Debugging With PostgreSQL – A Strategic Approach

- Debugging is the elephant in the room
- Can take 50 to 90% of the time
- But gets < 10% of the attention!
- PostgreSQL has great tools (see references)
- But they are most effective when deployed as part of an overall strategy

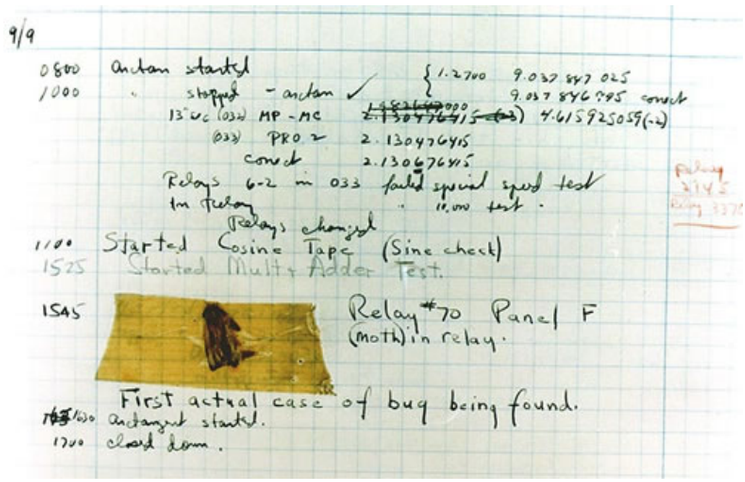
"Debugging is about:"

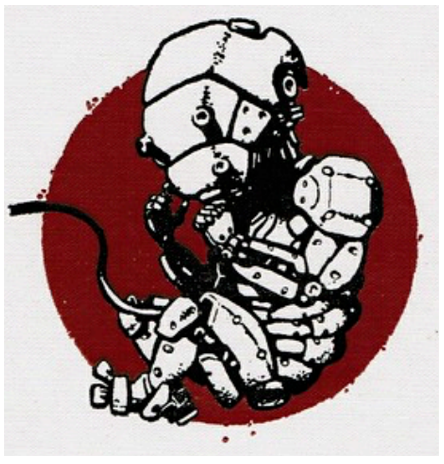
— John Sonmez
https://simpleprogrammer.com/

- Finding the source of a problem
- Identifying possible causes
- Testing out hypotheses
- Eliminating that cause
- Ensuring it will never happen again

Sounds a lot like the scientific method!

Introduction





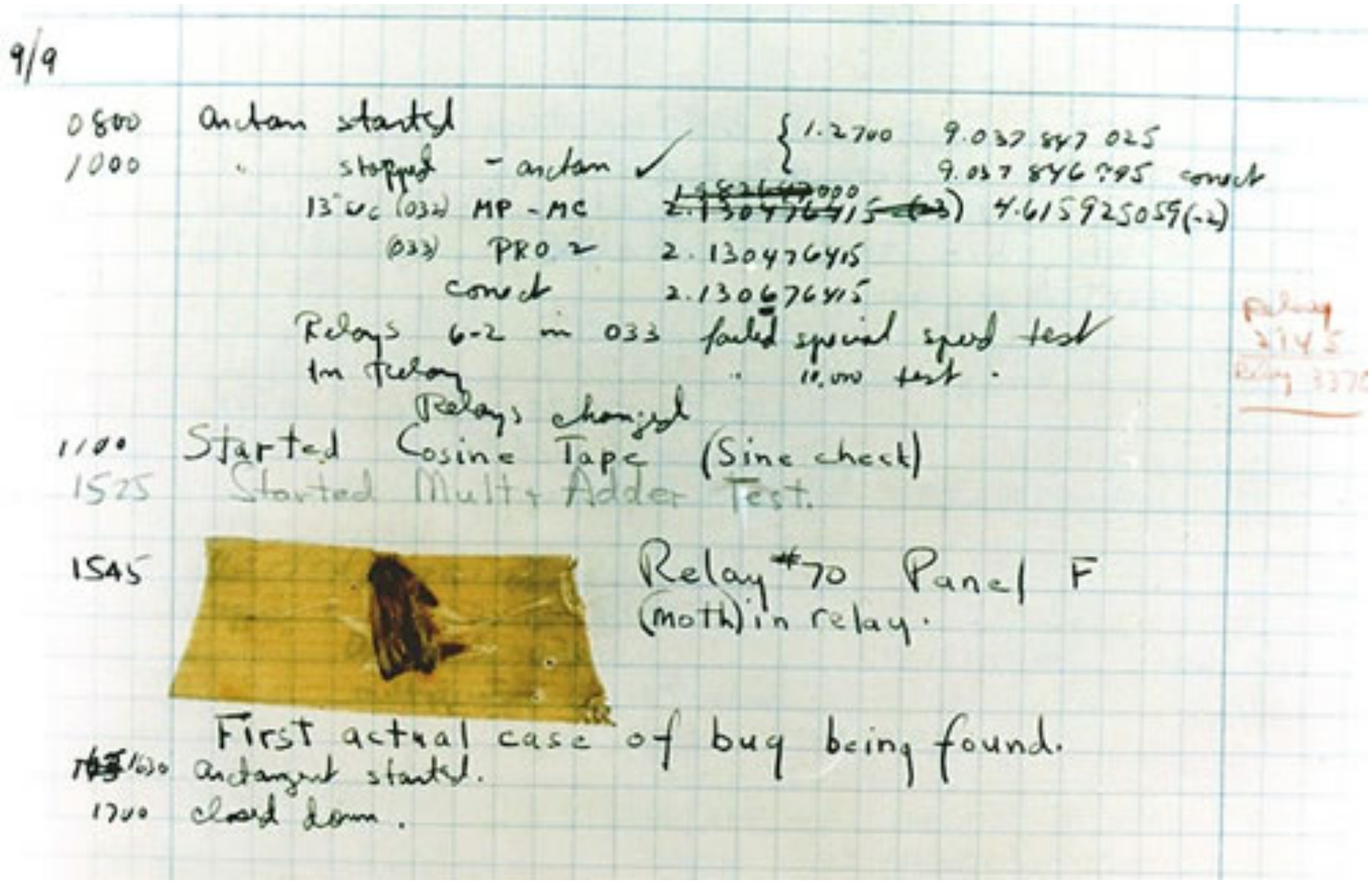
- First database
- First bug
- Root cause
- Life cycle approach

