CS 4350: Fundamentals of Software Engineering
CS 5500: Foundations of Software Engineering

Lesson 5.2 Test-Driven Development

Jon Bell, John Boyland, Mitch Wand

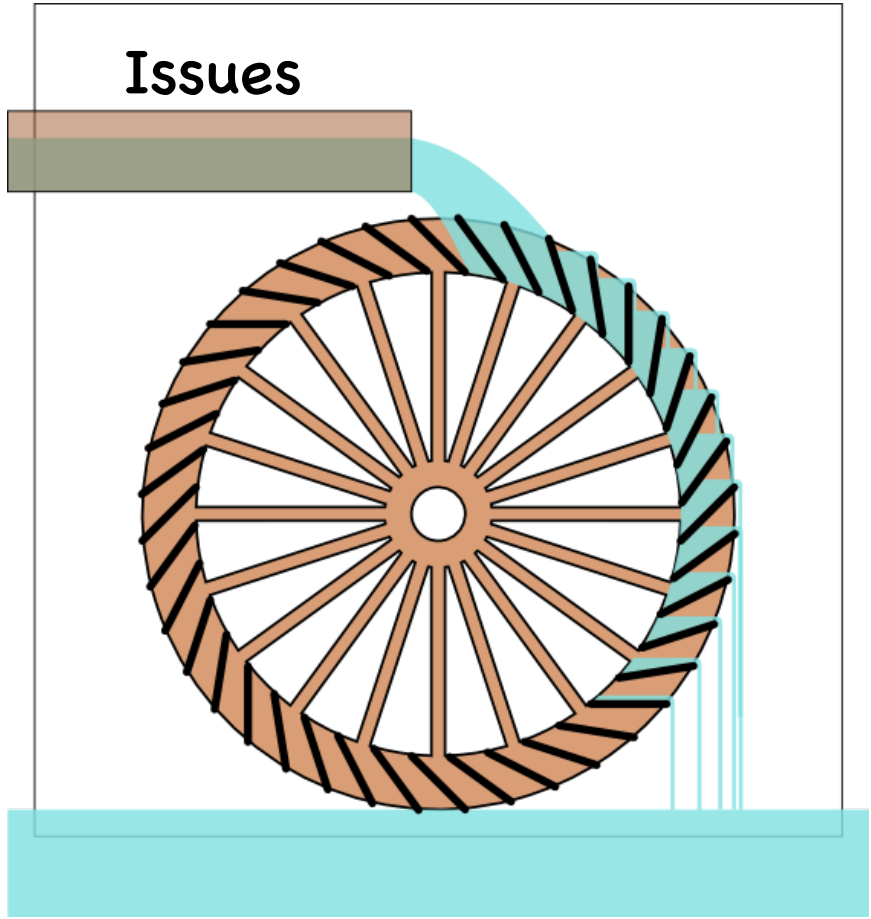
Khoury College of Computer Sciences

Learning Objectives for this Lesson

- By the end of this lesson, you should be able to:
  - Define “Test-Driven Development”;
  - Contrast two different phases for programming in TDD;
  - Outline the strengths and weaknesses of TDD.

Test-Driven Development (1)

