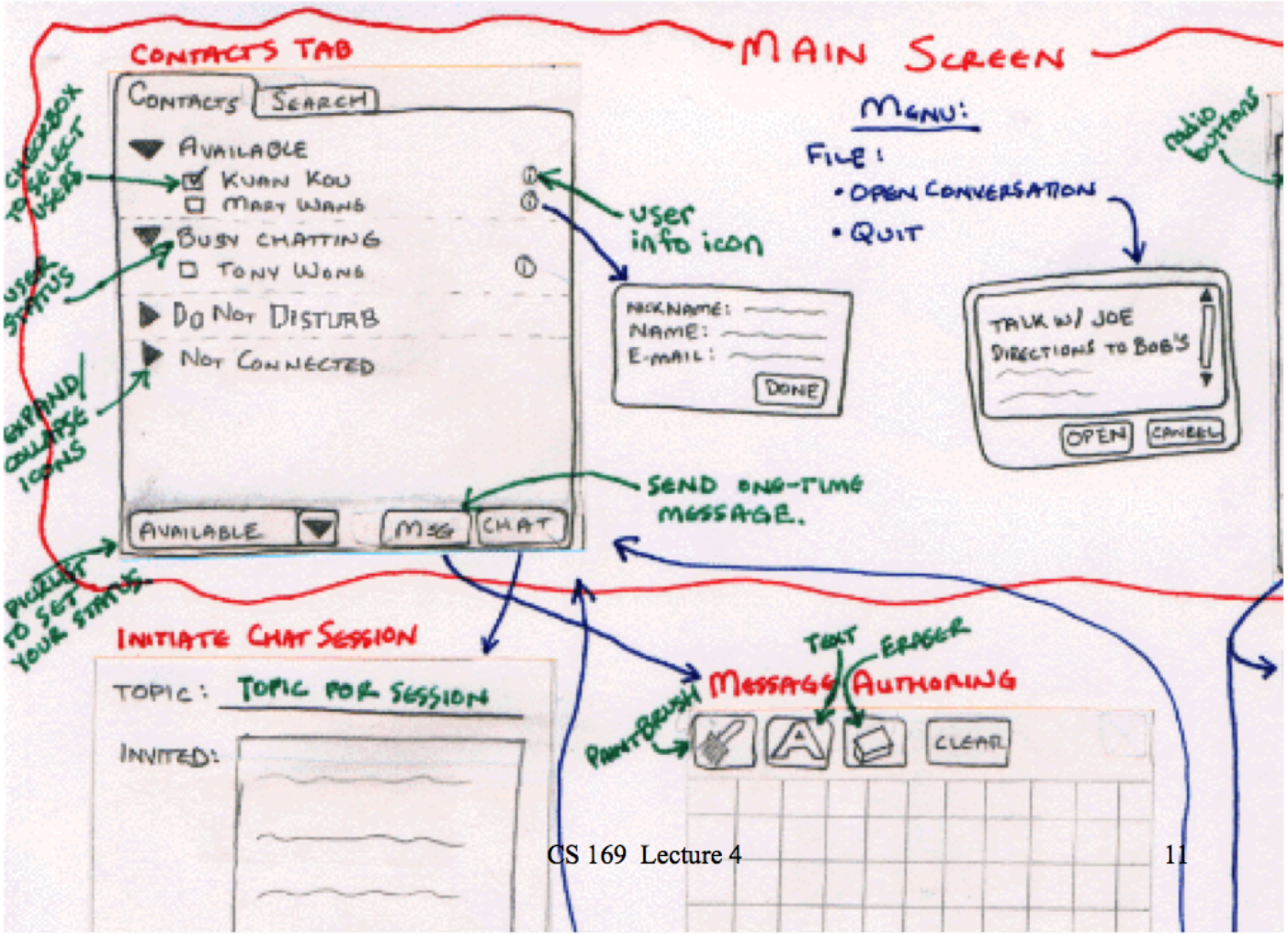
So that I can share a movie with other movie fans

Strawmen

- Sketch the product for the user/client
  - Storyboards
  - Flowcharts
  - HTML mock-ups
  - Illustrate major events/interfaces/actions

- Anything to convey ideas **without writing code**!


