Oak Ridge National Laboratory

A Scientific Approach to Developing Scientific Software

Gregory R Watson

CW3S19

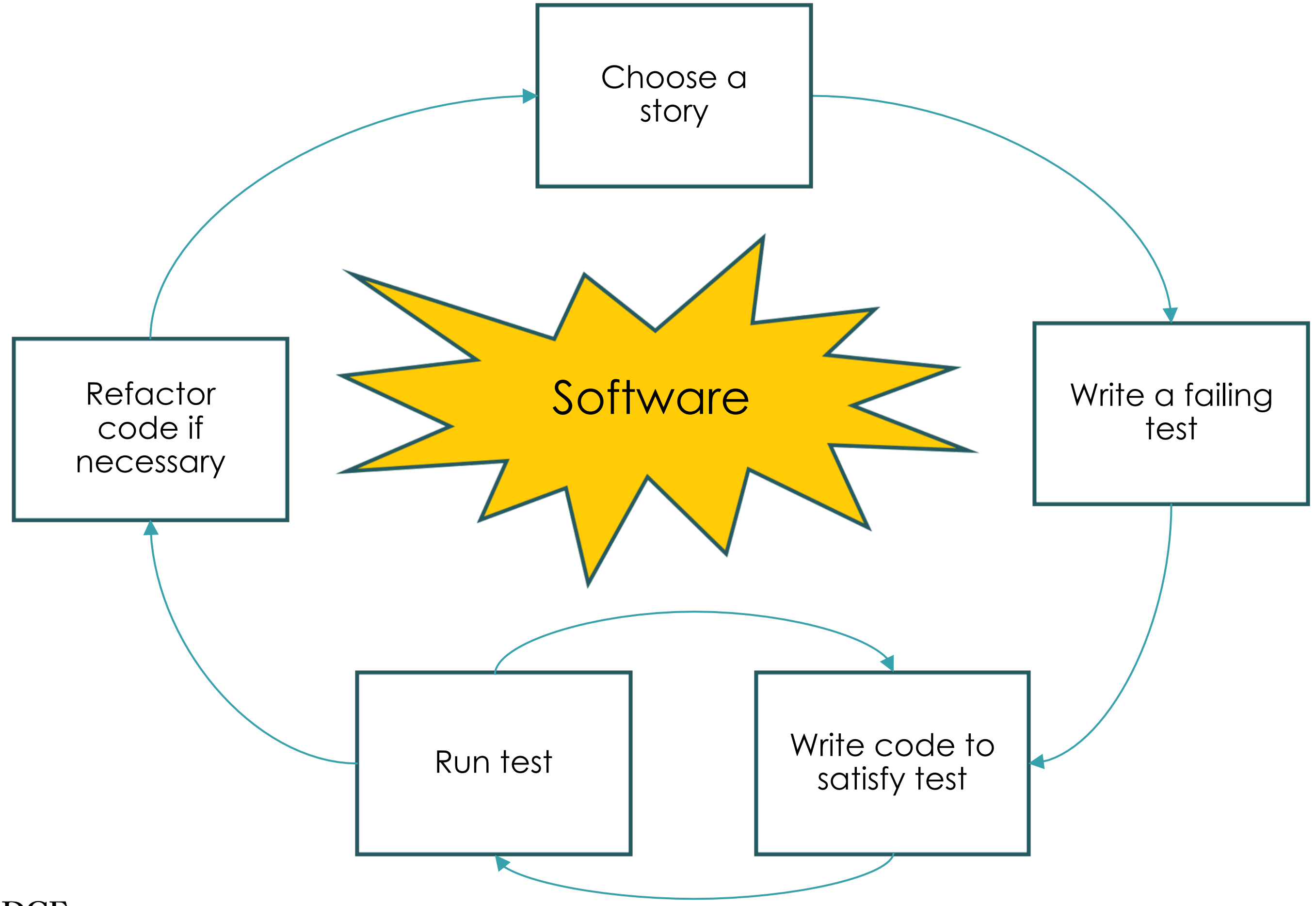
July 24, 2019

ORNL is managed by UT-Battelle, LLC
for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