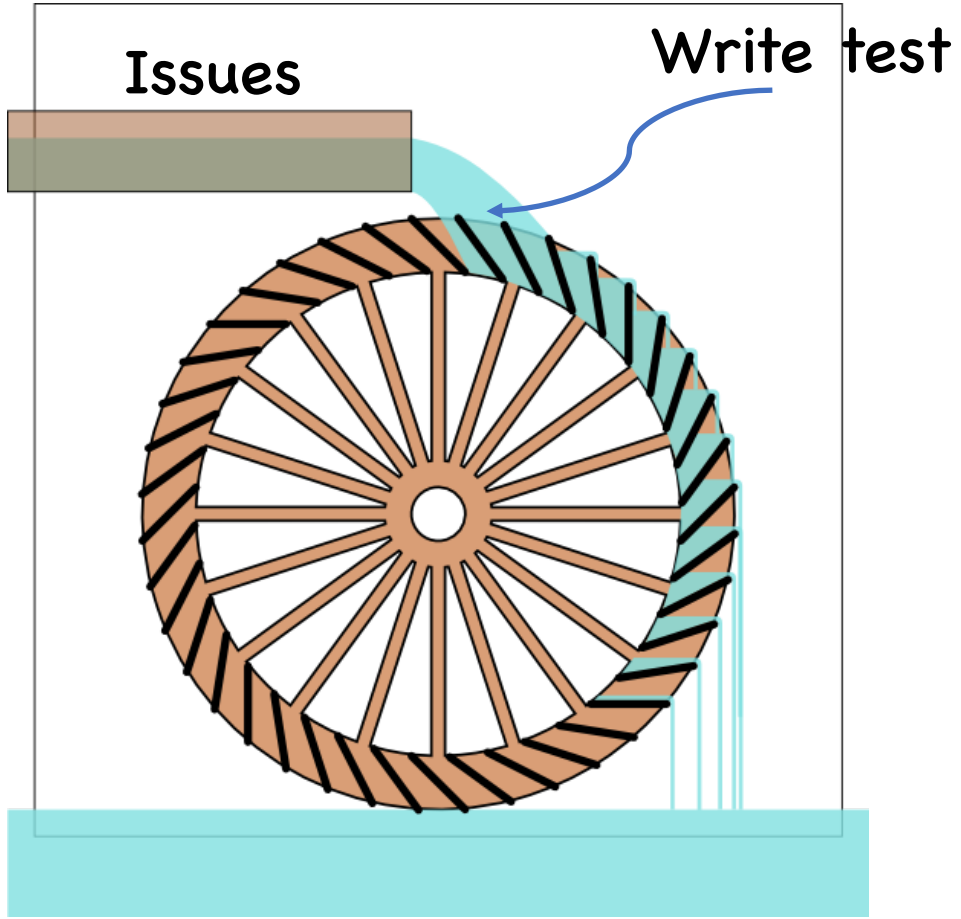
- From outside, the development is driven by “issues”:
  - New feature requests;
  - Enhancement requests;
  - Bug reports;
  - Internal feature requests.
- Issues are the “water” in our “TDD = water wheel” metaphor.



Test-Driven Development (2)

- The first task is to write a test.
  - The test should fail.
  - A bug report is not actionable until we have replicated it.
  - A feature request is not actionable until we know what how it should work.

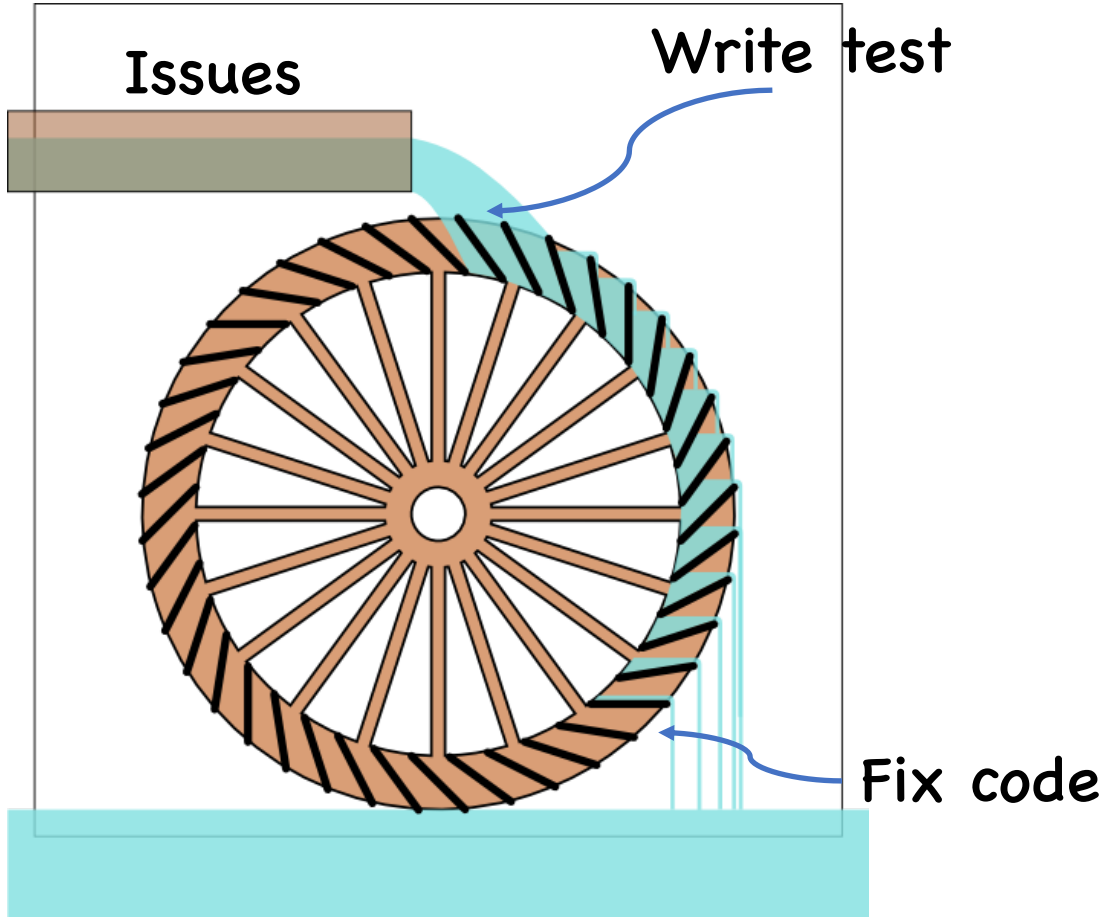
- Tests are the “buckets” in our metaphor.



