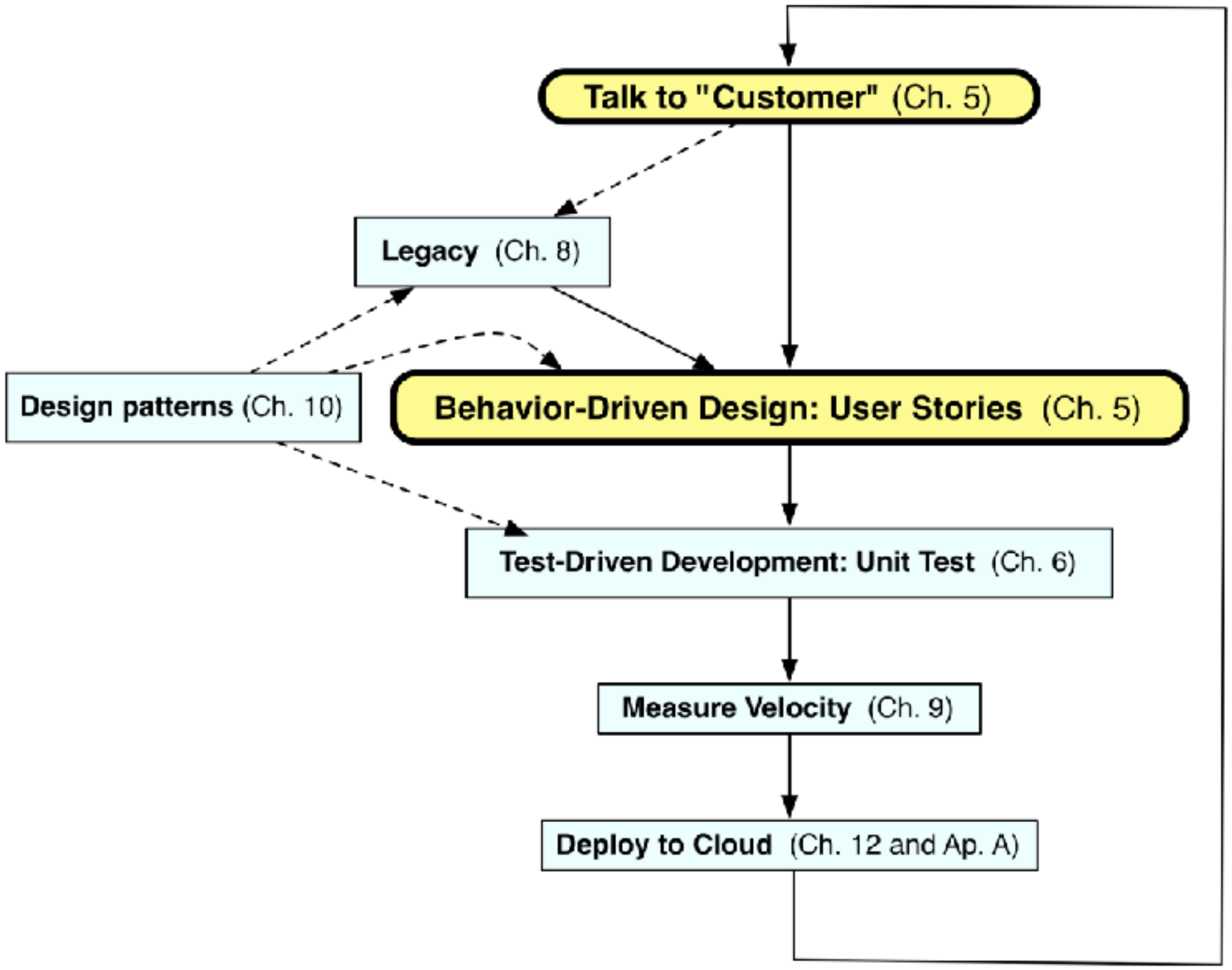
Introduction to Behavior-Driven Design and User Stories

