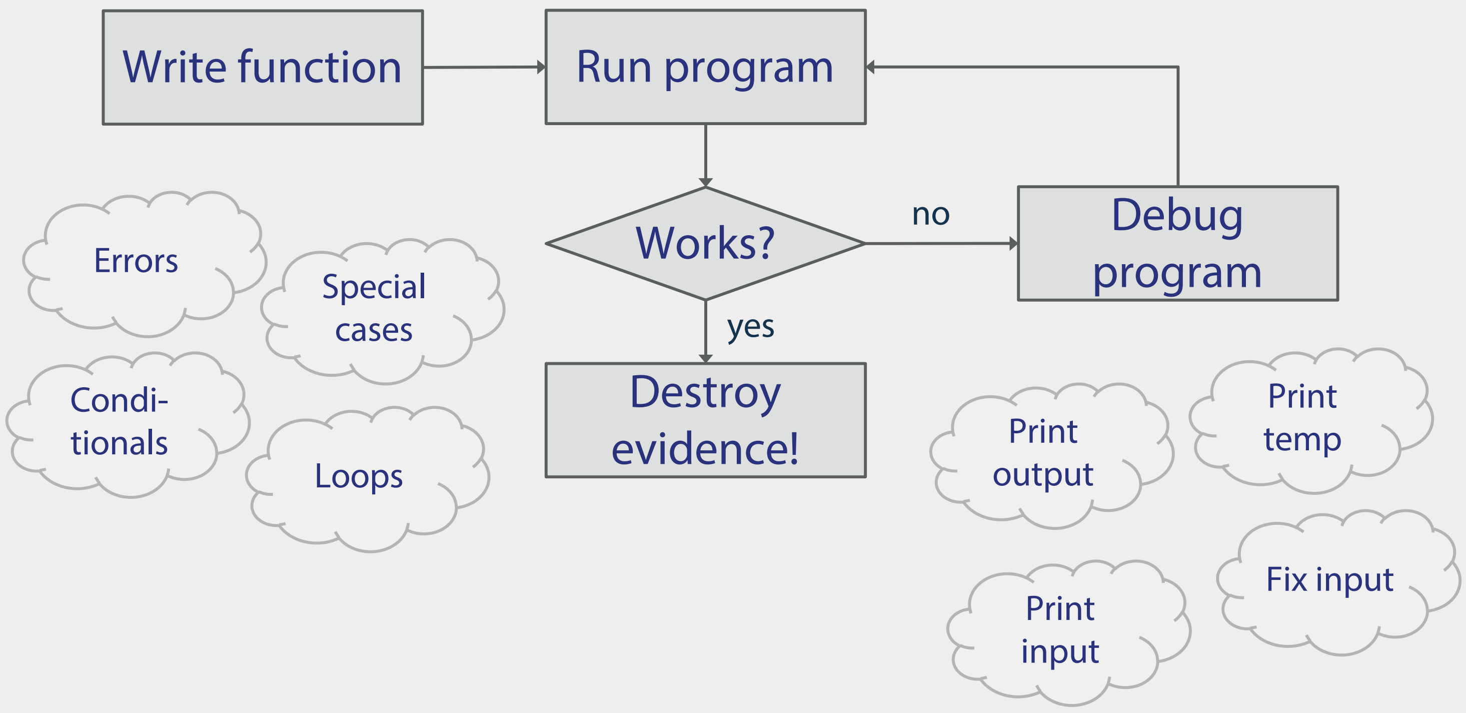
Test-Driven Development

Dr. Michael König, Blue Yonder GmbH

michael.koenig@blue-yonder.com
@turbodbc

Confessions of a Physicist



BlueYonder
Best decisions, delivered daily

Bugs!

They are everywhere!

Would you kindly…

- Change!
  - Bug
  - Feature
  - Performance
- Mommy!
  - Unclear behavior
  - New & old bugs
  - Manual testing
  - Not my code!
  - Wait, that was me?!