MAIN SCREEN
CONTACTS TAB
CONTACTS
SEARCH
AVAILABLE
KUMAR KOU
MARY WANG
BUSY CHATTING
TONY WONG
DO NOT DISTURB
NOT CONNECTED
AVAILABLE
MSG
CHAT
MENU:
FILE:
• OPEN CONVERSATION
• QUIT
NICKNAME:
NAME:
E-MAIL:
DONE
TALK W/ JOE
DIRECTIONS TO BOB'S
OPEN
CANCEL
SEND ONE-TIME MESSAGE.
SEARCH TAB
CONTACTS
SEARCH
SEARCH FOR USER:
NICKNAME
FIRST NAME
LAST NAME
E-MAIL
AVAILABLE USERS
ON-GOING CHAT SESSIONS
FIND
INITIATE CHAT SESSION
TOPIC: TOPIC FOR SESSION
INVITED:
LIST OF PEOPLE YOU INVITED TO CHAT.
SEND INVITATIONS
CANCEL
MESSAGE AUTHORING
CLEAR
SEND
CANCEL
RESULTS: SEARCH FOR USER
2 USERS FOUND
AVAILABLE (1 USER)
JASON SMITH
BUSY CHATTING
AWAY
NOT CONNECTED (1 USER)
JOHN SMITH
ADD TO CONTACTS
MSG
CHAT
CANCEL
RESULTS: AVAILABLE USERS
15 AVAILABLE USERS
ADAM WHITE
BILL BROWN
ADD TO CONTACTS
MSG
CHAT
CANCEL
MENU: FILE:
• SAVE CONVERSATION
• LEAVE SESSION/ROOM
• QUIT
USERS:
• INVITE USER
• USERS IN ROOM
CHAT SCREEN
SCROLLABLE CONSOLE WINDOW
MESSAGE AUTHORING WINDOW
CLEAR
FULL-SCREEN AUTHORING WINDOW
HIDE AUTHORING PANEL
SEND TO...
SEND TO EVERYONE
DRAGGING A MESSAGE TO AUTHORING WINDOW COPIES MESSAGE THERE FOR EASY MODIFICATIONS.
USERS IN ROOM
3 USERS IN THE ROOM
BOB WRIGHT
JOHN SMITH
RESULTS: ON-GOING CHAT
4 ON-GOING CHAT SESSIONS
CHAT TOPIC ONE
AMY WILSON
BETH JOHNSON
CHAT TOPIC TWO


CONTACTS TAB
MAIN SCREEN
CONTACTS
SEARCH
AVAILABLE
KUAN KOU
MARY WANG
BUSY CHATTING
TONY WONG
DO NOT DISTURB
NOT CONNECTED
AVAILABLE
MSG
CHAT
CHECKBOX TO SELECT USERS
USER STATUS
EXPAND/ COLLAPSE ICONS
PULLDOWN TO SET YOUR STATUS
USER INFO ICON
MENU:
FILE:
• OPEN CONVERSATION
• QUIT
RADIO BUTTONS
NICKNAME:
NAME:
E-MAIL:
DONE
TALK W/ JOE
DIRECTIONS TO BOB'S
OPEN
CANCEL
SEND ONE-TIME MESSAGE.
INITIATE CHAT SESSION
TOPIC: TOPIC FOR SESSION
INVITED:
MESSAGE AUTHORING
TEXT
ERASER
PAINTBRUSH
CLEAR

Summary of Requirements

- Find out what users/clients need
  - Not necessarily what they say they want

- Use
  - Interviews
  - User stories
  - Strawmen
  - As appropriate . . .

Specifications

- Describe the functionality of the product
  - Precisely
  - Covering all circumstances
- Move from the finite to the infinite
  - Finite examples (acceptance tests) to infinite set of possible cases
  - This is not easy

Different Views of Specifications

- Developer's
  - Specification must be detailed enough to be implementable
  - Unambiguous
  - Self-consistent
- Client's/user's
  - Specifications must be comprehensible
  - Usually means not too technical
- Legal
  - Specification can be a contract
  - Should include acceptance criteria
    - If the software passes tests X, Y, and Z, it will be accepted

Problems with Informal Specs

- Informal Specs: written in natural language
- Informal specs of any size inevitably suffer from serious problems
  - Omissions
    - Something missing
  - Ambiguities
    - Something open to multiple interpretations
  - Contradictions
    - Spec says both “do A” and “do not do A”

These problems will be faithfully implemented in the software unless found in the spec

Informal Specifications Example

“If sales for current month are below target sales, then report is to be printed, unless difference between target sales and actual sales is less than half of difference between target sales and actual sales in previous month, or if difference between target sales and actual sales for the current month is under 5%”

Informal Specifications Example

- “If sales for current month are below target sales, then report is to be printed, unless difference between target sales and actual sales is less than half of difference between target sales and actual sales in previous month, or if difference between target sales and actual sales for the current month is under 5%”

What are sales? Orders received but not yet paid for? Only orders paid for?

Informal Specifications Example

- “If sales for current month are below target sales, then report is to be printed, unless difference between target sales and actual sales is less than half of difference between target sales and actual sales in previous month, or if difference between target sales and actual sales for the current month is under 5%”

Specification implies, but does not say, that there are monthly sales targets. Are there separate monthly targets, or is the monthly target e.g., 1/3 of the quarterly target?