The First Databases

Early tablet recording the allocation of beer,
3100-3000 B.C.E

- Clay tablets
- Taxes
- Origin of writing & counting
- Fully baked!

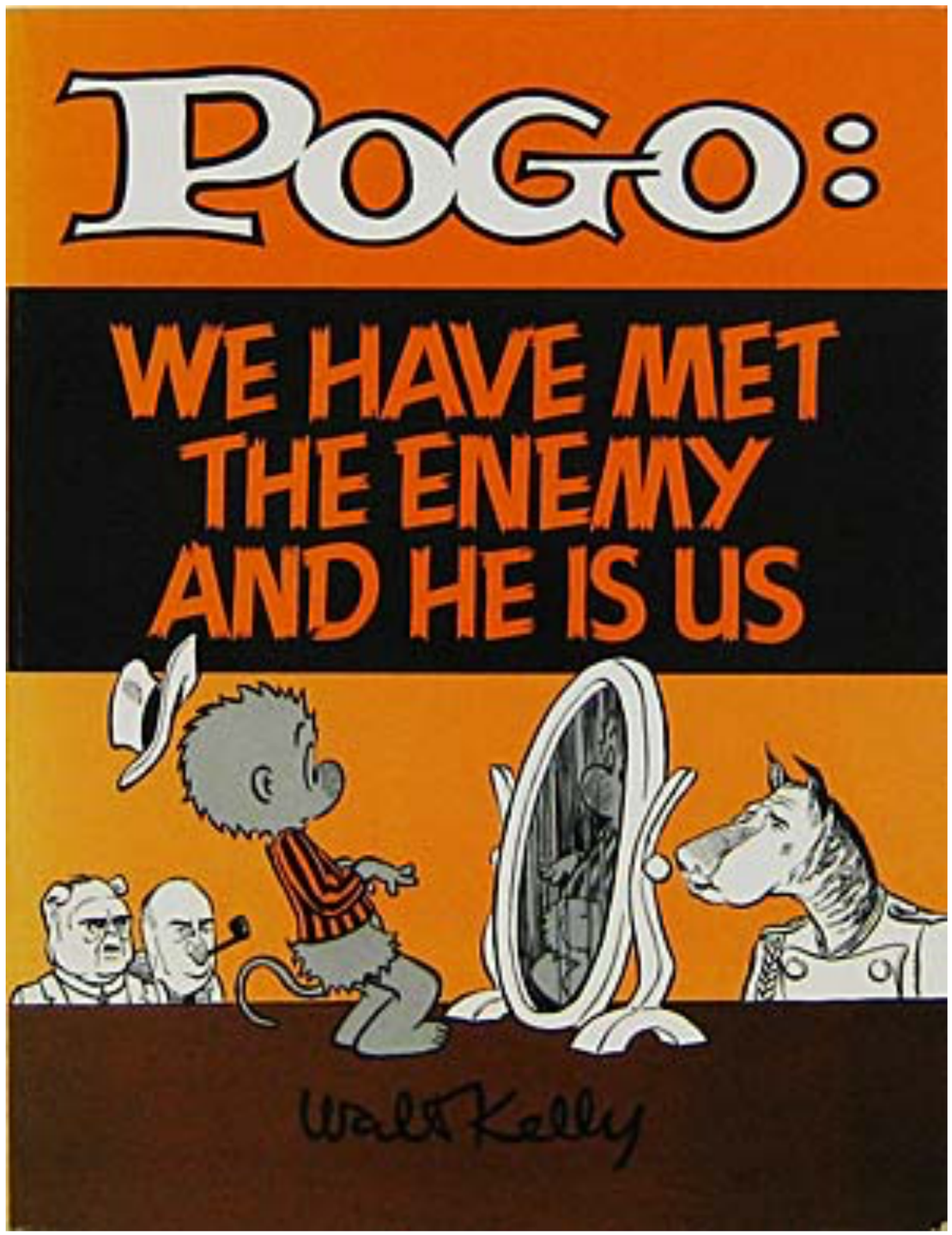
The first Bug


9/9

0800 Arctan started
1000 " stopped - arctan ✓ { 1.2700 9.037 847 025
9.037 846 ?95 correct
13"UC (032) MP - MC ~~1.982[illegible]000~~ ~~2.130476415 (-2)~~ 4.61592505 9(-2)
(033) PRO 2 2.13047 6415