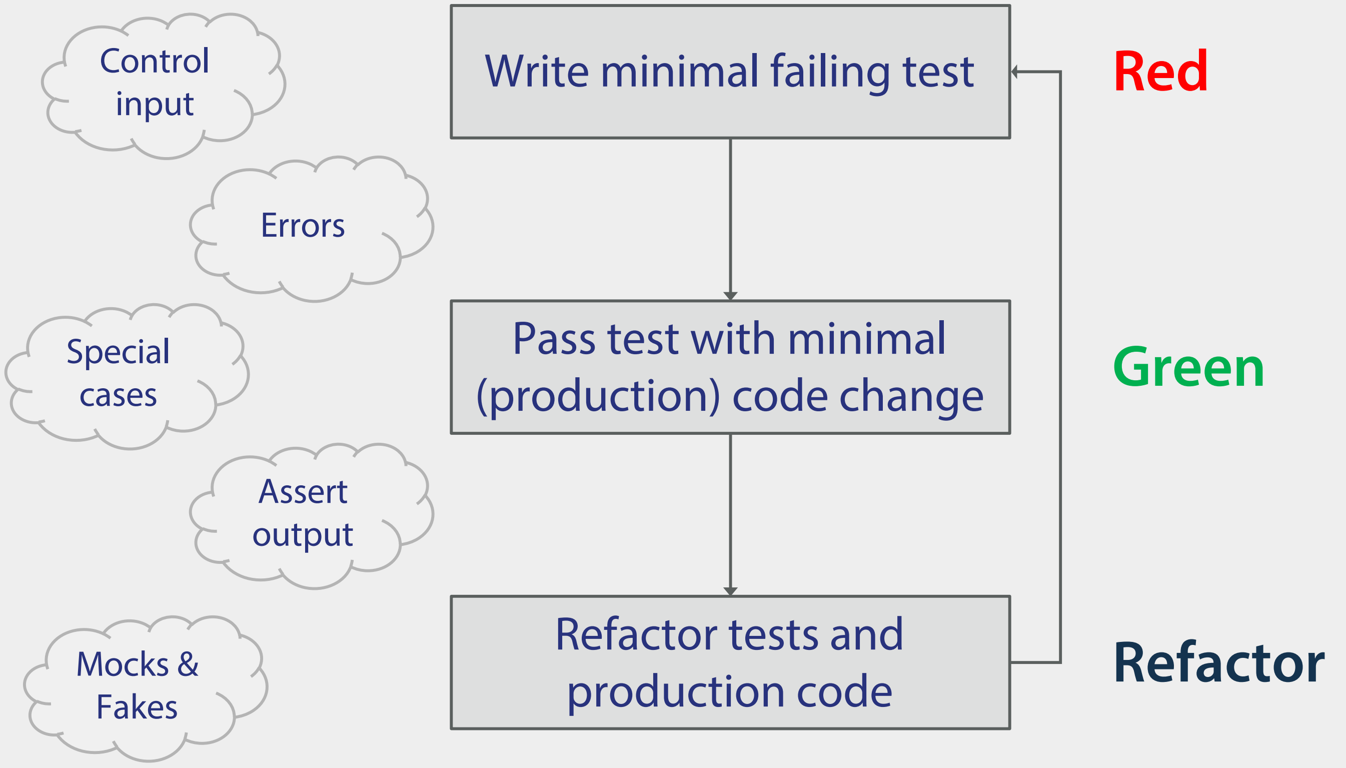
“Never change a running system”

“Fix nothing which ain’t broken”

BlueYonder
Best decisions, delivered daily

This far,
no further!

Test-driven development


Control input
Errors
Special cases
Assert output
Mocks & Fakes
Write minimal failing test
Pass test with minimal (production) code change
Refactor tests and production code
Red
Green
Refactor

BlueYonder
Best decisions, delivered daily

Red: Write minimal failing test

- Minimal!
    - Prevents complexity
- Execute all tests
    - Prevents slow tests
- Assert new test fails
    - Prevents inactive tests
    - Prevents bugs in tests
    - Prevents complexity

Minimal means:

- Missing import
- Missing class
- Missing function
- One assertion a time
- Simple to complex
    - Error cases first
    - Corner cases next
    - General behavior last

BlueYonder
Best decisions, delivered daily

Green: Pass test with minimal change

- Minimal!
  - Prevents missing tests
- Execute all tests
  - Prevents slow tests
- Assert all test succeed
  - Prevents bugs in code
  - Prevents bugs in tests

Minimal means:

- Add file stub
- Add class stub
- Add function stub
- Unconditionally raise
- Hard-coded results
- Correctly sized results
- Defer conditionals
- Defer loops

Refactor: Clean up test/production code

- Remove superseded tests
    - Better signal/noise ratio
- Clean code principles
    - Reduce complexity
- Execute all tests
    - Prevents slow tests
    - Prevents refactoring bugs
    - Prevents brittle tests

Principles

- DRY
- SRP
- SLA
- KISS
- POLA
- LoD

BlueYonder
Best decisions, delivered daily

Hands-on

Roman numerals

- Task description & C++ quickstart at
  `https://github.com/github.com/CppUserGroupKarlsruhe/2017_02_TDD.git`

- Virtual environment recommended

```
> git clone https://github.com/CppUserGroupKarlsruhe/
                                    2017_02_TDD.git
> cd 2017_02_TDD
> git checkout cpp
> cmake

> make && ctest --verbose
```

BlueYonder
Best decisions, delivered daily

Emotions

Is TDD that *painfully* slow?

- Babysteps… really?
  - Not necessarily
  - Write failing test
  - Write obvious implementation

- TDD lets you work as fast as you can

"The best race drivers know when to *brake*"

BlueYonder
Best decisions, delivered daily

TDD boosts your code

- Impact on code
  - Modular design
  - Cleaner code
  - Less bugs

- Impact on tests
  - Full automation
  - 100% coverage
  - Executable specs

TDD boosts your work life

- Steady sense of progress
- Ease of mind
- Courage

“We ain’t got time for tests!”