Informal Specifications Example

"If sales for current month are below target sales, then report is to be printed, unless difference between target sales and actual sales is less than half of difference between target sales and actual sales in previous month, or if difference between target sales and actual sales for the current month is under 5%"

"unless A or B" means:

- If!(A||B)
- If(!A||B)
- May be the comma before "or" matters

Informal Specifications Example

“If sales for current month are below target sales, then report is to be printed, unless difference between target sales and actual sales is less than half of difference between target sales and actual sales in previous month, or if difference between target sales and actual sales for the current month is under 5% ”

5% of the target sales, or of the actual sales?

Comments on Informal Specification

- Informal specification is universally disliked
  - By academics
  - By “how to” authors
- Informal specification is also widely practiced – Why?

Why Do People Use Informal Specs?

- The common language is natural language
  - Customers can't read formal specs
  - Neither can most programmers
  - Or most managers
  - A least-common denominator effect takes hold
- Truly formal specs are very time-consuming
  - And hard to understand
  - And overkill for most projects

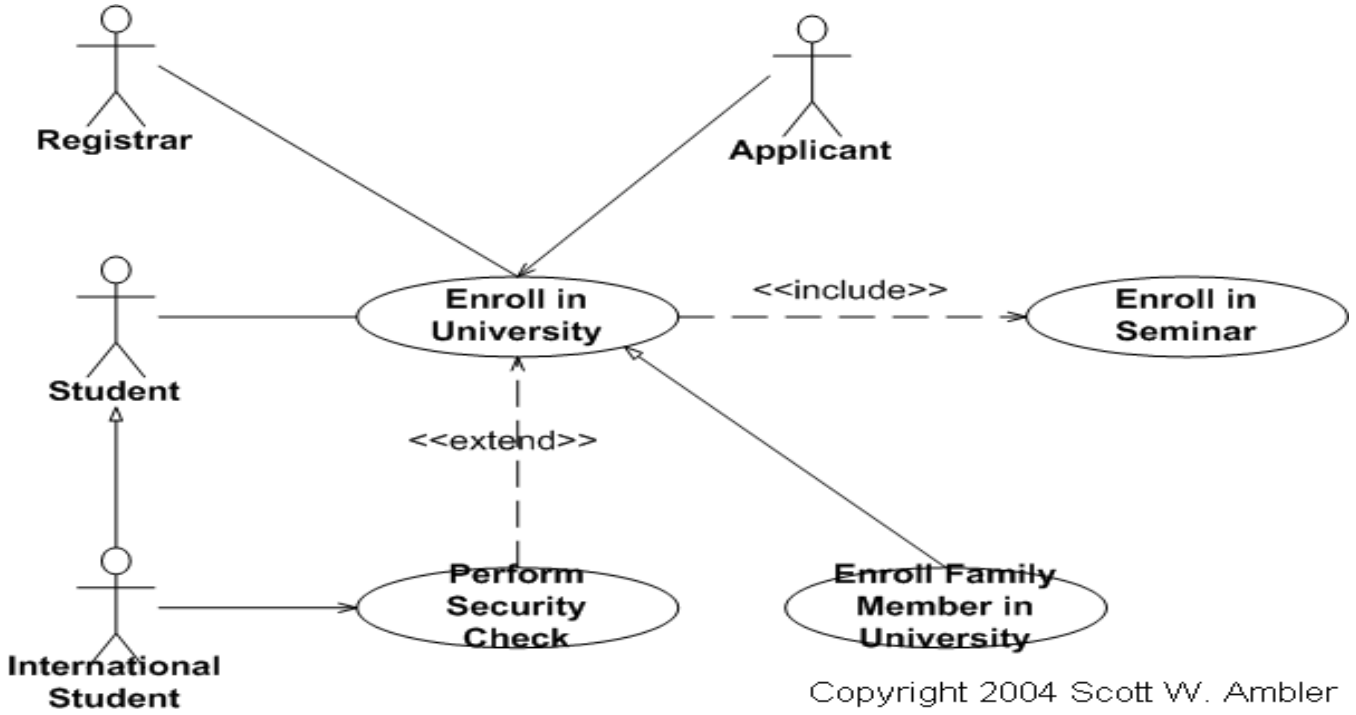
Formal Spec Example (TLA+)

```
---------- MODULE Euclid ----------
EXTENDS Integers

p | q == \E d \in 1..q : q = p * d
Divisors(q) == {d \in 1..q : d | q}
Maximum(S) == CHOOSE x \in S : \A y \in S : x \geq
y
GCD(p,q) == Maximum(Divisors(p) \cap Divisors(q))
Number == Nat \ {0}

CONSTANTS M, N
VARIABLES x, y

Init == (x = M) /\ (y = N)

Next == \/ /\ x < y
        /\ y' = y - x
        /\ x' = x
     \/ /\ y < x
        /\ x' = x-y
        /\ y' = y

Spec == Init /\ [][Next]_<<x,y>>
ResultCorrect == (x = y) => x = GCD(M, N)
THEOREM Correctness == Spec =>
[]ResultCorrect

==================================
```