correct 2.13067 6415
Relays 6-2 in 033 failed special speed test
in relay . 11.00 test .
Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Multy Adder Test.
1545 Relay #70 Panel F
(moth) in relay.
First actual case of bug being found.
1700 closed down.

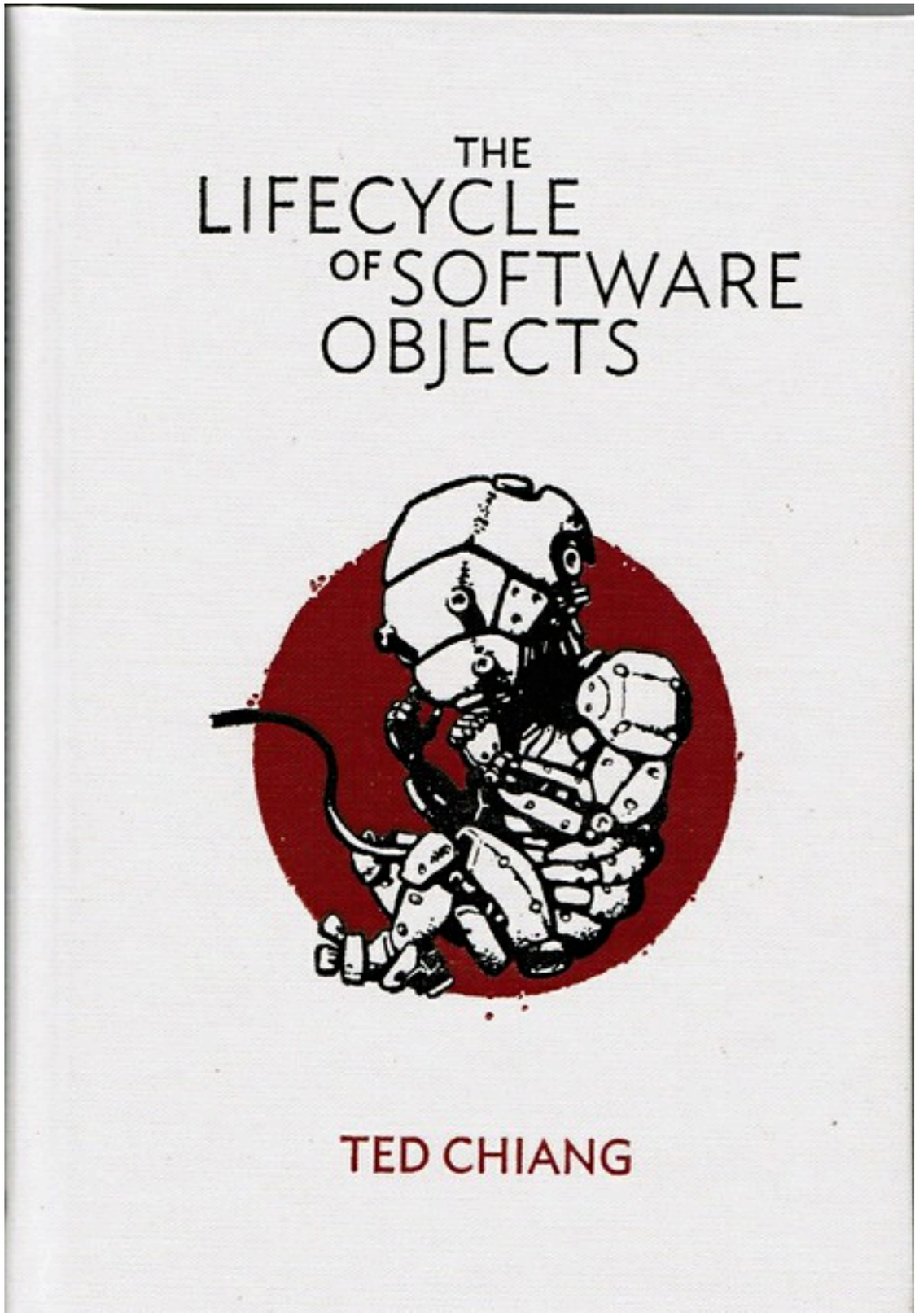
- Middle English "scarecrow", "hobgoblin"
- Edison 1878
- Hopper 1945
- Error in process, not just an error