Test driven development

- Often described as an iterative process to create software



Choose a story

Write a failing test

Write code to satisfy test

Run test

Refactor code if necessary

Software

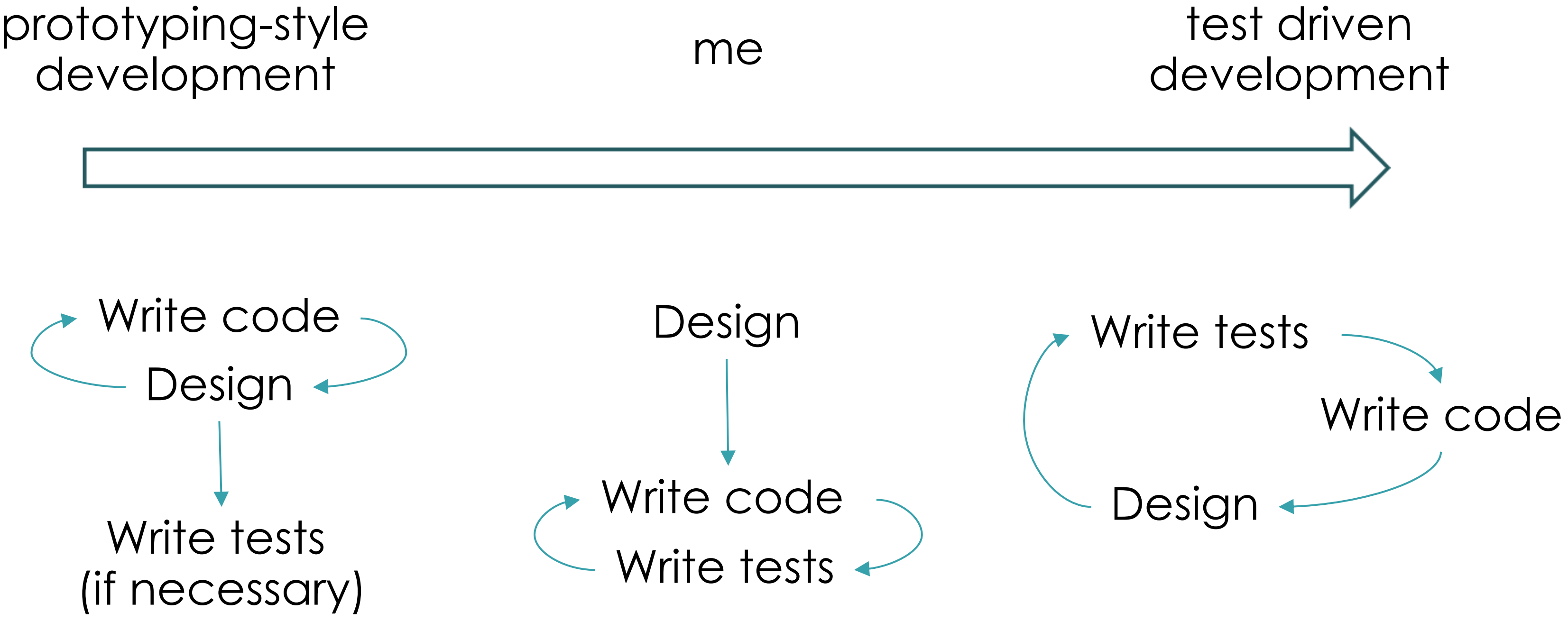
Questions that arise with this approach

- Most descriptions gloss over how you past the first step. i.e. how you get from a feature to writing a test
- What should the scope of the test be?
- How much code should be written on each iteration?
- What code should be refactored, and what that actually means?
- Should I be thinking about design issues as I'm doing this?
- Etc.

Why is it hard to use TDD?

- Most(?) developers start with a prototyping approach
- Many organizations view testing as a necessary evil or a defensive mechanism
- It’s easier to start doing something than to spend time thinking about it
- It’s more interesting to start doing something…
- TDD is a different paradigm so requires some education
  - It’s easy to get bogged down in minutia e.g. how much code should I write, what is refactoring, etc.
  - Prototyping is just coding so you can start as soon as you know how to code

Development “spectrum”



prototyping-style development

me

test driven development

Write code

Design

Write tests
(if necessary)