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
- Inspired Agile Lifecycle

Agile Iteration

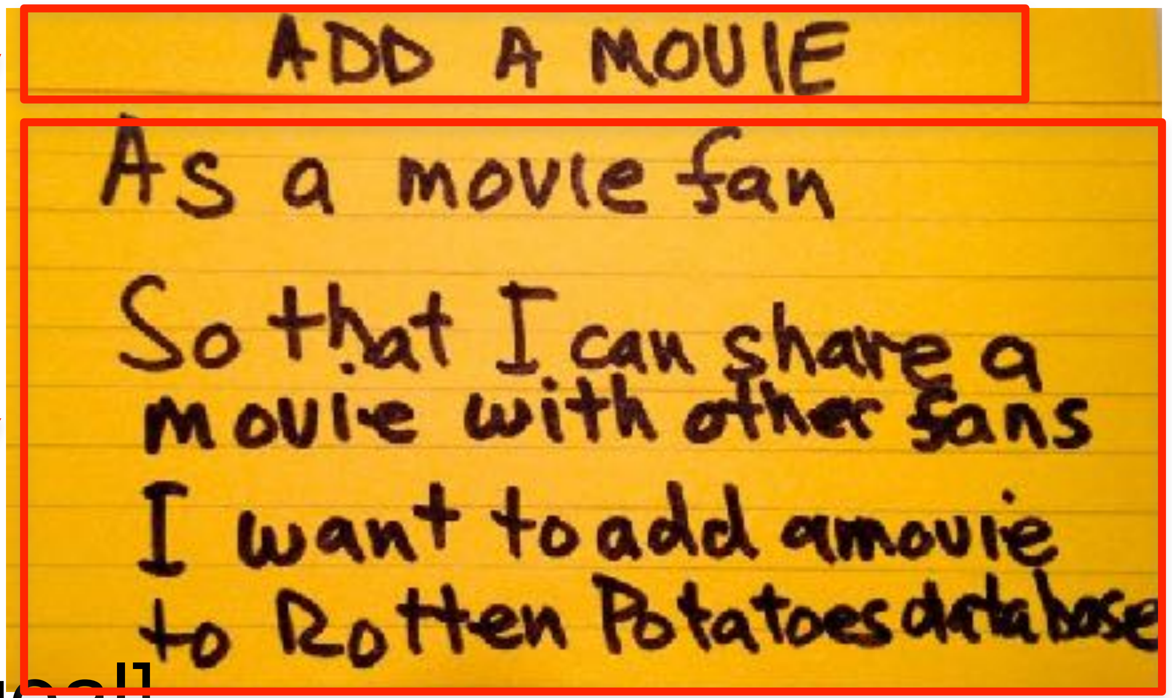

Talk to "Customer" (Ch. 5)
Legacy (Ch. 8)
Design patterns (Ch. 10)
Behavior-Driven Design: User Stories (Ch. 5)
Test-Driven Development: Unit Test (Ch. 6)
Measure Velocity (Ch. 9)
Deploy to Cloud (Ch. 12 and Ap. A)

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 or TDD (next chapter) 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