The root cause



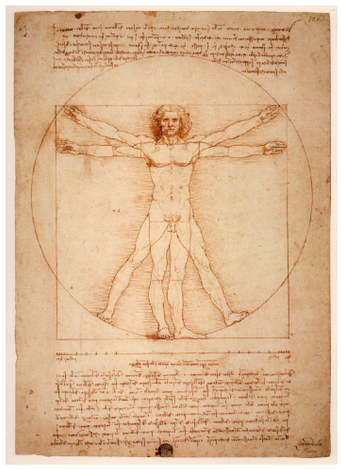
- Depending on project, 10 to 50 to 90% of effort
- Generally self-inflicted
- Can also be the result of bitrot, evolving system, changing expectations …

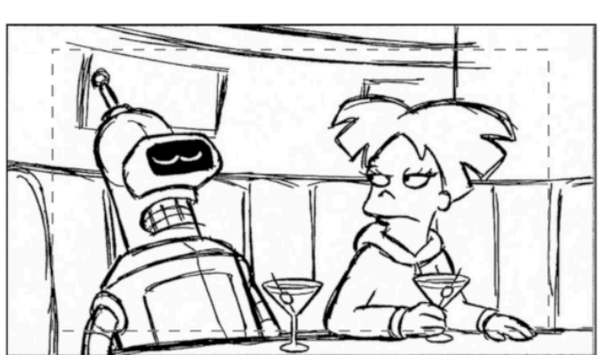
Life cycle



- Development
- Debugging
