Why do SW projects fail?

Failing projects:
- Don't do what customers want
- Are late
- Over budget
- Hard to maintain and evolve
- All of the above

*How does Agile try to avoid failure?*

Adapted from Armando Fox and David Patterson (Berkeley cs169) under CC-BY-SA-NC license.
What is the difference between validation and verification? Validation is did we build the right thing, verification is did we build it right.

Adapted from Armando Fox and David Patterson (Berkeley cs169) under CC-BY-SA-NC license.
Recall we covered TDD and deployment in previous classes.

Adapted from Armando Fox and David Patterson (Berkeley cs169) under CC-BY-SA-NC license.
Behavior-Driven Design (BDD)

- BDD is a conversation about app behavior *before and during development* to reduce miscommunication
  
  Recall “Individuals and interactions over processes and tools” in Agile manifesto

- Requirements written down as *user stories*
  
  Lightweight descriptions of how application is used

- BDD concentrates on *behavior vs. implementation* of application
  
  Test Driven Development (TDD) focuses on implementation

Concentrate on what the application does as opposed to how the application does it.

Adapted from Armando Fox and David Patterson (Berkeley cs169) under CC-BY-SA-NC license.
User Stories

• 1-3 sentences in everyday language
  Fits on an index card
  Written by or with the customer

• Often in “Connextra” format:
  Feature name
  As a [kind of stakeholder],
  I want to [some task],
  So that [some result or benefit].
  (all 3 phrases are needed, but can be in any order)

User stories will ultimately become work items in our product backlog (our team’s prioritized “to-do list”)

Why index cards?
Nonthreatening: All stakeholders participate in brainstorming
Easy to re-arrange: All stakeholders participate in prioritization
Helps keep stories short and low-cost to change during development

Adapted from Armando Fox and David Patterson (Berkeley cs169) under CC-BY-SA-NC license.
Write 1-2 user stories about existing Film Explorer features

Feature name

As a [kind of stakeholder],
I want to [some task],
So that [some result or benefit].

Film Explorer: https://stormy-plains-38289.herokuapp.com
S.M.A.R.T. user stories

- **Specific**
- **Measurable** (with specific, implies testable)
- **Achievable** (ideally implement in 1 iteration)
- **Relevant** (discover “business” value or kill)
- **Time-boxed** (know when to split/stop)

As a user, I want to click on a movie, so that I get more information.

As a user, I want to click on a movie to get plot details, so that I can see if I will like the movie.

Is our example user story SMART?
Definitely achievable and readily time-bound (we could set an amount of time to complete this feature). Measurable in the sense that we could imagine how to test this feature. But both specific and not, and not clearly relevant. How could we rewrite our story to be more specific and relevant?

As a user, I want to click on a movie to get a plot summary and other details, so that I can see if I will like the movie.

Adapted from Armando Fox and David Patterson (Berkeley cs169) under CC-BY-SA-NC license.
Student advice: Stories vs. Layers

• “Dividing work by stories helps all team members understand app & be more confident when changing it”
• “Tracker helped us prioritize features and estimate difficulty”
• “We divided by layers [front-end vs. back-end vs. JavaScript, etc.] and it was hard to coordinate getting features to work”
• “It was hard to estimate if work was divided fairly...not sure if our ability to estimate difficulty improved over time or not”

Adapted from Armando Fox and David Patterson (Berkeley cs169) under CC-BY-SA-NC license.
The customer wants “login with Facebook” integrated into their site. Nobody on your team is familiar with how to do this. You should:

A. Break up the story into very small user stories, to be on the safe side about how long each chunk takes.
B. Do a spike on Facebook integration, then propose one or more stories to implement.
C. Apologize to the customer that they can’t have this functionality

Answer: B

I think B is the best answer, but A could be arguable. C is not correct in this context, but that doesn’t mean you should never say no to the customer.

A spike is a product-testing method originating from Extreme Programming that uses the simplest possible program to explore potential solutions[1]. It is used to determine how much work will be required to solve or work around a software issue.

Adapted from Armando Fox and David Patterson (Berkeley cs169) under CC-BY-SA-NC license
Epics, User stories, Scenarios

**Epic**

As a <stakeholder>
I want to do <something>
so that <result or benefit>.

**User Stories**

*has many*

**Scenarios**

*has many*

Given <a context>,
when <an event happens>,
then <an outcome should occur>.

Not all work items may be user stories. Some work-items will be bugs. Sometimes a task is necessary but far removed from the user, e.g. read an arbitrary byte range from a local or remote file.
Epic > User Stories > Scenarios

User Stories are expanded into scenarios

Scenarios are formal but not code.

   Creates a “meeting point” between developers and customers.

With Gherkin syntax, we turn scenarios into automated acceptance tests:

   *Given* [a context],
   *When* [an event happens],
   *Then* [an outcome should occur]

1. `Given` steps represent state of world before event, the preconditions
2. `When` steps represent event, e.g. simulate user pushing a button
3. `Then` steps represent expected post-conditions, the test expectations
4. `And` and `But` extend any previous step.
Note that you don't have to use Cucumber to implement "Given-When-Then"-style tests. For simplicity, we will use code (and the Enzyme library) instead of trying to incorporate Cucumber (although you could if you wanted to do so in your project). Why? Cucumber brings non-trivial overhead (implementing functions for the different steps) in an already packed semester.
BDD is all about conversation

“Having conversations is more important than capturing conversations is more important than automating conversations”

Liz Keough
Which of the following statements most accurately describes the goals and use of BDD?

A. BDD is designed to support validation (build the right thing) and verification (build it right)
B. The best user stories include information about implementation choices
C. User stories have no counterpart in plan-and-document processes
D. Functionality should only be featured in a single user story for a single stakeholder

Answer: A

User stories are about behavior not implementation, and are similar to requirements in P&D. Multiple user stories may describe the same functionality, but from different stakeholders perspectives. Imagine a movie ticketing system that integrates with a social network. From the user's perspective "so that I can see movies with my friends", from a theater owner's perspective "so that I can sell more tickets". Knowing these perspectives can help us during the design and implementation of the feature.

Adapted from Armando Fox and David Patterson (Berkeley cs169) under CC-BY-SA-NC license
Building Successful UI

Our apps often faces users, thus needs UI
• How to get customer to participate in the UI design so they are happy with results?
  Avoid WISBNWIW* UI
  UI version of User Story index cards?
• How to get feedback cheaply?


Adapted from Armando Fox and David Patterson (Berkeley cs169) under CC-BY-SA-NC license.
What would a storyboard look like for Film Explorer?

Adapted from Armando Fox and David Patterson (Berkeley cs169) under CC-BY-SA-NC license.
Lo-Fi to React, HTML and CSS

Sketches and storyboards are tedious, but easier than code! And...

- Less intimidating to non-technical stakeholders
- More likely to suggest changes to UI if not code behind it
- More likely to focus on interaction rather than colors, fonts, ...

What you think is cool may not be what your users (customers) think is valuable.

Remember that what you think is cool, is not what your customer/user may think is valuable.

Adapted from Armando Fox and David Patterson (Berkeley cs169) under CC-BY-SA-NC license.
Student Advice: BDD & Lo-Fi Prototyping

• “Lo-fi and storyboards really helpful in working with customer”
• “Frequent customer feedback is essential”
• “What we thought would be cool is not what customer cared about”
• “We did hi-fi prototypes, and invested a lot of time only to realize customer didn’t like it”
• “Never realized how challenging to get from customer description to technical plan”

Adapted from Armando Fox and David Patterson (Berkeley cs169) under CC-BY-SA-NC license.