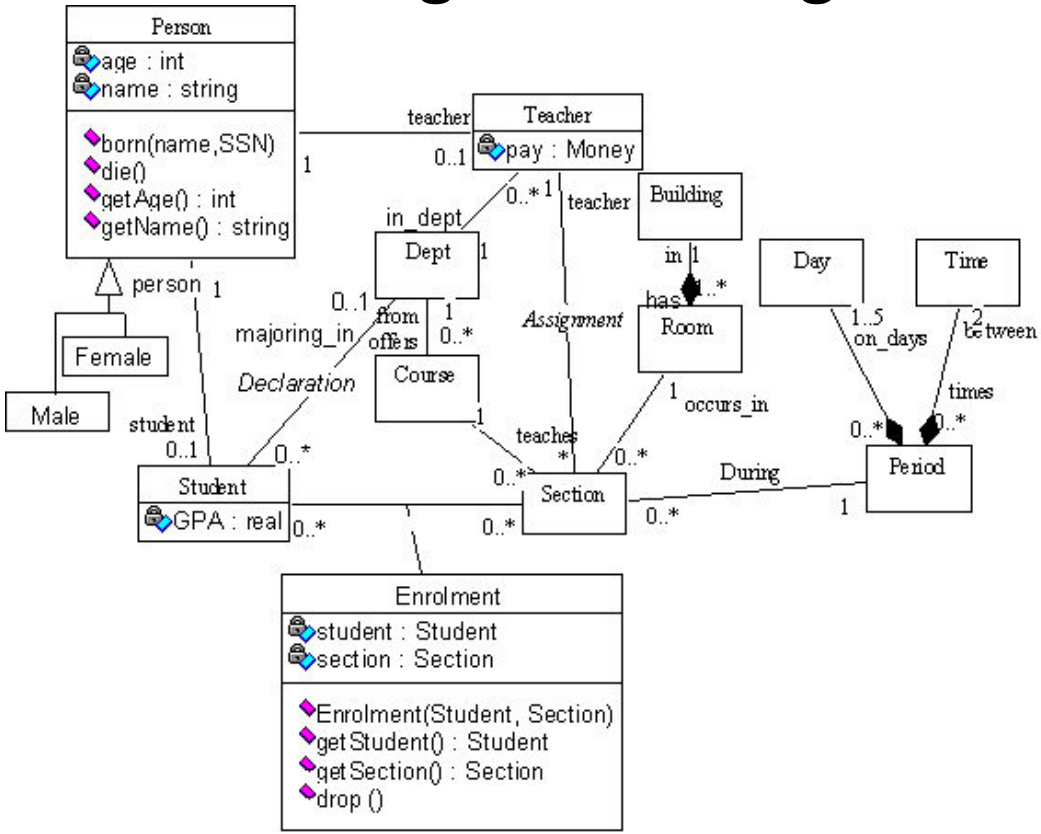
Semi-Formal Specs

- Best current practice is “semi-formal” specs
  - Allows more precision than natural language where desired
- Usually a boxes-and-arrows notation
  - Must pay attention to:
  - What boxes mean
  - What arrows mean
  - Different in different systems!

Semi-formal Spec Example (UML Use Case)



Semi-formal Spec Example
(UML Class Diagram – Design Phase)



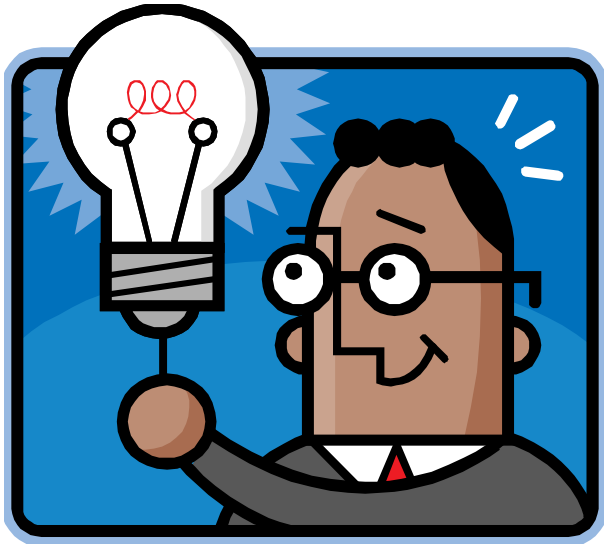
SMART User Stories

Creating User Stories

- How do you know if you have a good user story vs. bad user story?
  - Right size?
  - Not too hard?
  - Is worthwhile?