- Maintenance

Development

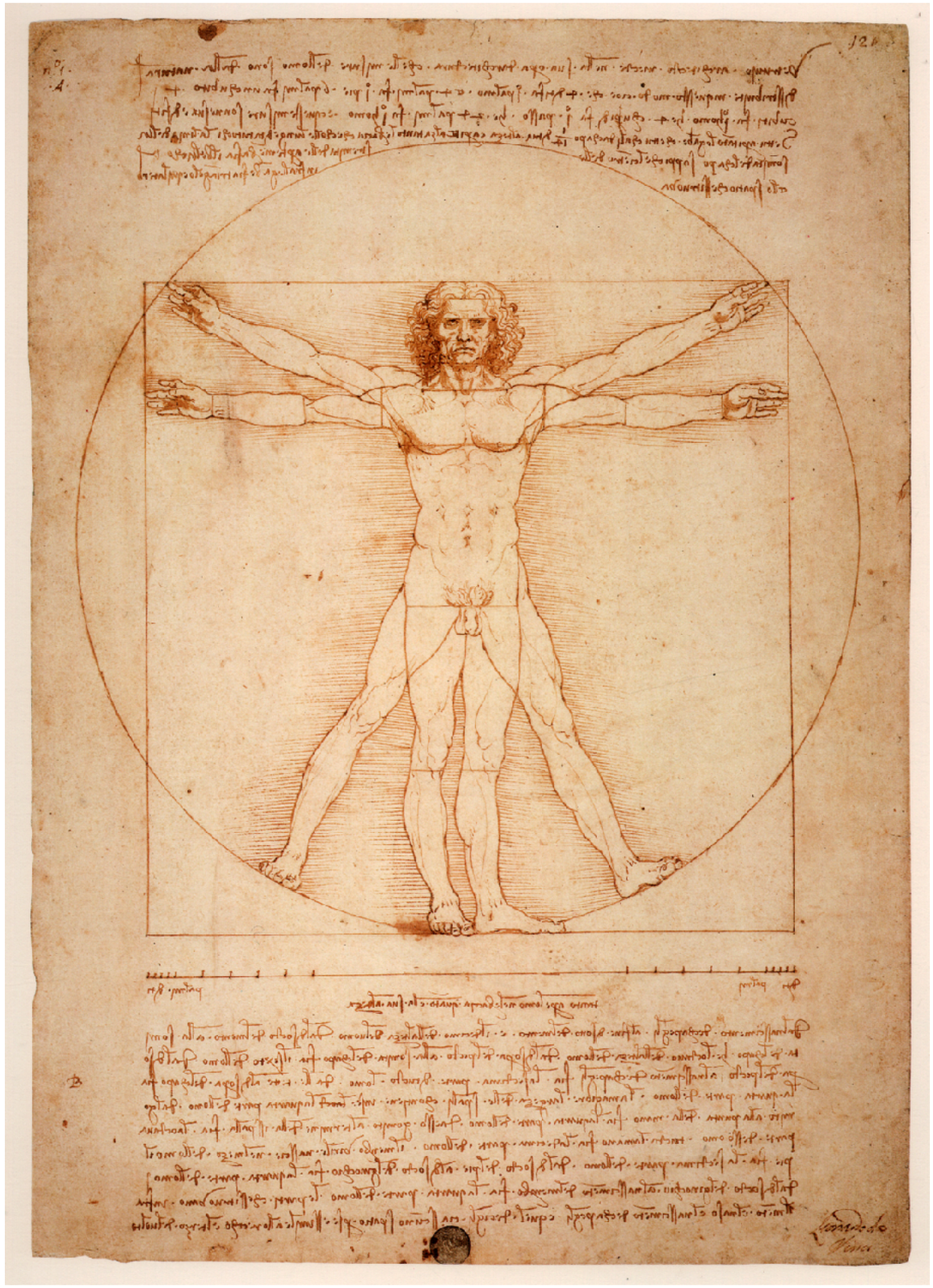




- Specs
- Structure
- Story
- Scenes
- Systems

The Wreck of the Vasa

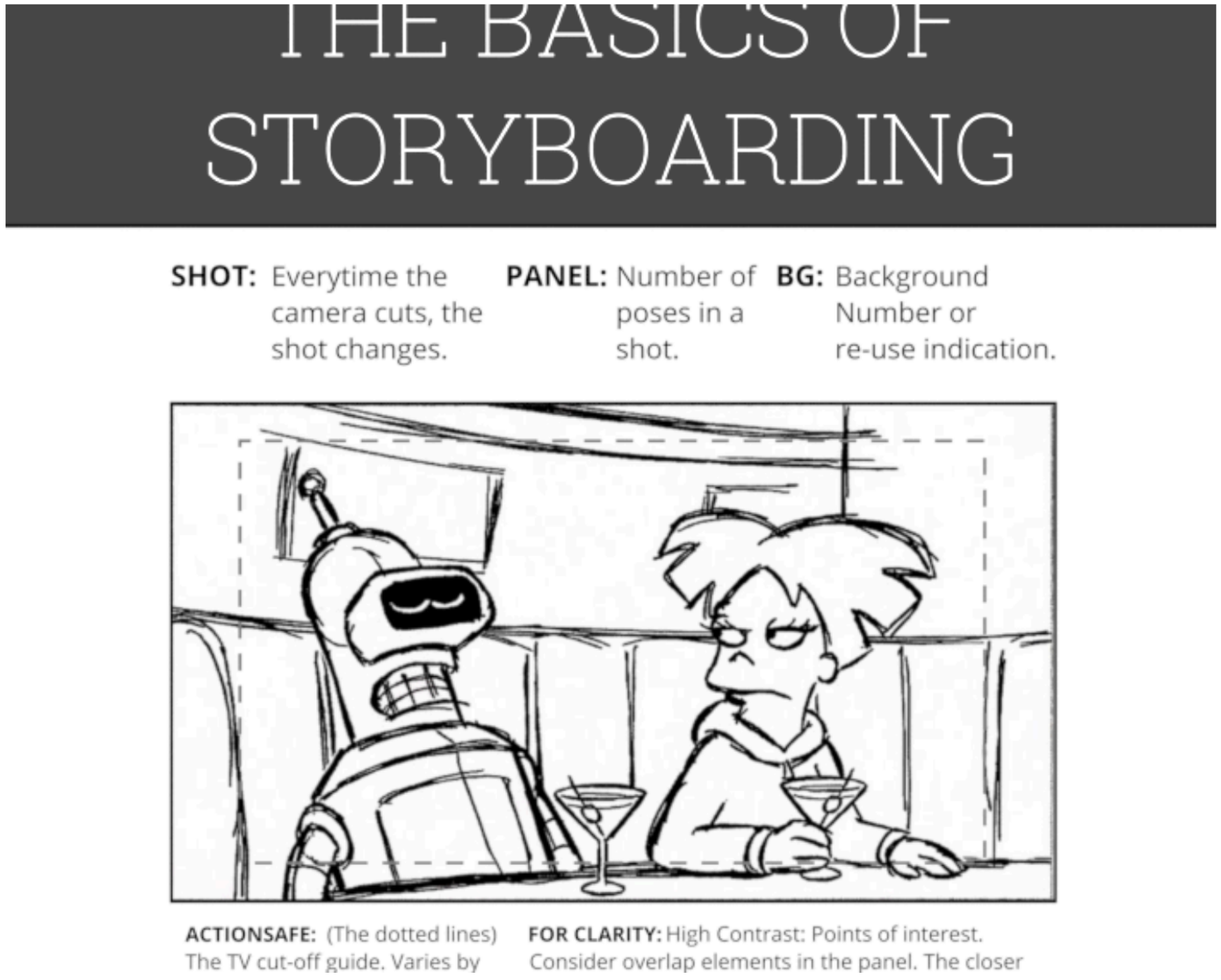
- Top heavy design
- Last minute changes
- Unnecessary features
- Not enough ballast
- It is never the King's fault



