Unit 2 Notes

- Contrast requirement-based versus scenario-based testing
- Apply the equivalence partitioning testing technique
- Apply cause-effect testing technique
- Test asynchronous events
- Apply state-based testing technique
- Describe model-based testing strategies
- Apply the boundary value testing technique

Input Sampling Techniques Part 1

- Contrast requirement-based versus scenario-based testing
    - Find in slides

Scenario based testing: Review of reading

- Use cases.
- Can have a normal scenario, and multiple different scenarios.
- Each scenario will have a test cases, maybe more than one
- Characteristics of good scenarios
    - 5 key characteristics. It is a story that is motivating, credible, complex, and easy to evaluate
- Twelve ways to Create Good Scenarios
    - Designing scenario tests is much like doing a requirements analysis, but is not requirements analysis. They rely on similar information but use it differently.
        - The requirements analyst tries to foster agreement about the system to be built. The tester exploits disagreements to predict problems with the system
        - The tester doesn't have to reach conclusions or make recommendations about how the product should work. Her task is to expose credible concerns to the stakeholders
        - The tester doesn't have to make the product design tradeoffs. She exposes the consequences of those tradeoffs, especially unanticipated or more serious consequences than expected.
        - The tester doesn't have to respect prior agreements. (Caution: testers who belabor the wrong issues lose credibility.)
        - The scenario tester's work need not be exhaustive, just useful.
    - Consider disfavored users: how do they want to abuse your system?
        - As Gause and Weinberg point out, some users are disfavored. For example, consider an accounting system and an embezzling employee. This user's interest is to get more money. His objective is to use this system to steal the money. This is disfavored: the system should make this harder for the disfavored user rather than easier
    - List "system events." How does the system handle them?
        - An event is any occurrence that the system is designed to respond to. In Mastering the Requirements Process, Robertson and Robertson write

about business events, events that have meaning to the business, such as placing an order for a book or applying for an insurance policy. As another example, in a real-time system, anything that generates an interrupt is an event. For any event, you'd like to understand its purpose, what the system is supposed to do with it, business rules associated with it, and so on. Robertson and Robertson make several suggestions for finding out this kind of information.

- Interview users about famous challenges and failures of the old system.
  - Meet with users (and other stakeholders) individually and in groups. Ask them to describe the basic transactions they're involved with. Get them to draw diagrams and explain how things work. As they warm up, encourage them to tell you the system's funny stories, the crazy things people tried to do with the system. If you're building a replacement system, learn what happened with the predecessor. Along with the funny stories, collect stories of annoying failures and strange things people tried that the system couldn't handle gracefully. Later, you can sort out how "strange" or "crazy" these attempted uses of the system were. What you're fishing for are special cases that had memorable results but were probably not considered credible enough to mention to the requirements analyst. Hans Buwalda talks about these types of interviews (www.stickyminds.com).
- Try converting real-life data from a competing or predecessor application.
  - Running existing data (your data or data from customers) through your new system is a time-honored technique.
  - A benefit of this approach is that the data include special cases, allowances for exceptional events, and other oddities that develop over a few years of use and abuse of a system.
  - A big risk of this approach is that output can look plausible but be wrong. Unless you check the results very carefully, the test will expose bugs that you simply don't notice. According to Glen Myers, The Art of Software Testing, 35% of the bugs that IBM found in the field had been exposed by tests but not recognized as bugs by the testers. Many of them came from this type of testing.