Test-Driven Development (3)

- Then we fix the code:
  - Change code until test passes;
  - All previous tests must pass too (no regression!);
  - No redesigns; goal is to fix as quickly as possible.

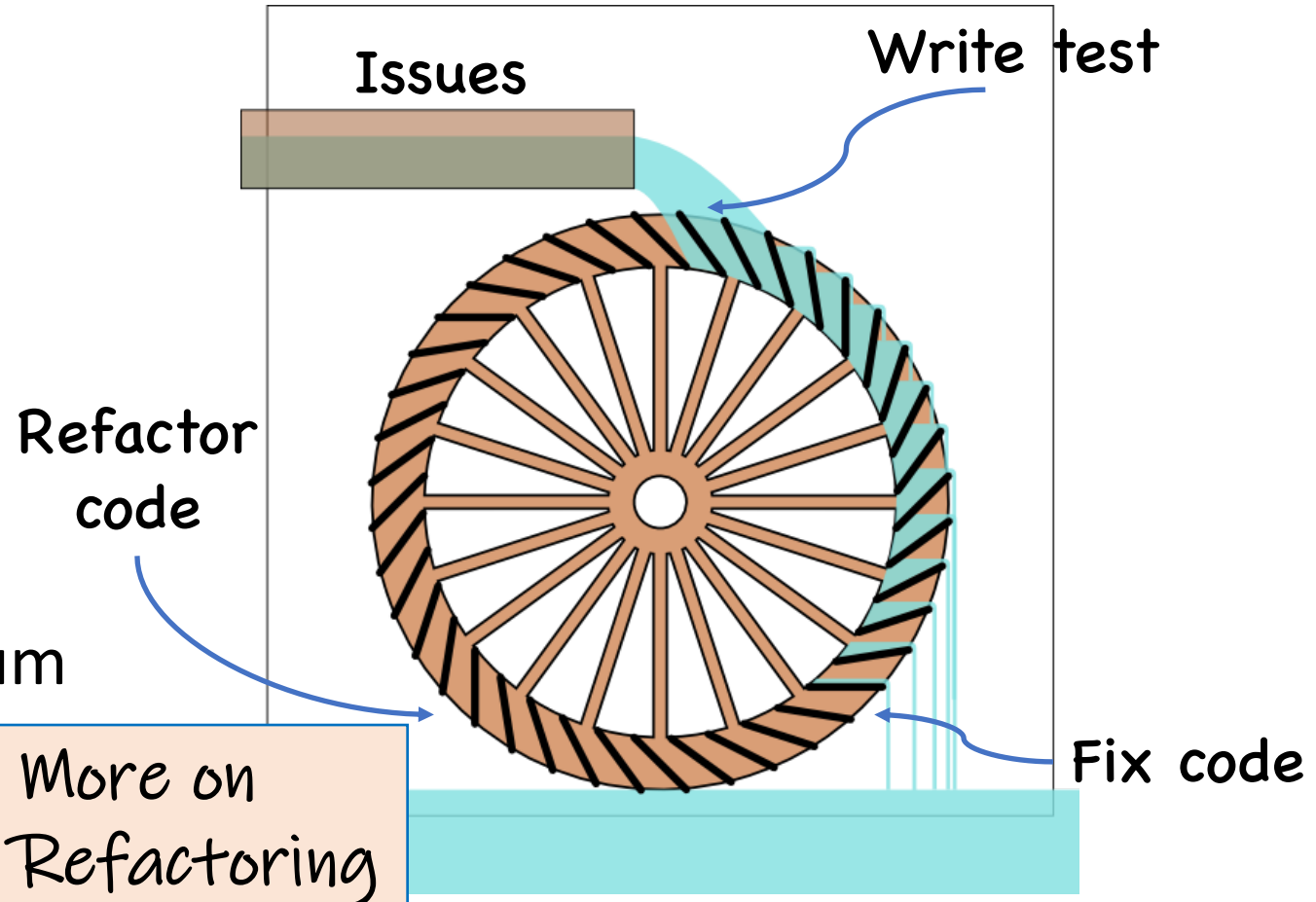
- Coding turns the wheel until the "water" is gone (issue is fixed).



