Why do SW projects fail?

Failing projects:

• Don't do what customers want
• Are late
• Over budget
• Hard to maintain and evolve
• All the above

How does Agile try to avoid failure?
Recall: Agile Lifecycle

• Work closely and continuously with stakeholders to develop requirements, tests
  Users, customers, developers, maintenance programmers, operators, project managers, ...

• Maintain a working prototype while deploying new features every 1-2 weeks

• Check in with stakeholders on what’s next, to validate you’re building the right thing (vs. verify)
*DD in our Agile iterations

- Talk to customer
- Behavior Driven Design: User stories...
- Test Driven Development (TDD)
- Deploy (to the cloud)
- Design patterns
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 the 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
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”)

Movie Details
As a user,
I want to click on a movie,
so that I can get more info
I.N.V.E.S.T. criteria

- Independent: Can be developed in any sequence
- Negotiable: Up to the team to decide how implement
- Valuable: Delivers some value to end users
- Estimable: We can predict how long it will take to implement
- Small: Implement in one sprint, ideally
- Testable: Clear acceptance criteria
Example user story for Film Explorer

As a user,
I want to see film details,
so that I can get more information.
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 Berkeley CS169
Epics, User stories, Scenarios

- **Epic**
  - has many

- **User Stories**
  - has many

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

- As a <stakeholder>
  I want to do <something>
  so that <result or benefit>.
Epic > User Stories > Scenarios

Epics provide a higher-level view of the project goals, e.g.,

*Search in a music streaming app*

- As a listener, I want to search from every page so that I can find music I am interested in
- As a listener, I want to search by lyrics, theme, etc. so that I can find songs when I can’t remember the title or artist
- As a listener, I want my search customized to my previous listening so that I get more relevant results
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]
Testing scenarios

Map to function

Given I open the url 'http://the/test/url'
When I click on the element 'Jurassic World'
Then I expect the element 'img[src="http://the/poster"]' is visible

Arguments extracted with RegEx
BDD is all about conversation

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

Liz Keough
Building a successful UI

Our apps often face users, thus need UI

• How to get customers to participate in the UI design so they are happy with results?
  
  Goal is to avoid WISBNWIW*

• How to get feedback cheaply?
  
  Is there a UI version of User Story index cards?

Lo-fi Storyboards

(Figure 4.4, Engineering Long Lasting Software by Armando Fox and David Patterson, Alpha edition, 2012.)
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.*
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 Berkeley CS169