Input Sampling Techniques Part 2

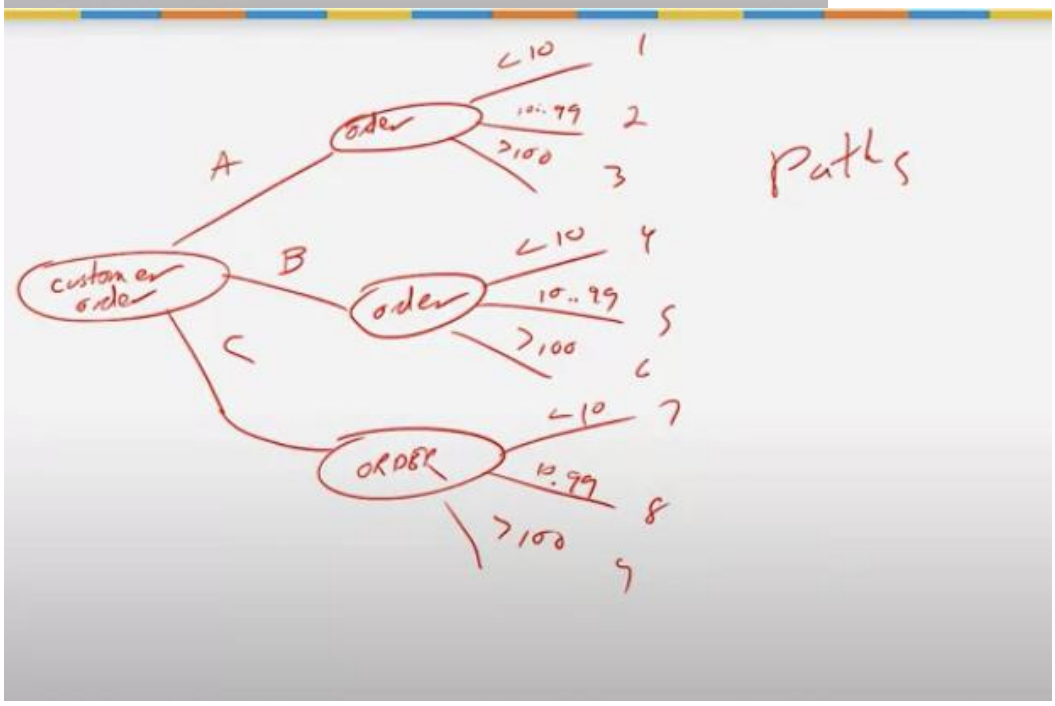
- Equivalence Partitioning Testing
  - See slides
- Equivalence Partitioning Testing Knowledge check
  - True or False? Equivalence partitioning is a good technique to utilize when there are multiple independent inputs.
    - True. Equivalence partitioning must be applied with independent inputs. In the lecture example of abs(x), it is necessary to test a negative value, 0, and a positive value which are all independent inputs.
  - What is NOT an equivalence partitioning step?
    - Write test cases covering as many of the uncovered invalid equivalence partitions as possible. There must only be one test case that covers each uncovered invalid equivalence partition at a time. Since equivalence

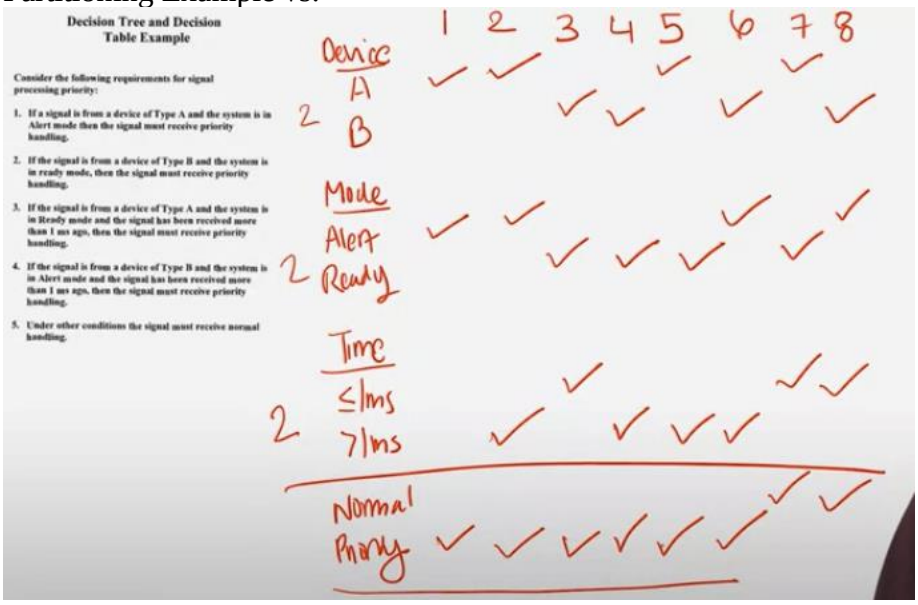
partitioning is used during black box testing, if there are multiple invalid partitions tested, it is not known which invalid partition caused the error.