Test-Driven Development (4)

- Clean up code:
  - At leisure, “refactor” code;
  - Not driven by issues;
  - No (visible) behavior changes;
  - All tests must still pass;
  - Improve maintainability.

- Refactoring borrows momentum to turn the wheel without the action of “water.”



Caveats & Qualifications

- Typically, a new feature will require multiple tests
- The “fix” should not just be the minimum to pass the test(s)
  - The programmer should keep in mind the spec/requirements.
  - But the fix should be the simplest possible that addresses the issue.
- Tests are run frequently and thus must be fast and deterministic.
- Occasionally, the tests may need to be fixed as well.

Strengths

- Goals are concrete and actionable.
- We revisit requirements frequently:
  - We make sure we are building the right product;
  - Mistakes are fixed earlier.
- Separate refactoring stage means code hygiene is not forgotten.
- Test portfolio gives confidence in maintenance.

Weaknesses

- Often the same person writes the test and implements the code being tested
  - Blind spots: programmer may overlook something;
  - Gentleness: programmer may avoid “hard” tests.
- Tests can add to maintenance problems
  - Slow, flaky or brittle tests can slow down ”wheel” (both fixing code and refactoring)
- As defined, TDD is perhaps overly strict. (Discuss!)

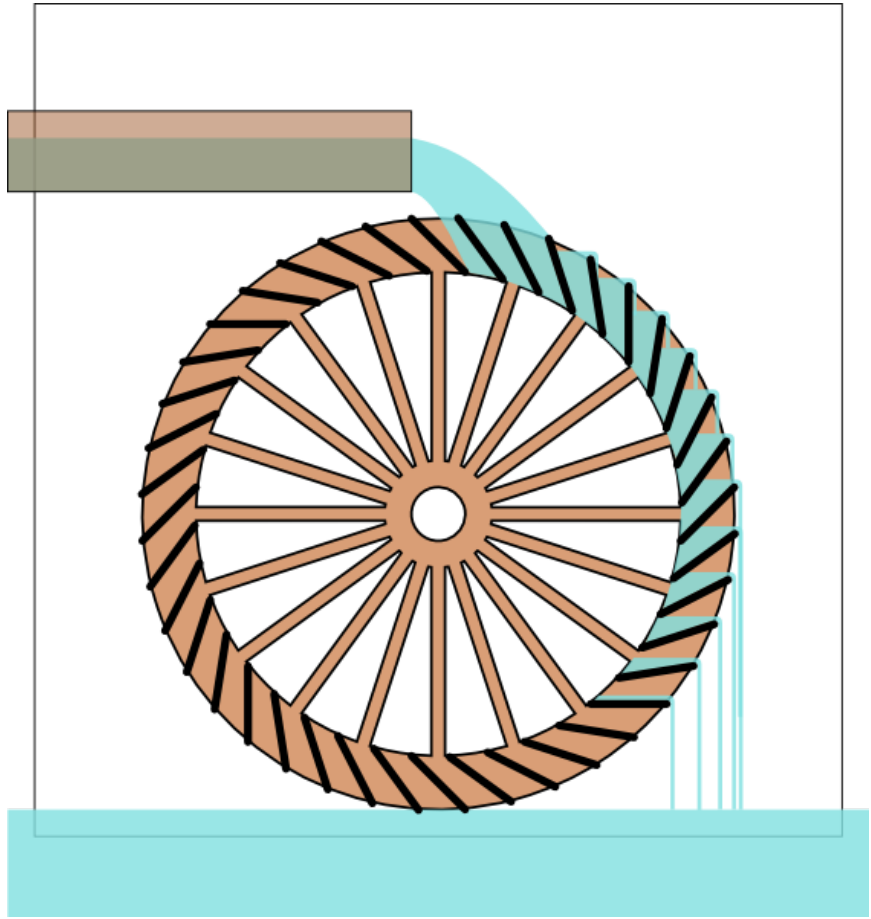
Flaky and Brittle:
See next Lesson!

Variants

- Acceptance Test Driven Development (ATDD)
  - Write “system” tests to express user requirements.
  - These tests may be “large” and/or “slow”.
  - Some may not be automatable.
- Behavior Test Driven Development (BTDD)
  - Uses structured natural language to describe user stories with desired behavior.
  - Also “system” tests.

Review

- Now that you've studied this lesson, you should be able to:
  - Define “Test-Driven Development”;
  - Contrast two different phases for programming in TDD;
  - Outline the strengths and weaknesses of TDD.



Looking Forward

- In our next lesson, we'll learn about how to evaluate tests. What makes a test suite good?