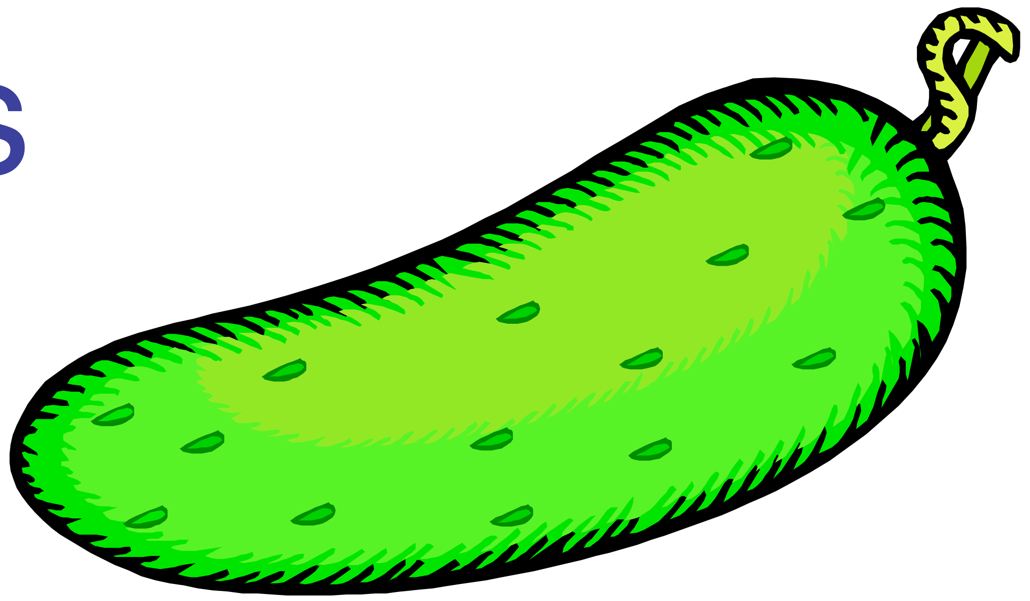

ADD A MOVIE

As a movie fan
So that I can share a movie with other fans
I want to add a movie to Rotten Potatoes database

Product Backlog

- Real systems have 100s of user stories
- *Backlog*: User Stories not yet completed
  - (We'll see Backlog again with Pivotal Tracker)
- Prioritize so most valuable items highest
- Organize so they match SW releases over time

Going from Stories to Tests- Introducing Cucumber



Big Idea

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

Example User Story

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

1 Feature

```
Scenario: Add a movie
```

≥1 Scenarios / Feature

```
  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"
```

3 to 8 Steps / Scenario

User Story, Feature, and Steps

- User story: refers to a single feature
- Feature: 1 or more scenarios that show different ways a feature is used
  - Keywords `Feature` and `Scenario` identify the respective components
- Scenario: 3 to 8 steps that describe scenario
- Step definitions: Ruby code that tests steps
  - Usually many steps per step definition

Cucumber: 5 Step Keywords

1. Given steps represent the state of the world before an event: preconditions
2. When steps represent the event (e.g., push a button)
3. Then steps represent the expected outcomes; check if its true
4. / 5. And and But extend the previous step

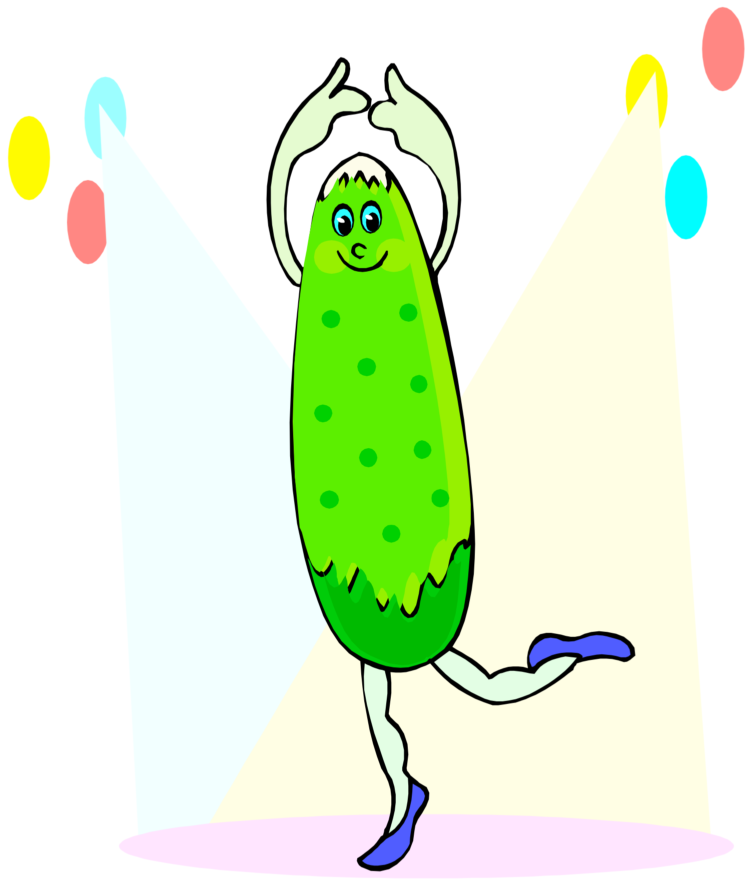
Steps, Step Definitions, and Regular Expressions