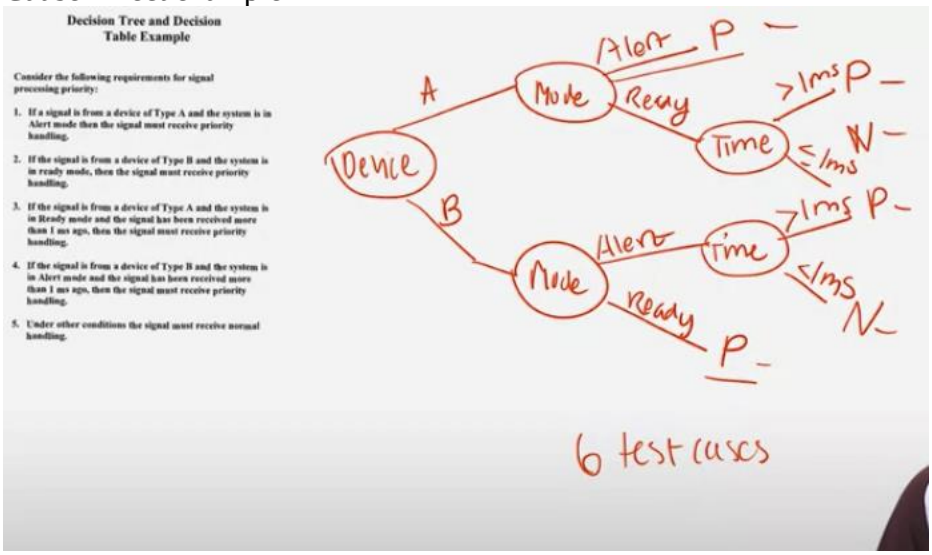
- What are a good set of equivalence partitions for a password testing program where a password must be between 6-8 characters?
  - 6-8
  - < 6 characters
  - > 8 characters
  - These equivalence partitions test all the possible inputs there could be for a password: a valid password between 6-8 characters, an invalid password with < 6 characters, and an invalid password with > 8 characters.

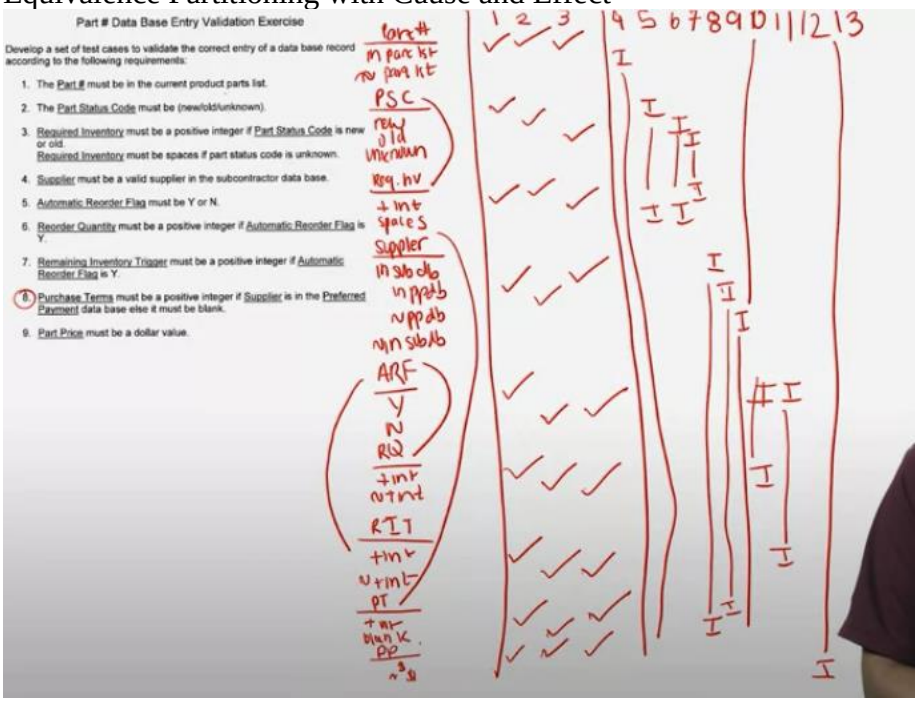
- Boundary Value Testing
  - See slides
- Boundary Value Testing Knowledge check
  - A valid student ID can only contain digits between and including 2 and 7. What are the correct boundary values that need to be tested?
    - 1,2,7,8. 1 and 2 test valid and invalid values for the first boundary of a digit greater than or including 2. 7 and 8 test valid and invalid values for the second boundary.
  - True or False? Boundary value analysis requires selecting test cases that are only on or above the edges of the input and output equivalence partitions.
    - False. BV analysis requires selecting test cases that are on, above, and below the edges of the partitions.
- BV EP Testing: Review of Reading
  - Important points
    - Uni dimensional equivalence partitioning
    - multi-dimensional ep
    - EPxCE (Cause Effect)