SMART stories

- **S**pecific
- **M**easurable
- **A**chievable
  (ideally, implement in
  1 iteration)
- **R**elevant
  (“the 5 why’s”)
- **T**imeboxed
  (know when to give up)



Specific & Measurable

- Each scenario testable
  - Implies known good input and expected results exist
- Anti-example: "UI should be user-friendly"
- Example: Given/When/Then.
  1. *Given* some specific starting condition(s),
  2. *When* I do X,
  3. *Then* one or more specific thing(s) should happen

Achievable

- Complete in 1 iteration
- If can't deliver feature in 1 iteration, deliver subset of stories
  - Always aim for **working code** @ end of iteration
- If <1 story per iteration, need to improve point estimation per story

Relevant: “business value”

- Discover business value, or kill the story:
  - Protect revenue
  - Increase revenue
  - Manage cost
  - Increase brand value
  - Making the product remarkable
  - Providing more value to your customers

5 Whys to Find Relevance

- *Show patron's Facebook friends*

  As a box office manager

  So that I can induce a patron to buy a ticket

  I want to show her which Facebook friends are going to a given show

  1. *Why?*
  2. *Why?*
  3. *Why?*
  4. *Why?*
  5. *Why?*

5 Whys to Find Relevance

- *Show patron's Facebook friends*

  As a box office manag

  So that I can induce a
  buy a ticket

  I want to show her w
  friends are going to

  1. *Why?*
  2. *Why?*
  3. *Why?*
  4. *Why?*
  5. *Why?*

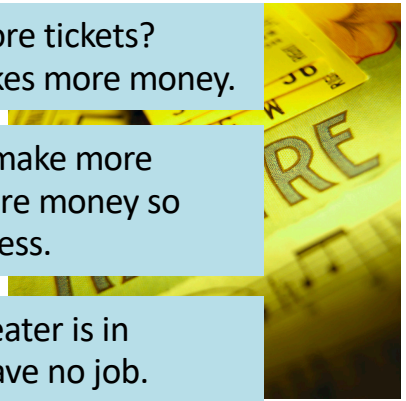
1. Why add the Facebook feature? As box office manager, I think more people will go with friends and enjoy the show more.

2. Why does it matter if they enjoy the show more? I think we will sell more tickets.

3. Why do you want to sell more tickets? Because then the theater makes more money.

4. Why does theater want to make more money? We want to make more money so that we don't go out of business.

5. Why does it matter that theater is in business next year? If not, I have no job.



Behavior-Driven Design and User Stories

Why do SW Projects Fail?

- Don't do what customers want
- Or projects are late
- Or over budget
- Or hard to maintain and evolve
- Or all of the above
- How does Agile try to avoid failure?

Agile Lifecycle Review

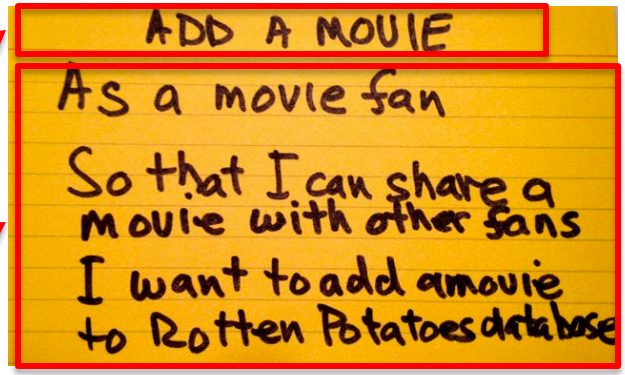
- Work closely, continuously with stakeholders to develop requirements, tests
  - Users, customers, developers, maintenance programmers, operators, project managers, …
- Maintain working prototype while deploying new features every **iteration(sprint)**
  - Typically every 1 or 2 weeks
- Check with stakeholders on what's next, to validate building right thing (vs. verify)

Behavior-Driven Design (BDD)

- BDD asks questions about behavior of app *before and during development* to reduce miscommunication
- Requirements written down as *user stories*
  - Lightweight descriptions of how app used
- BDD concentrates on *behavior* of app vs. *implementation* of app
  - Test Driven Design (TDD) tests implementation

User Stories

- 1-3 sentences in everyday language
  - Fits on 3” x 5” index card
  - Written by/with customer
- “Connextra” format:
  - Feature name
  - As a [kind of stakeholder],
    So that [I can achieve some goal],
    I want to [do some task]
  - 3 phrases must be there, can be in any order
- Idea: user story can be formulated as *acceptance test before* code is written



Why 3x5 Cards?

