Designing for Testability

Characteristics of testable software

Controllable — components have inputs that can be manipulated to fully exercise the component's capabilities.

Observable — the effect of a component can be captured and used to infer correctness.

Isolatable — each component can be tested (mostly) independently.

Scenario

I’m the founder of a local startup making IoT coffee makers. We’ve completed our first prototype using *your* money and now you want to make sure you made a good investment. You’ll test our prototype to make sure it conforms to the specifications agreed upon.

Specifications

The specifications are in the README.md file of the ***brewer*** repository.

Clone the repo: https://github.com/nickbradley/brewer

Activity 1 (10 min)

Collaborate with your neighbour(s) to come up with the 5 highest-value tests that you want to run to make sure the you are satisfied that the prototype correctly implements the specification.

Activity 2 (10 min)

Now, try and fill in the body of the tests.

Discussion questions:

- Were you able to implement reliable tests for each of the test cases?
- What features of the current implementation made testing challenging?
- How would you change to the implementation to make testing easier?

Characteristics of testable software

Controllable — components have inputs that can be manipulated to fully exercise the component's capabilities.

Observable — the effect of a component can be captured and used to infer correctness.

Isolatable — each component can be tested (mostly) independently.

Activity 3 - Live refactoring

Refactor the implementation to improve testability.