Software Testing and Quality Engineering

Maurício Aniche and Arie van Deursen

TU Delft

Kazimir Malevich, Eidght Red Rectangles, Wikipedia

Why do we test?

1. To make *informed decisions* about expected quality when releasing software

2. To guide *requirements elicitation*, by identifying (simple to understand) *execution scenarios.*

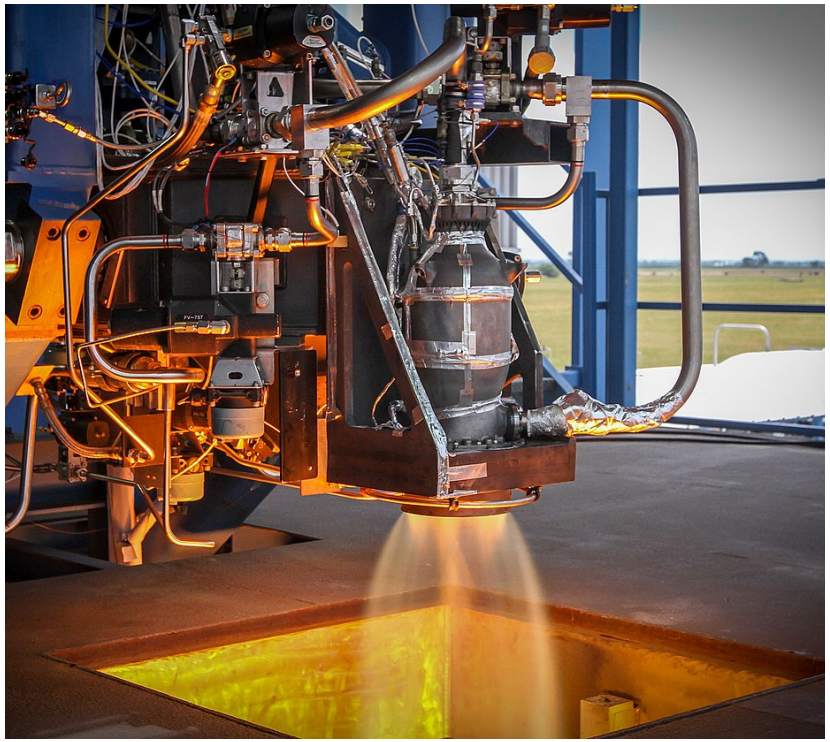
3. To guide the *design* of the software that we create.

How we test our software: *Test Execution*

- In modern software development, we *release often*

- Releasing often implies testing often:
  - Automate test execution as much as possible

- Build a testing system aimed at
  - exercising the system under test
  - and verifying the observed behavior



https://www.spacex.com

How do we test our software: *Test Design*

- Decide which (of the infinitely many possible) test cases to create
  - Maximize *information gain*
  - Minimize *cost*

- A test strategy:
  - A systematic approach to arrive at test cases
  - Targeting specific types of faults
  - Until a given adequacy criterion is achieved

- *Test design begins at the start of your project*

What do we test?

Test Levels

- Different levels of granularity

- Unit testing
- Integration testing
- System testing
- …

Test Types

- Different objectives

- Functionality (old / new)
- Security
- Performance
- …

Learning Objectives

- **Knowledge level:**
  - Essential test methods, tools, techniques, ...
- **Application leve**l:
  - Actually use selected test techniques
- **Evaluation level**:
  - Decide what’s useful in your project
  - Criticize, analyze, investigate, reflect, research, innovate, ...