Structure

- Tables = bones
- Foreign keys = tendons
- Stored procedures = muscle
- Views = skin
- Domains = types of cells
- Exceptions = nerves?
- Logs = memory?

Story



- Tell me a story
- Database as telephone exchange in time
- Use cases
- Mr. Client, meet Ms. Server
- Getting the picture

Break Story into Scenes

- Block out the code
- Make a good entrance
- Identify the characters
- Rehearse the scene
- Make a good exit
- Get audience feedback

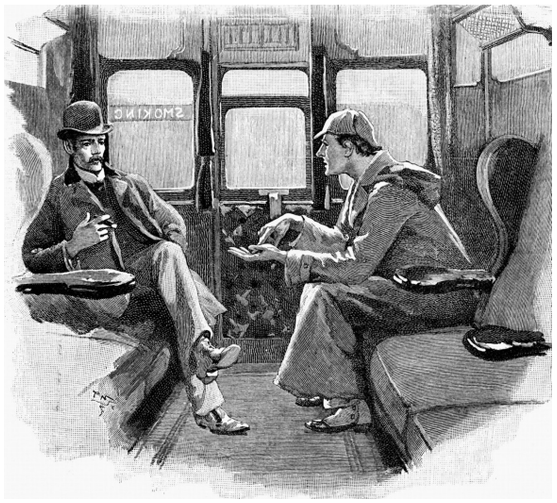
Systems Test

Ariane 5 Flight 501

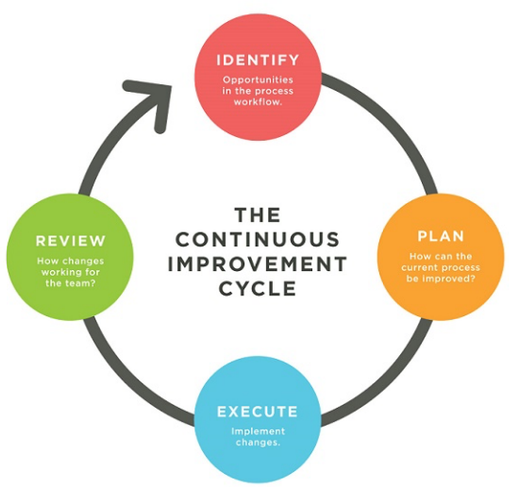
June 4, 1996

- French rocket reuses code from Ariane 4
- 5's faster engines triggers bug in arithmetic routine in flight computer
- Convert 64-bit FP to 16-bit signed int