- Case Effect Analysis
  - See slides
  - 

  - 

- Partitioning Example vs.
  - 

  - Cause Effect example
  - 

- Equivalence Partitioning with Cause and Effect
  - 

  -

Cause Effect Analysis Knowledge Check

- Given 2 input variables, age and height, an output of vitamins, and a function that computes the number of vitamins a person should take based on age and height, which method(s) can be used to test this example?
  - Cause Effect Analysis. Cause effect analysis can be used when the inputs are dependent on each other. Equivalence partitioning is only used when inputs are independent of each other.
- What is one way to reduce the number of test cases in a cause-and-effect decision table?
  - Make assumptions about how the partitions are related. From the example in lecture, making assumptions about how error handling between a customer and order happens allows us to reduce the number of test cases needed.

Input Sampling Techniques Part 3

Testing Asynchronous Events

- See Slides

Testing Asynchronous Events Knowledge Check

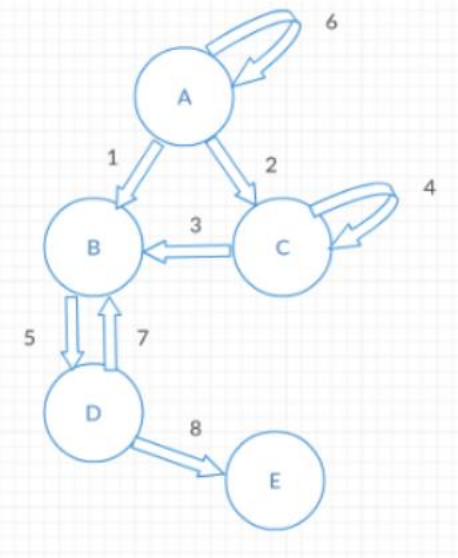
- Does creating a timeline to test synchronous events help with verification or validation?
  - Validation. Adding in events in a timeline helps identify missing requirements, which is necessary for validation.
- True or false? A timeline corresponds to a set of use cases mapped against time.
  - False. Each use case requires its own timeline.