Design

Write code

Write tests

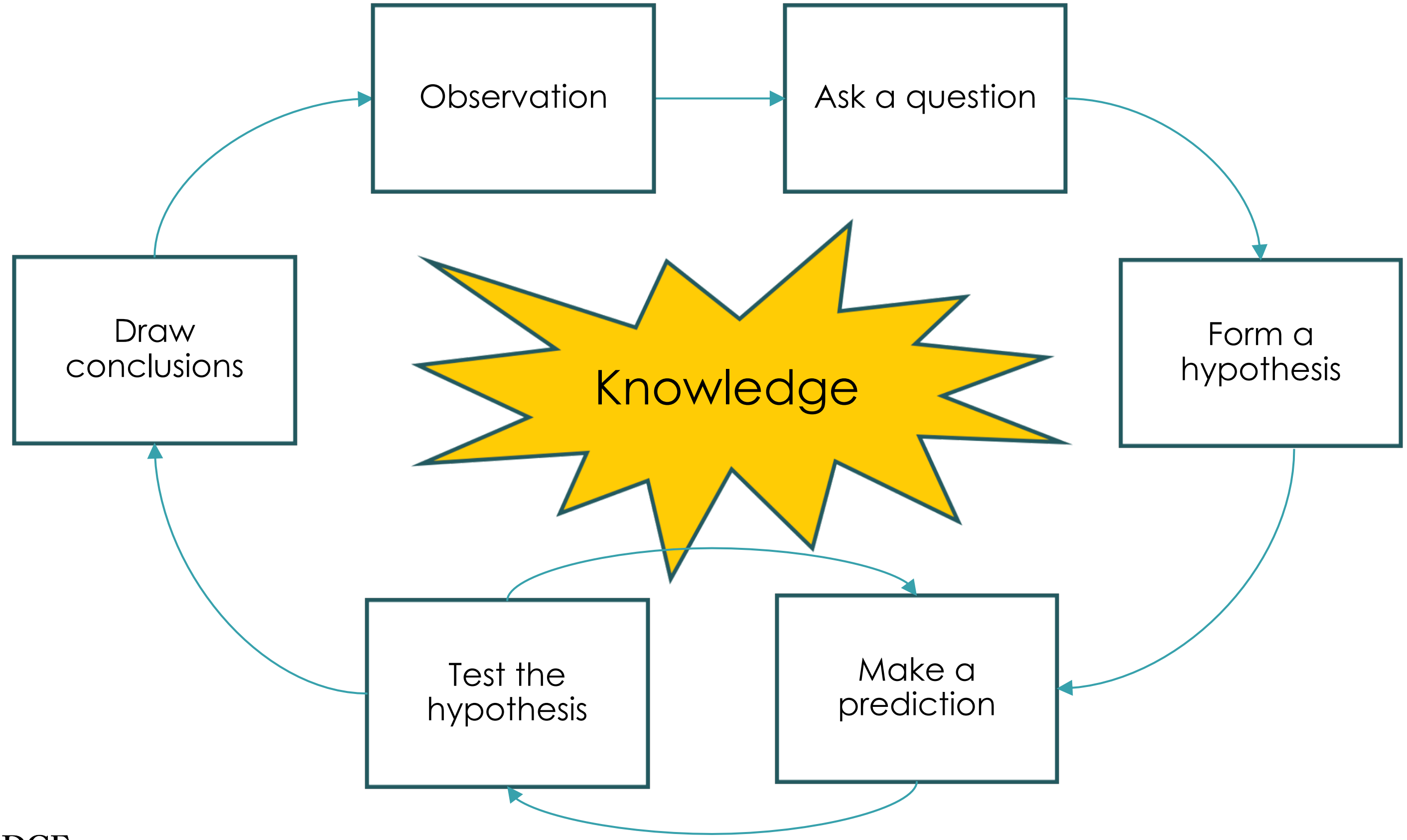
Write tests

Write code

Design

Scientific Method

- An iterative method for creating explanations



Observation

Ask a question

Form a hypothesis

Make a prediction

Test the hypothesis

Draw conclusions

Knowledge

Why is the Scientific Method so successful?

- "[Science] initiated the present era in human history, unique for its sustained, rapid creation of knowledge with ever-increasing reach" – David Deutsch, The Beginning of Infinity
- Key elements:
  - Rejection of authority
  - Testable, explanatory theories
  - A quest for good explanations†
- Two concrete outcomes:
  - Makes experimental results repeatable
  - Normalizes the process of performing experiments

† A "good" explanation is one which is hard to vary without changing the meaning. This is in contrast to a "bad" explanation, which is testable, but when refuted does not contribute anything towards understanding the phenomenon.

TDD as a method

- Suppose we consider TDD as a method (i.e. an empirical and iterative approach) rather than a mechanism (in the sense of an algorithm)
- This is the same approach that a scientist would take, namely
  - Formulate a hypothesis about the system and incrementally test those hypothesis against reality
  - If the resulting artifacts satisfy the demands placed on the system, they become the best model of the theory embodied in those demands
  - The artifact is never 100% correct, only true until falsified
- Thinking about TDD this way gives us a way of avoiding some of the previous pitfalls

A TDD method

- Think about the problem being solved and how to solve it
- Discuss ideas with someone or otherwise seek the knowledge of others about the problem in question.
- Try things out first, especially when starting a new task
- Ensure all existing assertions “fail to falsify” the system before changing anything
- Write some code that asserts something not currently true about the system (i.e. it falsifies the assertion)
- Run this code and confirm the new assertion - and only the new assertion - successfully fails
- Add just enough code to confirm the assertion “fails to falsify” (i.e. passes)
- Confirm that all the other assertions “fail to falsify."
- Change names, extract functions and do other refactorings necessary prior to adding another failing assertion
- Keep doing this until the problem has been solved

Final thoughts

- For sustainable software, extensive testing is essential (according to Michael Feathers, all code without tests is legacy code)
- But TDD requires great discipline
- We're scientists, so we're supposed to be applying scientific method in our work (in turns out this isn't always the case)
- TDD may be one way to help us think more scientifically about our software
- Conversely, as scientists, we should find TDD a more natural approach to developing software