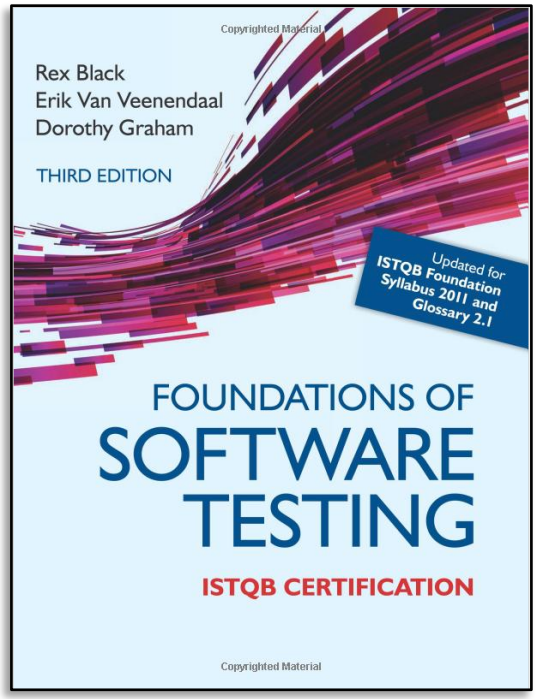
Reliable Knowledge in Software Testing?

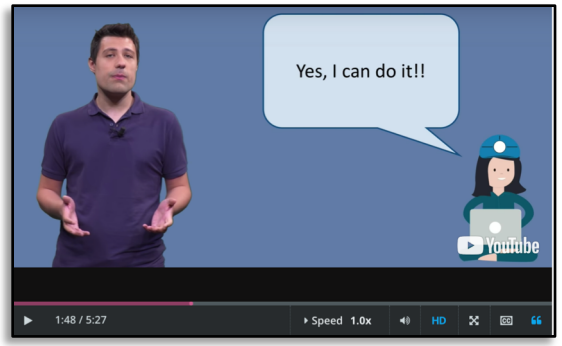
- Software testing is all about making *trade-offs*
  - Becomes easier with experience!

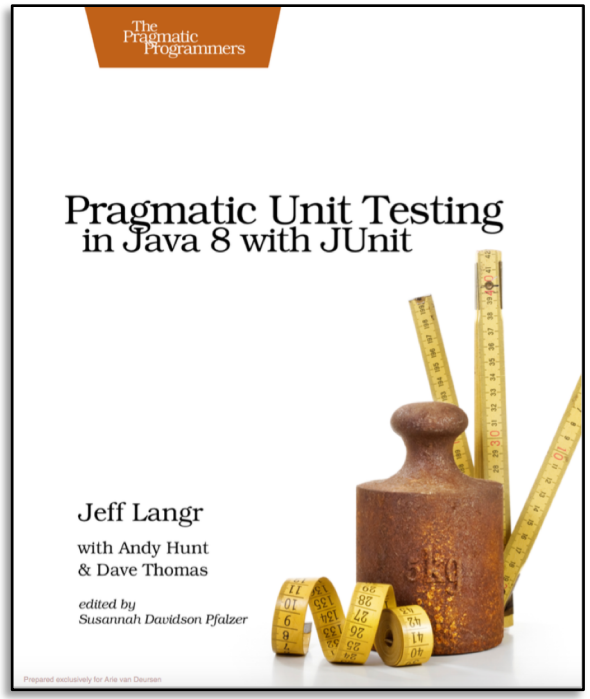
- Strategies, patterns, and processes are *codified experience*
  - You will need to know them!

- Our body of knowledge grows as reflective engineers / researchers:
  - Codify their knowledge and pass it on
  - Analyze successes and failures and report on those
  - Propose, implement, and evaluate novel testing strategies

Course Material







15 Lectures!

23/4: Introduction

26/4: Foundations

30/4: Functional testing

01/5: Model-based testing

03/5: Structural testing

07/5: Exploratory testing
(Jan Jaap Cannegieter)

10/5: Testability, mock objects

13/5: Software security 1
(Sicco Verwer)

15/5: Static/dynamic analysis
(Azqa Nadeem)

17/5: Test code quality

28/5: Web testing
(Frank Mulder)