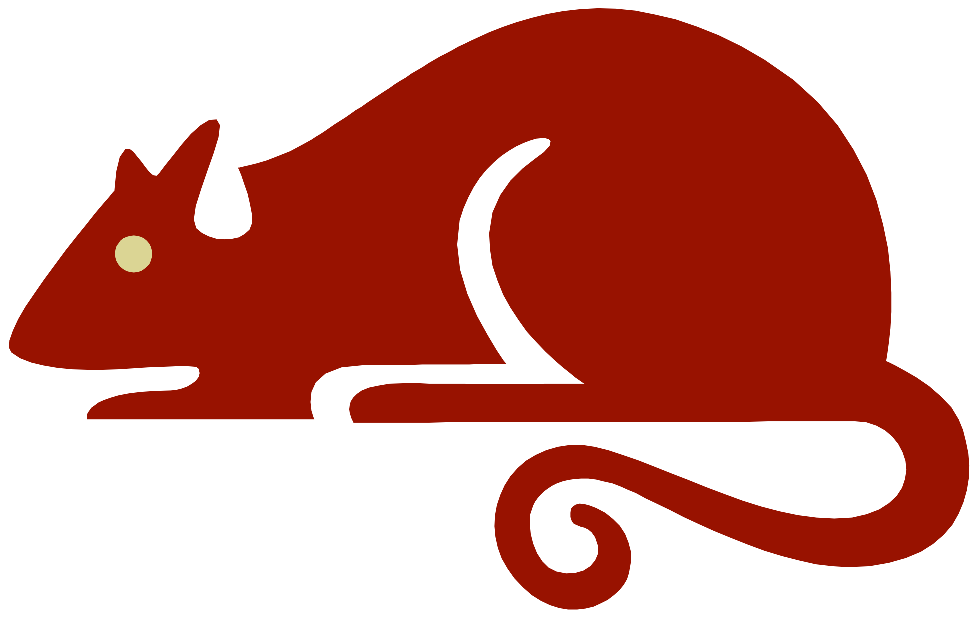
- User stories kept in one set of files: steps
- Separate set of files has Ruby code that tests steps: step definitions
- Step definitions are like method definitions, steps of scenarios are like method calls
- How match steps with step definitions?
- ***Regexes to match the English phrases in steps of scenarios to step definitions!***
  - `Given /^(?:|{}I )am on (.+)\$/`
  - `“I am on the Rotten Potatoes home page”`

Red-Yellow-Green Analysis

- Cucumber colors steps
- Green for passing
- Yellow for not yet implemented
- Red for failing
  (then following steps are Blue)
- Goal: Make all steps green for pass
  (Hence green vegetable for name of tool)

Running Cucumber and Introducing Capybara (web interaction access)





Capybara

- Need tool to act like user that pretends to be user follow scenarios of user story
- Capybara simulates browser
  - Can interact with app to receive pages
  - Parse the HTML
  - Submit forms as a user would
- Cannot handle JavaScript
  - Other tool (Webdriver) can handle JS, but it runs a lot slower, won’t need yet

Demo

- Add feature to cover existing functionality
  - Note: This example is doing it in wrong order – should write tests first
  - Just done for pedagogic reasons
- (Or can look at screencast:

  http://vimeo.com/34754747)

And in Conclusion

- Agile – prototypes, iterate with customer
- BDD – Design of app before implementation
- User Story – all stakeholders write what features want on 3x5 cards
- Tool: Cucumber – magically turns 3x5 card user stories into acceptance tests for app
  - Another suggestion: Minitest- good for those who are used to unit testing