SCENARIO TESTING

Scenario testing

- Tag line
  - Tell a persuasive story
- Fundamental question or goal
  - Challenging cases that reflect real use.
- Paradigmatic case(s)
  - Appraise product against business rules, customer data, competitors' output
  - Life history testing (Hans Buwalda's "soap opera testing.")
  - Use cases are a simpler form, often derived from product capabilities and user model rather than from naturalistic observation of systems of this kind.

Scenario testing

The ideal scenario has several characteristics:

- The test is *based on a story* about how the program is used, including information about the motivations of the people involved.
- The story is *credible*. It not only *could* happen in the real world; stakeholders would believe that something like it probably *will* happen.
- The story involves a *complex use* of the program *or a complex environment or a complex set of data*.
- The test results are *easy to evaluate*. This is valuable for all tests, but is especially important for scenarios because they are complex.

Why use scenario tests?

- Learn the product
- Connect testing to documented requirements
- Expose failures to deliver desired benefits
- Explore expert use of the program
- Make a bug report more motivating
- Bring requirements-related issues to the surface, which might involve reopening old requirements discussions (with new data) or surfacing not-yet-identified requirements.

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.

Sixteen ways to create good scenarios

1. Write life histories for objects in the system. How was the object created, what happens to it, how is it used or modified, what does it interact with, when is it destroyed or discarded?
2. List possible users, analyze their interests and objectives.
3. Consider disfavored users: how do they want to abuse your system?
4. List system events. How does the system handle them?
5. List special events. What accommodations does the system make for these?
6. List benefits and create end-to-end tasks to check them.
7. Look at the specific transactions that people try to complete, such as opening a bank account or sending a message. What are all the steps, data items, outputs, displays, etc.?
8. What forms do the users work with? Work with them (read, write, modify, etc.)
9. Interview users about famous challenges and failures of the old system.
10. Work alongside users to see how they work and what they do.
11. Read about what systems like this are supposed to do. Play with competing systems.
12. Study complaints about the predecessor to this system or its competitors.
13. Create a mock business. Treat it as real and process its data.
14. Try converting real-life data from a competing or predecessor application.
15. Look at the output that competing applications can create. How would you create these reports / objects / whatever in your application?
16. Look for sequences: People (or the system) typically do task X in an order. What are the most common orders (sequences) of subtasks in achieving X?

Soap operas

- Build a scenario based on real-life experience. This means client / customer experience.
- Exaggerate each aspect of it:
  - example, for each variable, substitute a more extreme value
  - example, if a scenario can include a repeating element, repeat it lots of times
  - make the environment less hospitable to the case (increase or decrease memory, printer resolution, video resolution, etc.)
- Create a real-life story that combines all of the elements into a test case narrative.

Examples of story lines when used for testing

- Pension fund
  - William starts as a metal worker for Industrial Entropy Incorporated in 1955. During his career he becomes ill, works part time, marries, divorces, marries again, gets 3 children, one of which dies, then his wife dies and he marries again and gets 2 more children….
- World wide transaction system for an international bank
  - A fish trade company in Japan makes a payment to a vendor on Iceland. It should have been a payment in Icelandic Kronur, but it was done in Yen instead. The error is discovered after 9 days and the payment is revised and corrected, however, the interest calculation (value dating)…

From a talk by Hans Buwalda

Soap operas (in testing) are <u>not</u> necessarily:

"Extreme"

Far fetched

Long and elaborate

Pieces of art and creativity

From a talk by Hans Buwalda

“Killer Soaps”

- More specifically aimed at finding hidden problems
- Run when everything else has passed
- One option: put a killer soap at the end of a normal cluster
- Ask the “specialists” for input

From a talk by Hans Buwalda

Risks of scenario testing

- Other approaches are better for testing early, unstable code.
  - A scenario is complex, involving many features. If the first feature is broken, the rest of the test can't be run. Once that feature is fixed, the next broken feature blocks the test.
  - Test each feature in isolation before testing scenarios, to efficiently expose problems as soon as they appear.
- Scenario tests are not designed for coverage of the program.
  - It takes exceptional care to cover all features or requirements in a set of scenario tests. Statement coverage simply isn't achieved this way.
- Reusing scenarios may lack power and be inefficient
  - Documenting and reusing scenarios seems efficient because it takes work to create a good scenario.
  - Scenarios often expose design errors but we soon learn what a test teaches about the design.
  - Scenarios expose coding errors because they combine many features and much data. To cover more combinations, we need new tests.
  - Do regression testing with single-feature tests or unit tests, not scenarios.