07/6: Design by contract

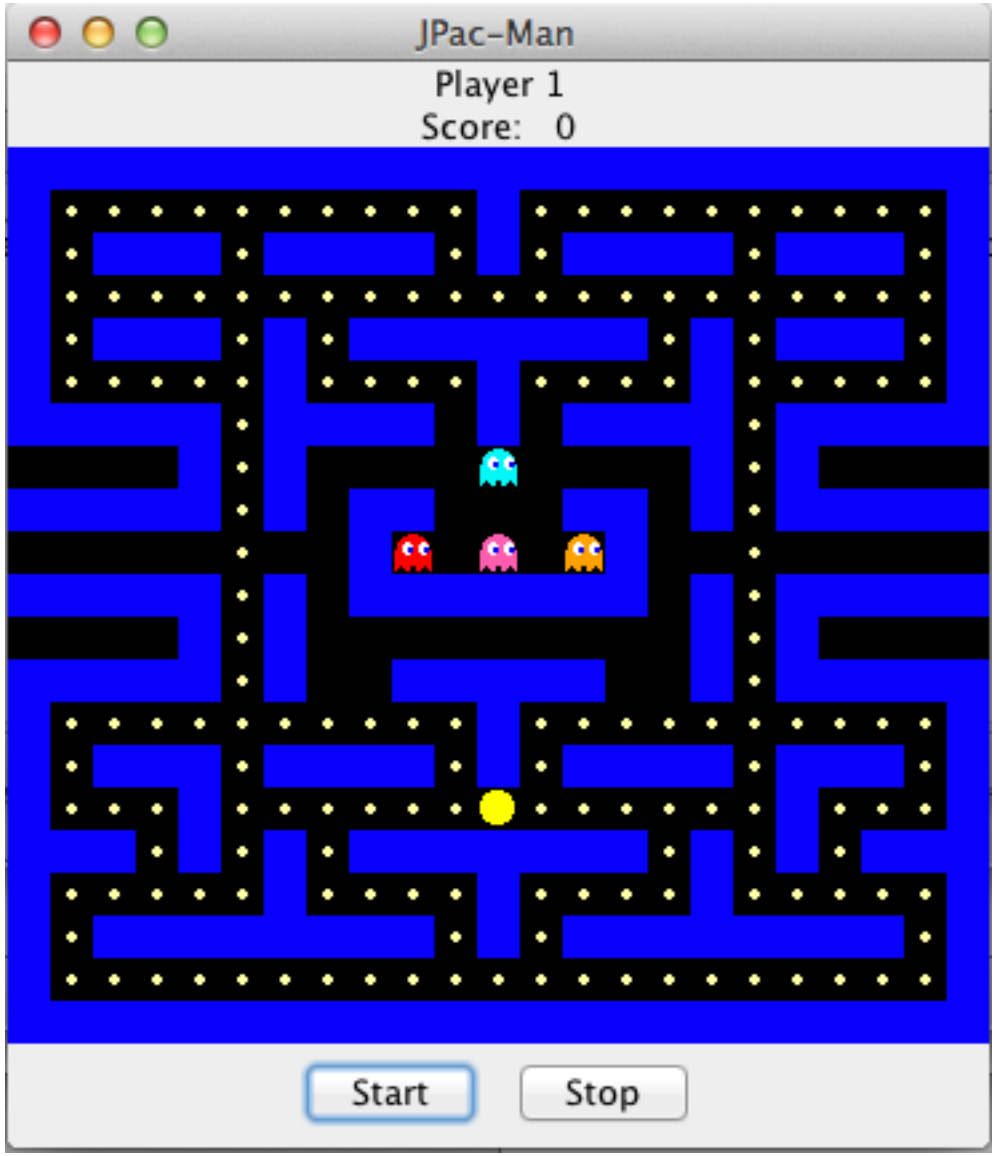
11/6: Search-based testing
(Annibale Panichella)

14/6: Breaking changes in OS
(Tim van der Lippe)

20/6: Testing at Spring
(Stéphane Nicoll)

