Module 7
Behavior Driven Design, User Stories

CS W169A: Software Engineering

1 Overview

This discussion contains mostly open ended questions that are meant to get you thinking about the concepts behind Behavior Driven Design. The second half of questions are more application oriented, and you'll get some hands on practice writing different Cucumber scenarios. For some of these questions, there's no one right answer, so feel free to get creative!

2 Behavior Driven Design

- What is the difference between validation and verification? Provide an example of each. Which does BDD attempt to address

  Validation checks if you built the right thing, whereas verification ensures you built the thing right.

  Think of it this way: let's say you're working with a customer who wants you to build a webapp that, among other things, takes in a python program and outputs its java translation. You built a webapp that takes in a java program and outputs a python program instead. All of the unit tests that you create for your webapp verify that it works correctly. These tests might check that given an System.out.println() statement in Java, you correctly output the python print() equivalent.

  Given a product, does it function as intended? But notice that while it might function perfectly (e.g. be verified), the behavior of your program is not what the user wanted—if you built tests that validate the behavior of your webapp, such as with cucumber, these should be failing.

  BDD addresses validation.

- User stories should be SMART. After refreshing your memory on what SMART stories are, give an example of a non-SMART story and a SMART story describing an expected behavior of a user dashboard.

  SMART stories are Specific, Measurable, Achievable, Relevant, and Timeboxed.

  A non-SMART story might be: "The user dashboard has an easy-to-use user interface." (not specific; says nothing about how you'd measure it)

  Better: "On the user dashboard, the most common actions should be no more than three clicks away." (But the story doesn't say which actions are most common or how to determine this)

  Best: "On the user dashboard, the Update Profile and See My Friends actions should each be reachable by one click." This is specific enough to mock up a possible design and storyboard.

- Describe the relationship between Features, Scenarios, and Steps in Cucumber.

  Features, or user stories, describe how the application is expected to be used. A feature is an action the user wants to do, and is often given as "As a <kind of stakeholder>, in order to <accomplish some goal>, I want to <do some specific action>".

Scenarios describe ways in which a particular feature is used. For example, one scenario might describe the "happy path" when the user exercises the feature as designed or intended; "sad path" scenarios might describe what happens when something goes wrong.

Steps are the actions required to complete a given scenario, such as interacting with the page, verifying the presence or absence of certain elements on a page, and so on.

- Cucumber allows steps to begin with Given, When, Then, And, But, etc. Even though they are functionally identical, when should each be used?
  `Given` is used when setting up the state of an application.
  `When` is used to describe the key action(s) taken by the user.
  `Then` is used to describe the expected outcome.
  `And/But` are used to modify a preceding Given/When/Then.

- The Connextra Format is outlined before. Rewrite the same SMART user story you wrote for the previous question in the Connextra Format.
  - As a [kind of stakeholder]: user
  - So that [I can achieve some goal]: I can see frequently asked questions that might solve any confusion I have
  - I want to [do some task]: click on the FAQ link and be able to see FAQs.

- Name the three advantages of using Lo-Fi mockups.
  - Not intimidating to stakeholders
  - Customers more likely to suggest changes to UI (allows for more rapid prototyping and avoids having to repeat work)
  - Lowers customer's expectations of front-end MVP for that iteration

3 Cucumber Scenarios Analysis

The following user stories were converted to Cucumber scenarios. Name three things that could be improved about each user story and how this relates to the ease of implementation of the acceptance test the story represents. There are many right answers. The main idea is that they should center around SMART concepts.

3.1

```
Scenario: user must pay for purchase upon buy now
    Given I am logged in and visiting a product page on my e-commerce website,
    When I buy the product,
    Then I must pay for the product.
```

Three potential improvements are:

1. E-commerce websites usually offer multiple ways to buy a product. The test is a bit vague since it concerns a process (buying) and not a specific action (such as clicking proceed to checkout). Which method is this particular user story testing? Left as is, this acceptance test must check against every possible purchasing method in order to function as expected, which would likely make the test too big or complex. To fix this, we could specify how we buy the product by mentioning the specific action we want the user to take, such as clicking proceed to checkout.

2. The process of paying is similarly vague. Should we display a banner with information on where to send a check? Should we show them a payment screen where they can give us their information? The test would be easier to implement if we knew how exactly the user was supposed to pay.

3. Additionally, the ambiguity of the payment step doesn't really specify when the payment must take place. Should we prompt the user during the buying action? After? Should we ensure that the user has paid for the products within a certain payment time period? This test isn't really implementable without this information.

3.2

```
Scenario: bad actor cannot edit other user's to-do lists
    Given I am not logged in and on the to-do app home page,
    When I try to add an item to another user's to-do list,
    Then I should be shown a stern warning.
```

1. "Stern" is vague and subjective. What does it mean in the context of this story? What kind of information should be included in the warning? We should clarify this.

2. "Shown" is also ambiguous. Do we display a flash notice up top? Is it a javascript alert? Do we redirect them to a page? All of these paths would require their own unique implementation as a cucumber step definition. The client usually has an idea in mind for what they want the warning to look like, so get this information and instill it into the test so that you don't need to repeat work. Needing to repeat work could happen if you implemented the wrong behavior and were told to go back and do it again with some other interface instead. This is common in 169 projects (and in software development for a client in general) for teams whose stories aren't specific enough.

3. A common pitfall of "sad path" tests is not properly verifying the output of the intended bad action. Notice that this test only asserts that the bad actor is shown a warning and not that the to-do list in question is untouched. If your cucumber test forgets to address this aspect, it's very possible for a bug to slip in where users are actually able to modify other users' lists—so long as they're shown a warning for it.

4 User Stories to Cucumber Scenarios

Turn the following user stories into Cucumber scenarios:

4.1

As an e-commerce website user, so that I can aggregate all items I want to buy in one place to look at later, I can add items of interest to my cart.

Given that I am logged in as a user and on a product page,
When I click the "add to cart" button for the product,
Then my cart should contain that product.

4.2

As an e-commerce website user that qualifies for the special checkout experience, so that I can experience a faster, simpler, and more contextual purchase experience, I use the special checkout experience whenever I click buy now.

Given that I am logged in as a user that qualifies for the special checkout
experience, And that I am on a product page,
When I click the “buy now” button,
Then I should be shown the special checkout experience.

Tip: You can use “And” and “But” keywords in addition to “Given,” “When,” and “Then.”