Specification-based Testing

Meta Activity: Team Disruptions

Common disruptions to learning in teams include: talking about topics that are off-task, teammates answering questions on their own, entire teams working alone, limited or no communication between teammates, arguing or being disrespectful, rushing to complete the activity, not being an active teammate, not coming to a consensus about an answer, writing incomplete answers or explanations, ignoring ideas from one or more teammates.

Questions (10 min) Start time:

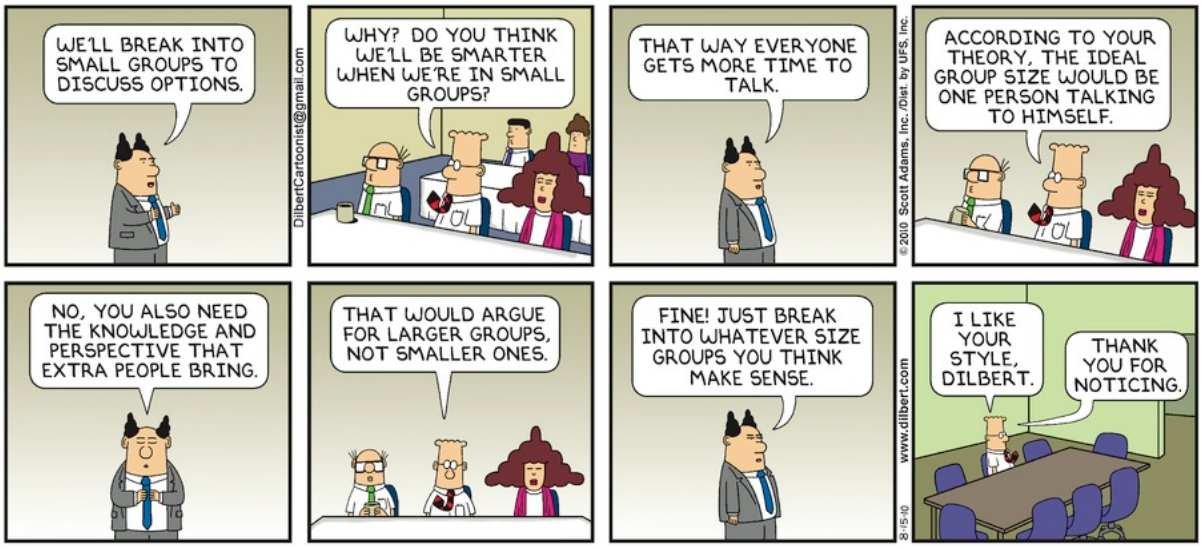
1. Pick four of the disruptions listed above. For each one, find something from the role cards that could help improve the team's success. Use a different role for each disruption.

a) Manager:

b) Presenter:

c) Recorder:

d) Reflector:



Dilbert by Scott Adams. © Andrews McMeel Syndication. http://dilbert.com/strip/2010-08-15

Model 1 Specification-based Testing (Black-box testing)

Last week we started categorizing tests. This week we expand on that categorizing and generalize the methodologies it encompassed. Two main methodologies in black-box testing (or specification-based testing) are ***equivalence partitioning*** and ***boundary value analysis***. Equivalence partitioning involves:

a) Looking at each input individually and asking: "What are the possible classes (or partitions) of inputs I can provide?"

b) Combining these partitions when dealing with multiple inputs.

c) Looking at the different classes of output: "Does it return arrays? Can it return an empty array? Can it return nulls?"

Boundary value analysis involves identifying ***on points***, ***off points***, ***in points*** and ***out points*** at the boundary between the different partitions and making sure that your test suite covers all those cases.

substringsBetween

Method `substringsBetween()` searches a string for substrings delimited by a start and end tag, returning all matching substrings in an array.

- str—The string containing the substrings. Null returns null; an empty string returns another empty string.
- open—The string identifying the start of the substring. An empty string returns null.
- close—The string identifying the end of the substring. An empty string returns null.

The method returns a string array of substrings, or null if there is no match.

2. What are the different categories and the partitions in each category for this problem?

3. The chapter claims that an upper bound for the number of tests can be obtained by combining the input partitions alone (4 x 4 x 4 x 5 = 320). Why were the partitions for the output not included?

4. In your own words, explain why and how we can prune the original set of 320 possible tests to get to the 21 described in the chapter?

5. What are some interesting boundaries between the partitions identified earlier? Provide concrete ***on point***, ***off point***, ***in point*** and ***out point*** tests for each.

addDigits

The method receives two numbers, left and right (each represented as a list of digits), adds them, and returns the result as a list of digits. Each element in the left and right lists of digits should be a number from [0–9]. An `IllegalArgumentException` is thrown if this pre-condition does not hold.

- left—A list containing the left number. Null returns null; empty means 0.
- right—A list containing the right number. Null returns null; empty means 0.

The program returns the sum of left and right as a list of digits.

6. What are the different categories and the partitions in each category for this problem?

7. What are the most interesting combinations between these partitions for our test suite?

8. What are some interesting boundaries between the partitions identified earlier? Provide concrete ***on point***, ***off point***, ***in point*** and ***out point*** tests for each.

subtractDigits

Let's consider the complement operation of add: subtract. This method is identical to the add method, except that it subtracts digits instead of adding them.

9. What are the different categories and the partitions in each category for this problem?

10. What are the most interesting combinations between these partitions for our test suite?

11. What are some interesting boundaries between the partitions for this problem? Provide concrete ***on point***, ***off point***, ***in point*** and ***out point*** tests for each.