- (from User Interface community)
- Nonthreatening => all stakeholders participate in brainstorming
- Easy to rearrange => all stakeholders participate in prioritization
- Since stories must be short, easy to change during development
  - As often get new insights during development

Different stakeholders may describe behavior differently

- *See which of my friends are going to a show*
  - As a theatergoer
  - So that I can enjoy the show with my friends
  - I want to see which of my Facebook friends are attending a given show
- *Show patron's Facebook friends*
  - As a box office manager
  - So that I can induce a patron to buy a ticket
  - I want to show her which of her Facebook friends are going to a given show

Product Backlog

- Real systems have 100s of user stories
- *Backlog*: User Stories not yet completed
- Prioritize so most valuable items highest
- Organize so they match SW releases over time

Related Issue

- Spike
  - Short investigation into technique or problem
    - E.g. spike on recommendation algorithms
  - After spike done, code *must* be thrown away
    - Now that know approach you want, write it right

Lo-Fi UI Sketches and Storyboards

Building Successful UI

- SaaS apps often faces users

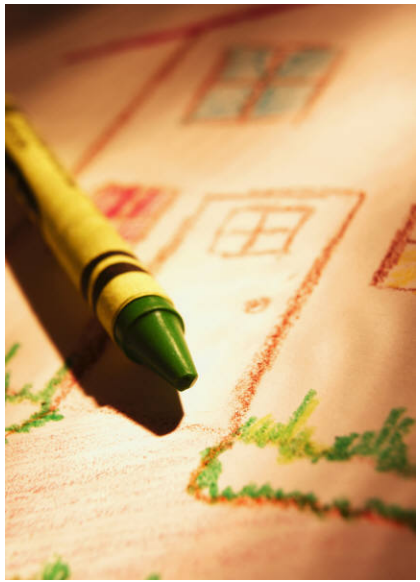
$\Rightarrow$ User stories need User Interface (UI)

- How get customer to participate in UI design so is happy when complete?
  - Avoid WISBNWIW* UI?
  - UI version of 3x5 cards?
- How show interactivity without building prototype?

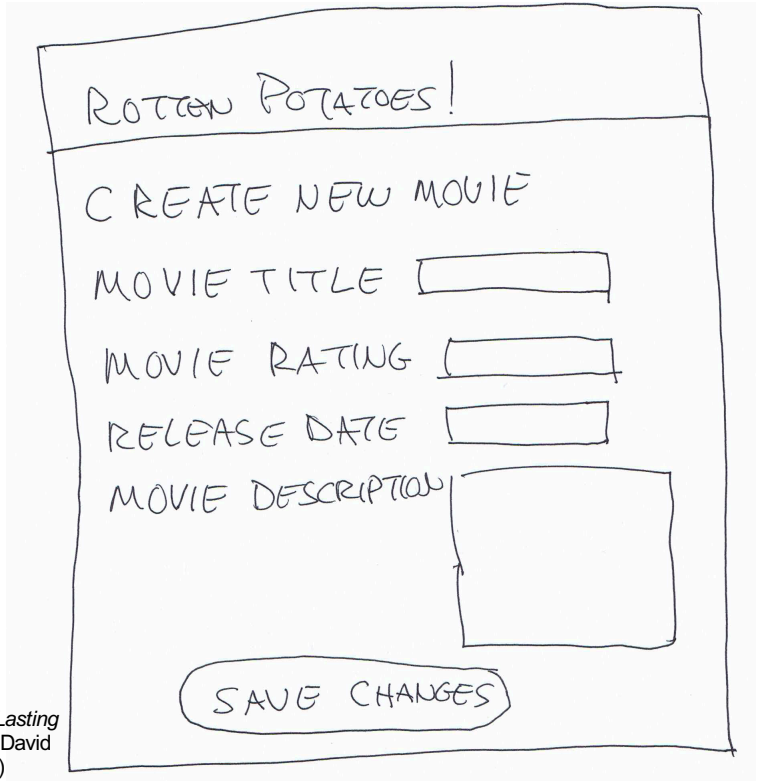
* What-I-Said-But-Not-What-I-Want

SaaS User Interface Design

- UI Sketches: pen and paper drawings or “Lo-Fi UI”



