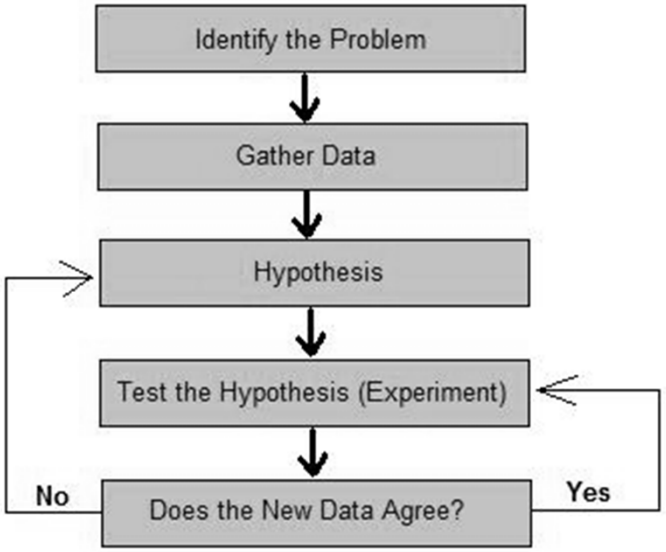
Debugging


Started Cosine Tape (Sine check)
Relay #70 Panel F
(moth) in relay.
First actual case of bug being found.


Identify the Problem
Gather Data
Hypothesis
Test the Hypothesis (Experiment)
Does the New Data Agree?
No
Yes

Debugging, in the Trenches

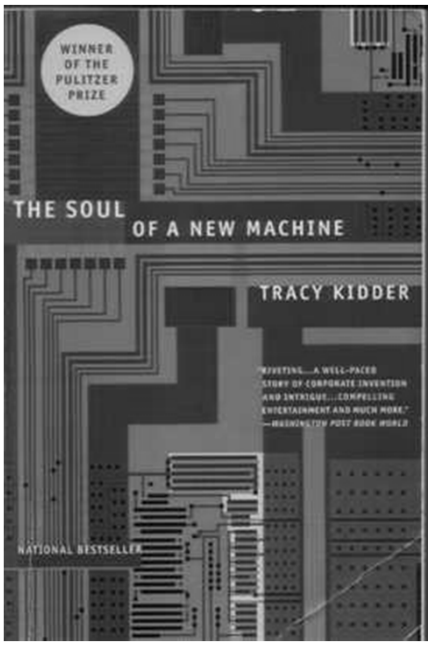
Rasala put his hands on his desk and buried his face in them.
It was just another routine day down at debugging headquarters.

In the back of Veres's mind still lies a small suspicion that the problem might after all be noise. And now – much to Guyer's delight, when he finds out later on – it is Veres himself who disconnects the I-cache. Then he runs the program past the point of failure, and everything works. He puts the I-cache back in and once again Gollum fails. This doesn't prove the IP is to blame, but it does tend to eliminate noise as a suspect, once and for all. . .

---

from THE CASE OF THE MISSING NAND GATE (Chapter 10 of Kidder's The Soul of a New Machine)

(The Soul of a New Machine)



Kidder's book tells the story of the development of a micro-computer in the early 80s. The book's a classic – won the American Book Award for non-fiction. Anyone who cares how computers are made (or how people work) should read it.

Chapter 10 is a classic story of debugging a hardware problem. The ideas apply just as well to software, and this is the best description of heavy-duty debug I've ever seen.

Debugging

- Debugging is really hard – even with a good failing test case in hand
- One of the most time-consuming tasks in software development
- Locating the fault is the most time-consuming part of debugging

Debugging

- Takes as much as 50% of development time on some projects
- Arguably the most scientific part of "computer science" practice
  - Even though it's *usually* done in a totally *ad hoc*, haphazard way!

"Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it." *- Brian Kernighan*

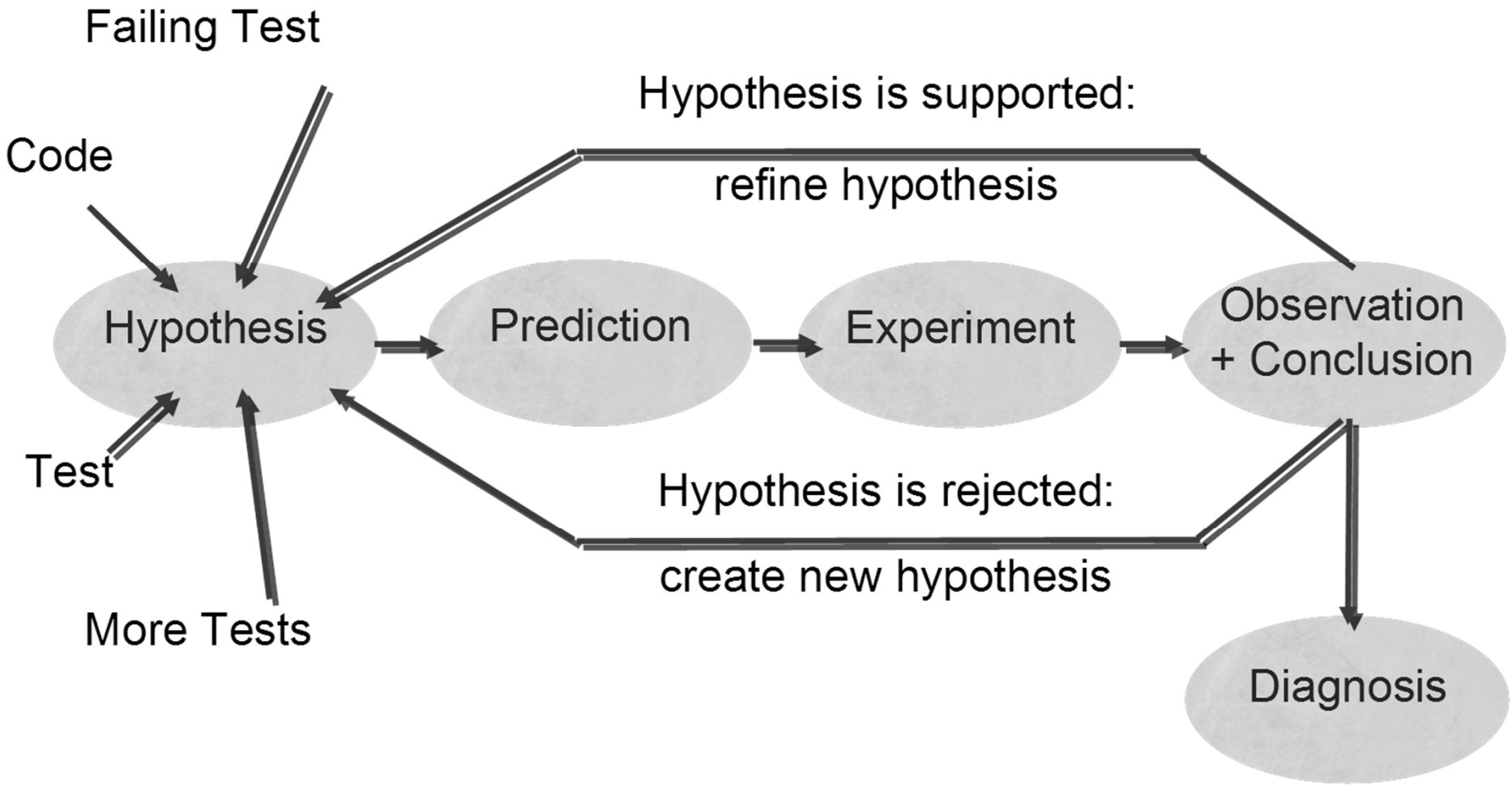
Debugging and Testing

- What are test cases *for*, anyway?
  - Often: so we can locate and fix a fault
  - Or: so we can understand how serious the failure is, and triage / “flight rule” it away
    - If we have many bugs, and some may not be important enough to merit resources
    - Or if there is a reason we *can’t* change the code and have to work around the problem
- In either case, we have a “debugging” task at hand – must at least *understand the failure*, even if only to triage

Scientific Debugging

- Test cases (ones that fail and ones that succeed) can be the experiments we perform to verify our hypotheses
  - The failing test case informs us that there is a phenomenon to explain (apple on the head)
  - Generate (or examine) more test cases to find out more about what is going on in the program

Scientific Debugging



Testing for Debugging

- Several ways to use test cases in debugging:
  - Test case minimization
  - Shrink the test case so we don't have to look at lots of irrelevant or redundant operations
  - Fault localization
    - Give suggestions about where the *fault* may be (based on test case executions)
  - Error explanation
    - Give a "story" of *causality*
      - *(A causes B; B causes C; C causes failure)*

attempt to automate part of scientific debugging