Learning in the Labwork

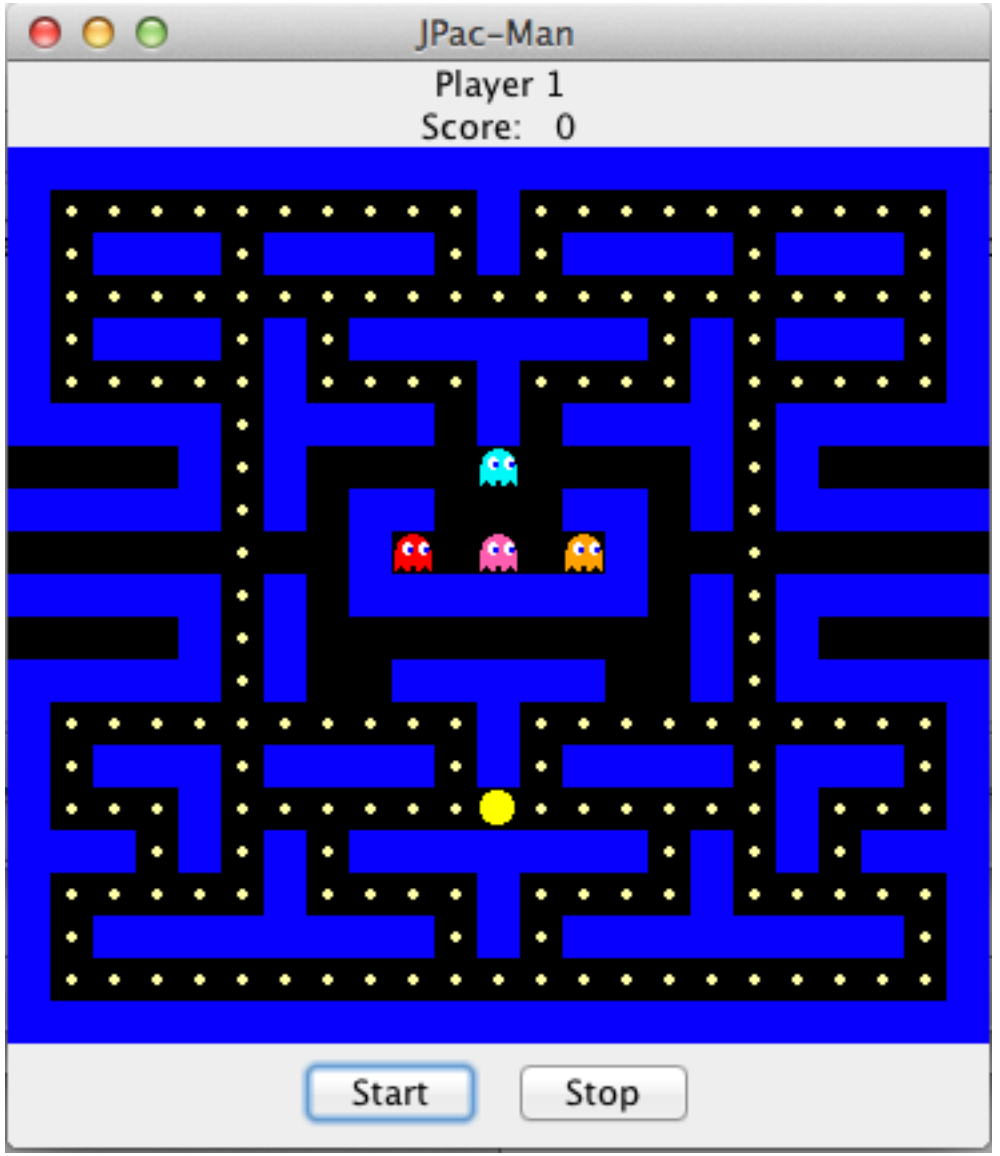
- End-to-end testing
- Structural testing
- Functional testing
- State-based testing
- Decision-table based testing
- Boundary-value testing
- CORRECT
- AAA



github.com/SERG-Delft/jpacman

Labwork Tools Used

- JUnit 5
- AssertJ
- Mockito
- Java 9
- IntelliJ
- Git
- Gradle
- GitLab CI



github.com/SERG-Delft/jpacman

Multiple Choice:
People *hate* git because:

A. git's commands are inconsistent and confusing.

B. rebasing and push forcing are overly complicated operations

C. handling merge conflicts can be a nightmare.

D. All of the above

E. None of the above.

Multiple Choice:
People *love* git because git

A. supports understanding a change in its historical context

B. supports isolating changes and moving them around

C. supports identifying and discussing changes

D. scales to 1000s of distributed developers

E. All of the above