Lo-Fi UI Example

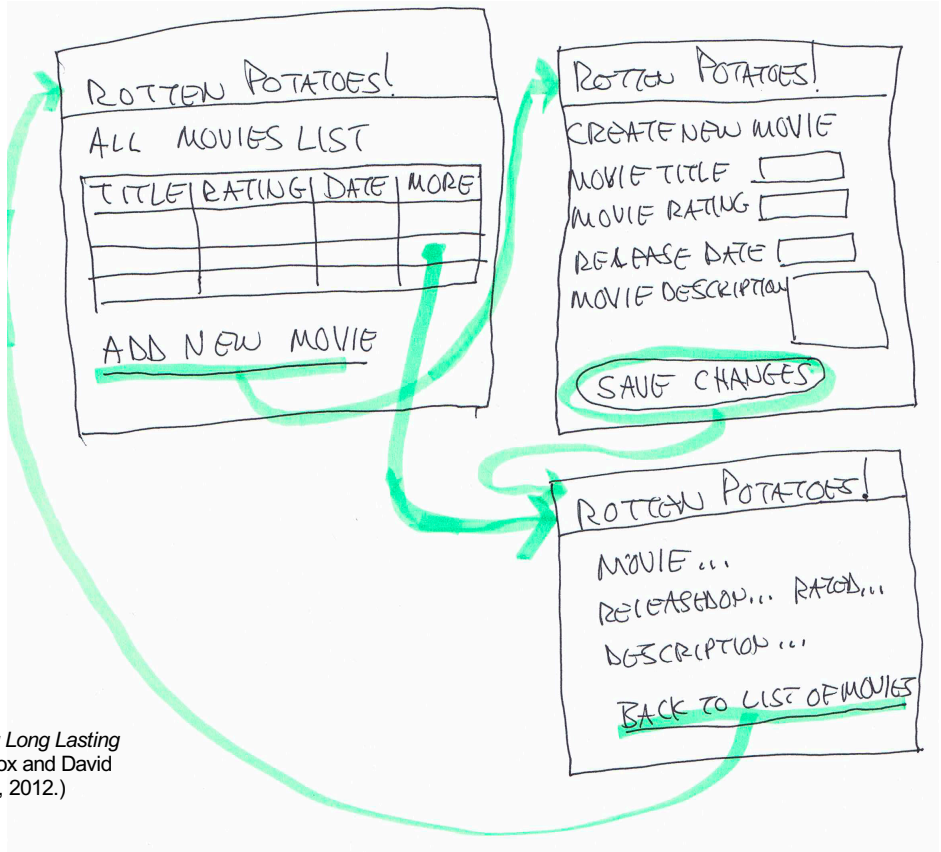


(Figure 4.3, *Engineering Long Lasting Software* by Armando Fox and David Patterson, Alpha edition, 2012.)

Storyboards

- Need to show how UI changes based on user actions
- HCI => “storyboards”
- Like scenes in a movie
- But not linear

Example Storyboard

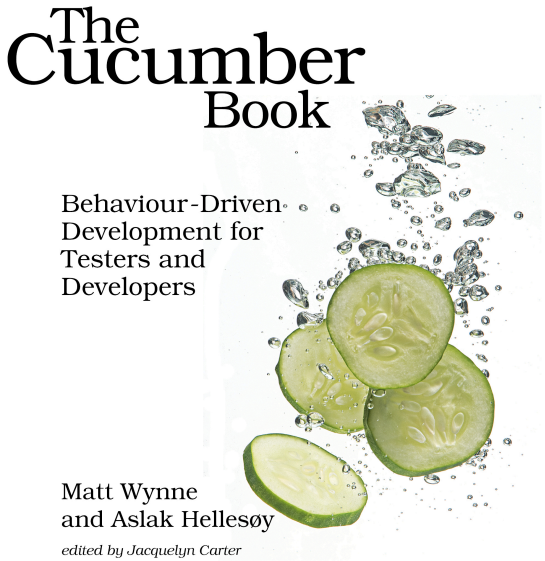


(Figure 4.4, *Engineering Long Lasting Software* by Armando Fox and David Patterson, Alpha edition, 2012.)

Lo-Fi to HTML

- Tedious to do sketches and storyboards, but easier than producing HTML!
  - Also less intimidating to nontechnical stakeholders => More likely to suggest changes to UI if not code behind it
  - More likely to be happy with ultimate UI
- Next step: CSS (Cascading Style Sheets)
  - Make it pretty *after* it works

Cucumber: A BDD Tool



User stories
=> Acceptance Tests?

- Wouldn’t it be great to automatically map 3x5 card user stories into tests for user to decide if accept the app?
- How would you match the English text to test code?
- How could you run the tests without a human in the loop to perform the actions?

Cucumber: Big Idea

- Tests from customer-friendly user stories
  - Acceptance: ensure satisfied customer
  - Integration: ensure interfaces between modules consistent assumptions, communicate correctly.
- Cucumber meets halfway between customer and developer
  - User stories not code, so clear to customer and can be used to reach agreement
  - Also not completely freeform, so can connect to real tests

https://github.com/cucumber/cucumber/wiki

Example User Story
(Cucumber understands Gherkin)

```
Feature: User can manually add movie
```

1 Feature

```
Scenario: Add a movie
  Given I am on the RottenPotatoes home page
  When I follow "Add new movie"
  Then I should be on the Create New Movie page
  When I fill in "Title" with "Men In Black"
  And I select "PG-13" from "Rating"
  And I press "Save Changes"
  Then I should be on the RottenPotatoes home page
  And I should see "Men In Black"
```