Speed vs. Quality

- Speed generates opportunities

- Quality
  - Builds trust
  - Keeps customers
  - Scales
  - Fosters sustainability

“I can meet any deadline if it needn’t work”

BlueYonder
Best decisions, delivered daily

What could possibly go wrong?

Tests to avoid

- Can’t say no
- Overly complex tests
- Parrots
- Riddles
- Nitroglycerine
- Mocking hell
- Refactoring clamps

Countermeasures

- Split
- Helpers for setup / assertions
- Hand-picked examples
- Express intent in names
- Eliminate *all* randomness
- Prefer fakes/stubs over mocks

- Improve production code design

- Last resort: Drop

BlueYonder
Best decisions, delivered daily

Boundaries of TDD

Is TDD perfect for anything?

- Prototype code
  - Quickly moving target without perspective
- Performance optimizations
  - Non-functional, without prior expectation
- Concurrent programming
  - *Hard* to control
- Declarative code
  - *How* more complex than *what*

BlueYonder
Best decisions, delivered daily

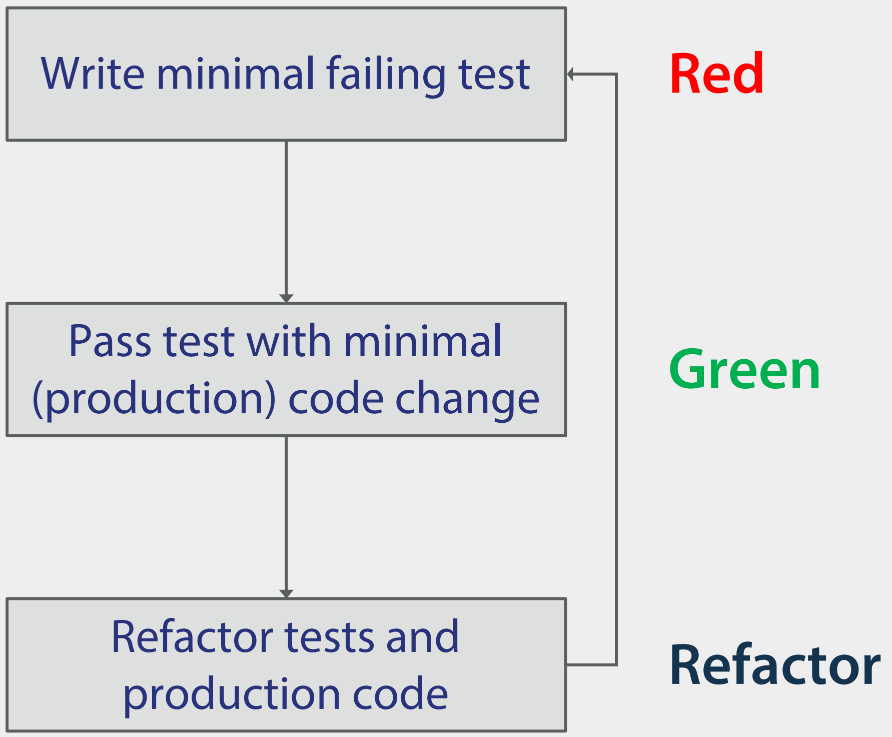
Summary

TDD: One tool in your belt

- Great for
  - Functional correctness
  - Black/white situations
  - Production code
  - Single-threaded code
  - Non-declarative code

- Don’t be dogmatic about it

Test-driven development



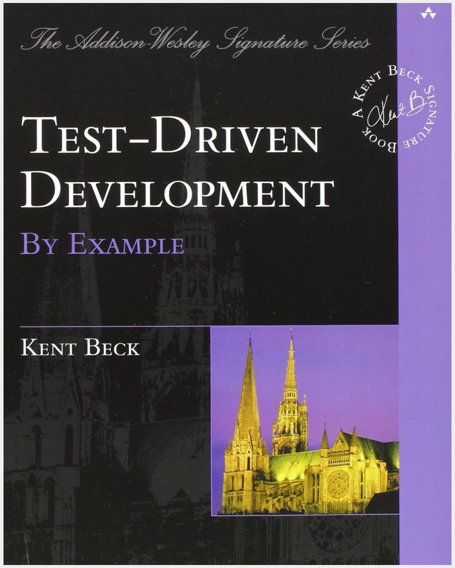
Write minimal failing test
Red
Pass test with minimal (production) code change
Green
Refactor tests and production code
Refactor

BlueYonder
Best decisions, delivered daily

Further material

- Kent Beck: *Test-driven development by example*

- Code Katas:
  - Poker hand classification
  - Hangman game
  - Roman to Arabic numerals
  - …


The Addison-Wesley Signature Series
A Kent Beck Signature Book
Test-Driven Development
By Example
Kent Beck