State Based Testing

- See slides

State Based Testing Knowledge check

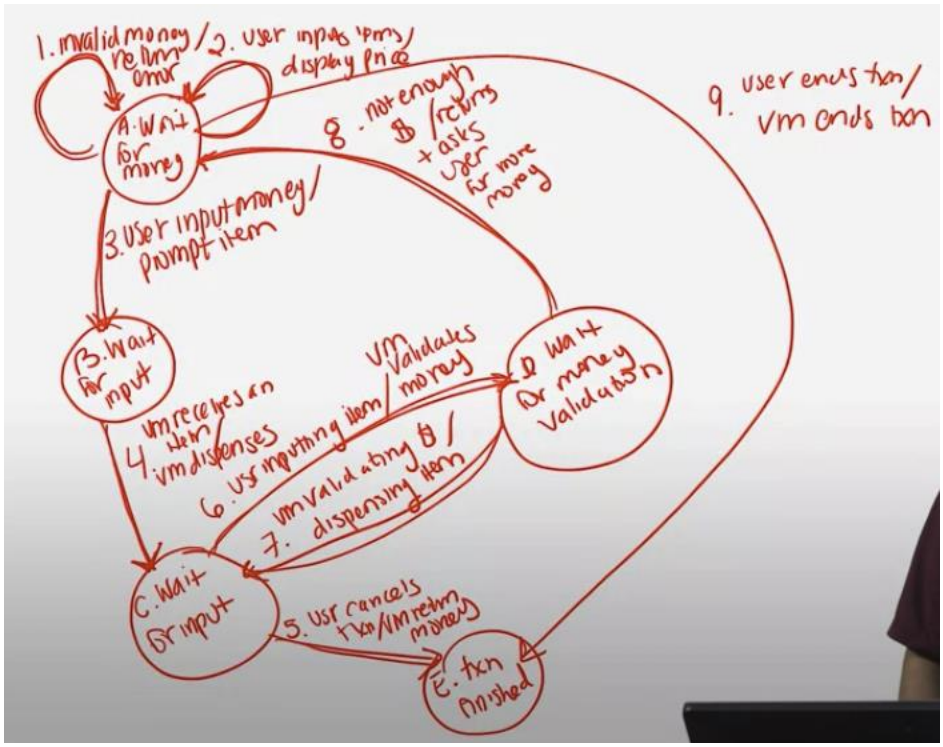
- True or False? The basic strategy of testing a state machine is to visit each state and test each transition.
    - True. Testing each state/event combination provides a minimal state machine cover.
- Given the state diagram below with start state A and end State E, which of the following is a test one would derive from the testing tree.



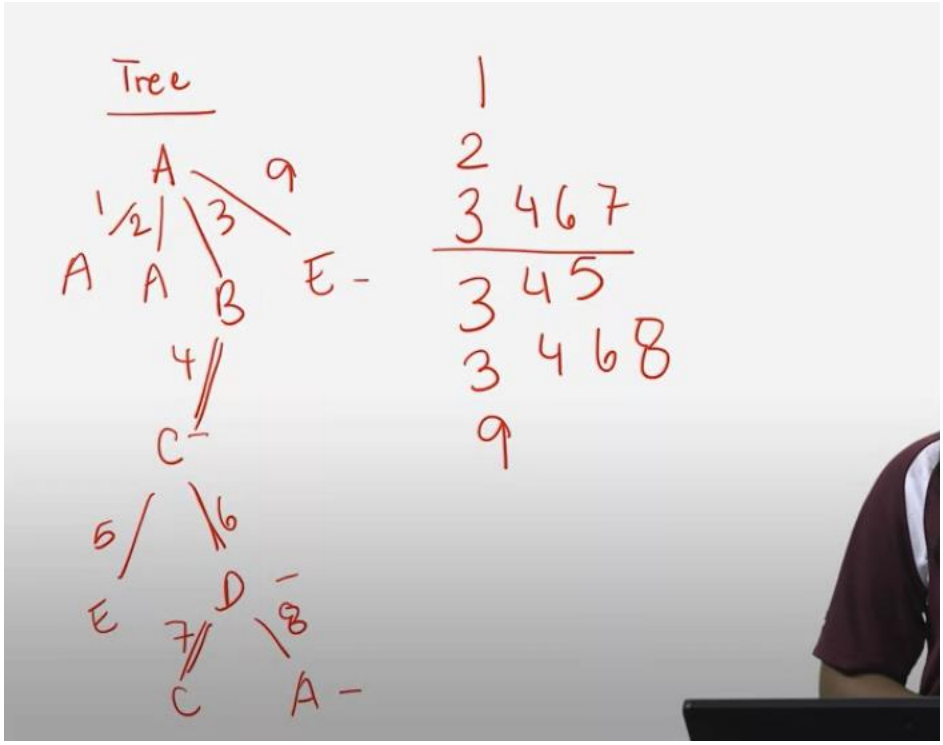
    - 1,5,8. The starting transition is 1, and then through transition 5 and 8, we reach the end state.

State Based Testing Problem example

State machine

- 

Tree derived from state machine

- 

Model Based Testing

Model Based Testing

- See Slides

Model Based Testing Knowledge Check

- What is NOT considered to be an advantage of model based testing?
  - Modeling Requires manual test case generation. Modeling gives potential for automated test generation, and does not require manual test case generation.
- What is the correct order of steps for model based testing?
  - Create a system model, select test generation criteria, generate tests, execute tests.