≥1 Scenarios / Feature

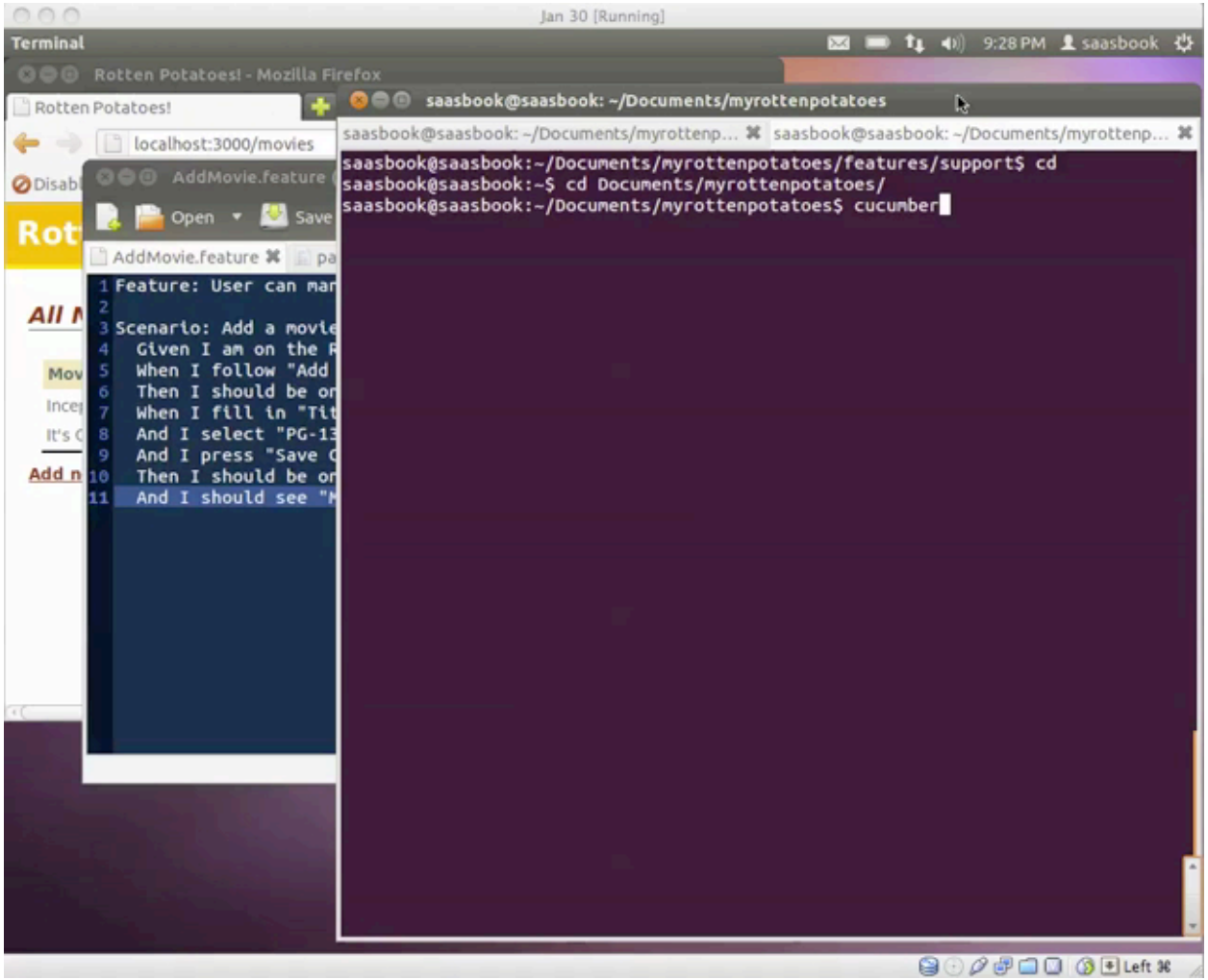
3 to 8 Steps / Scenario

Red-Yellow-Green Analysis

- Cucumber colors steps
  - Green for passing
  - Yellow for not yet implemented
  - Red for failing

- Once a user story is written, we can try to run it immediately
- Steps may be initially either in Red or Yellow
- Goal: Make all steps green for pass

  (Hence green vegetable for name of tool)


Jan 30 [Running]
Terminal
9:28 PM
saasbook
Rotten Potatoes! - Mozilla Firefox
Rotten Potatoes!
localhost:3000/movies
AddMovie.feature
Open
Save
AddMovie.feature
1 Feature: User can mar
2
3 Scenario: Add a movie
4 Given I am on the R
5 When I follow "Add
6 Then I should be or
7 When I fill in "Tit
8 And I select "PG-13
9 And I press "Save C
10 Then I should be or
11 And I should see "M
saasbook@saasbook: ~/Documents/myrottenpotatoes
saasbook@saasbook: ~/Documents/myrottenp...
saasbook@saasbook: ~/Documents/myrottenp...
saasbook@saasbook:~/Documents/myrottenpotatoes/features/support$ cd
saasbook@saasbook:~$ cd Documents/myrottenpotatoes/
saasbook@saasbook:~/Documents/myrottenpotatoes$ cucumber
Left ⌘