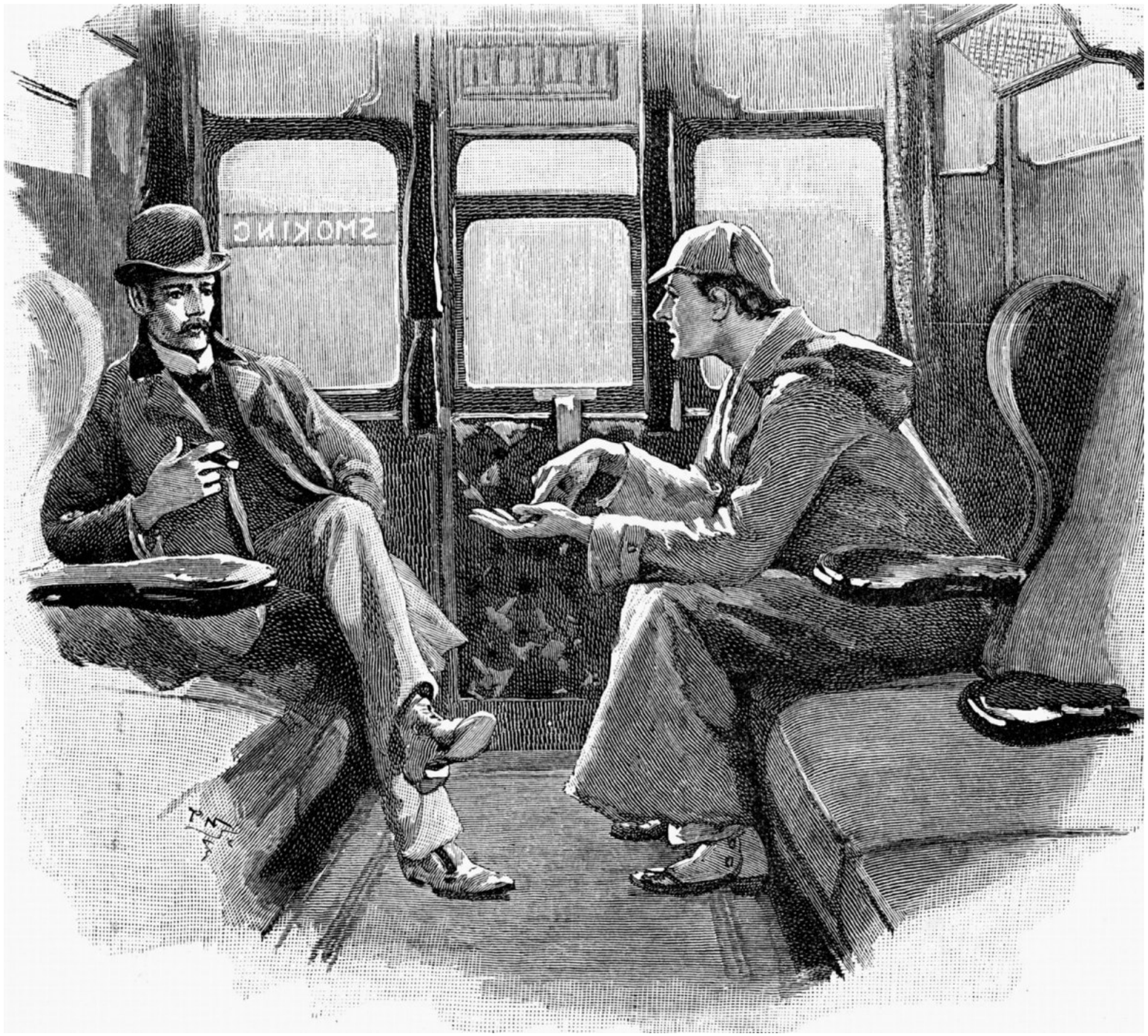
Debugging



- Detection
- Digging into the machine
- Taking your best shot at the bug
- Prevention

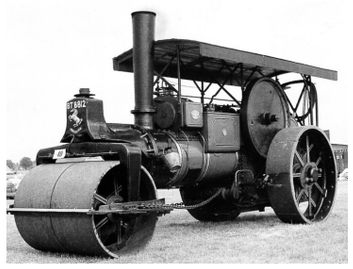


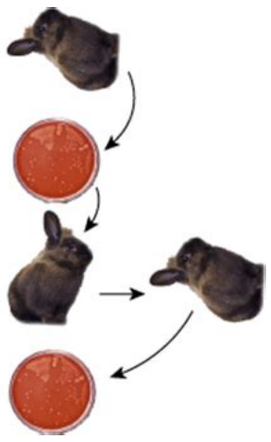
Detection

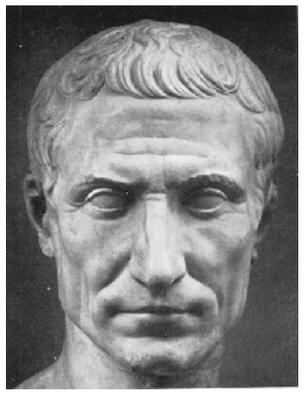


- Exceptions
- Error logging
- PGBadger & friends
- User reports
- …

Debugging How To







- Write perfect code
- Cat in a box
- Flatten ‘em
- Eyeball the code
- Give ‘em a taste of science!
- Divide et impera

