Moldable scenario editor

Master thesis final presentation, Bern University F2020
Ivan Kravchenko
Supervised by Nitish Patkar, Andrei Chiş and Nataliia Stulova
Problem

How can business idea be reflected in documentation and code?

• Code is often poorly described
• *Idea to implementation* flow spans through many tools and instruments
• Ubiquitous language is hard to develop and manage for a variety of business requirements
Agile idea

Iterative business process
Small iterations
Many interested process participants
Testing helps to assert that implementation is not outdated and still conforms to the requirements.
Tests need an environment

Tests on stage/test

Production

Difficult to manage and repeat
Tests as specifications

Executable specification

Executable anytime, easily repeatable

Code
BDD

![Diagram of BDD, DDD, and TDD relationships]
Research question 1

What are the limitations of the BDD tools regarding software behavior specification?

We analyzed 14 BDD tools:
Cucumber, FitNesse, Jbehave, Concordion, SpecFlow, Spock, Rspec, Mspec, LightBDD, ScalaTest, Specs2, Jgiven, phpspec, Gauge
Main observations

1. Input and Output are strictly defined

2. Two ways of writing specifications:
   - plain text language and 'glue code'
   - code with annotations

3. Test output is pass or fail; best case – customized report (colors, charts, coverage)

4. Providing objects as an input in a scenario is not very common

5. Code generation is not possible

6. No tools provide a graphical representation of specifications
Research question 2

How can we closely couple specification and implementation and what advantages does it give over existing tools?

Unite documentation and implementation
User-friendly scenario editing
Code generation
Demo
Results

✓ Connect requirements and behavior
✓ Allow users experiment with properties
✓ Generate code based on user input
✓ Scenario editor and creator
Future work

Reduce test development time by writing less ‘glue’ code

Think of unified way of defining ubiquitous language

Composable parameterized examples

Visual UI improvements

Default object views for widget elements
Summary

Main observations

1. Input and Output are strictly defined
2. Two ways of writing specifications:
   a. Plain text language and glue code
   b. code with annotations
3. Test output is pass or fail; best case - customized report (colors, charts, coverage)
4. Providing objects as an input in a scenario is not very common
5. Code generation is not possible
6. No tools provide a graphical representation of specifications