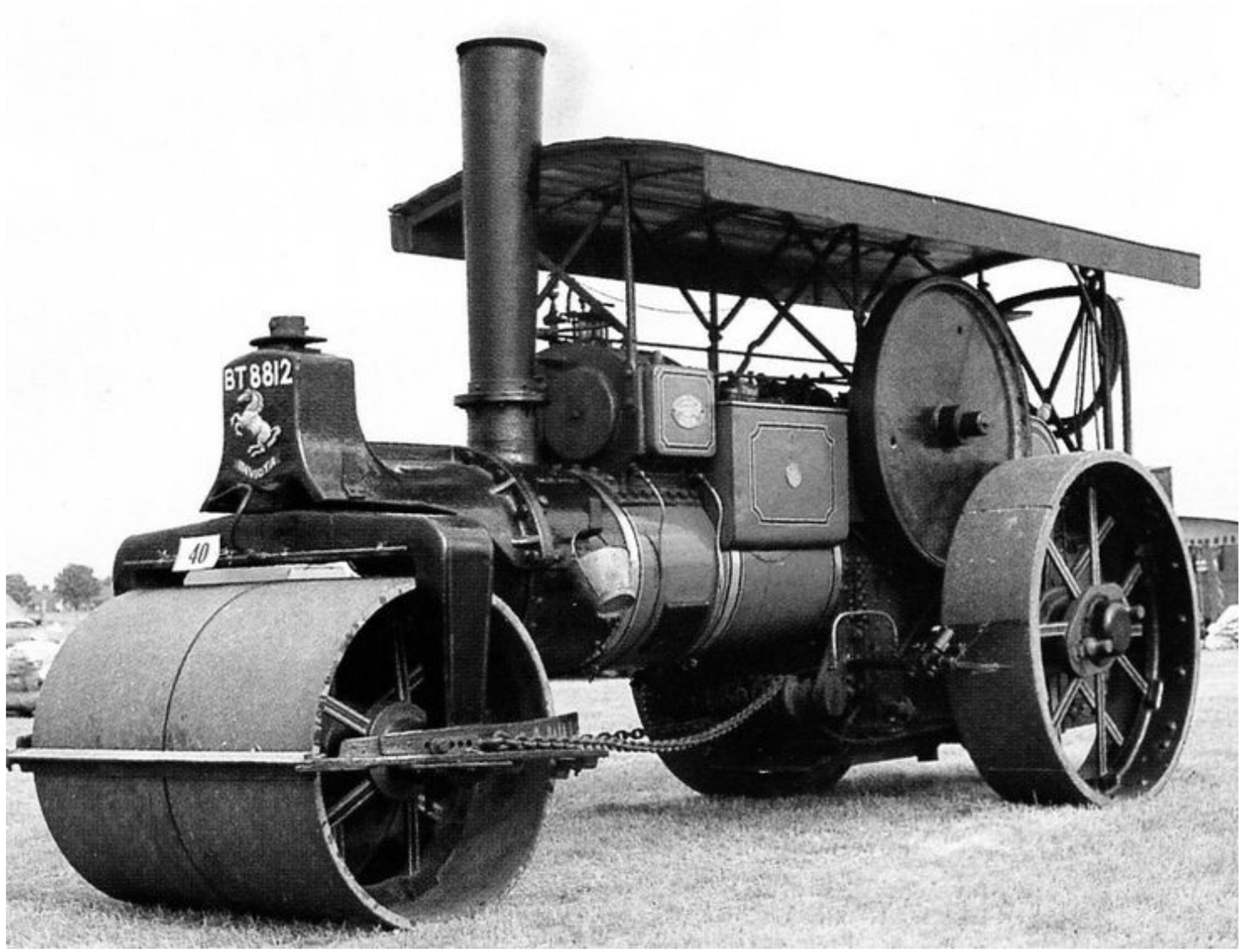
Write perfect code

- At a conference, I asked Bjarne:
- How do I debug C++?
- "Write perfect code!"
- Annoying …
- But not without a grain of truth

Cat in a Box

- Claw in every direction until you see daylight
- Sometimes necessary
- Futzing

Linear Debugging

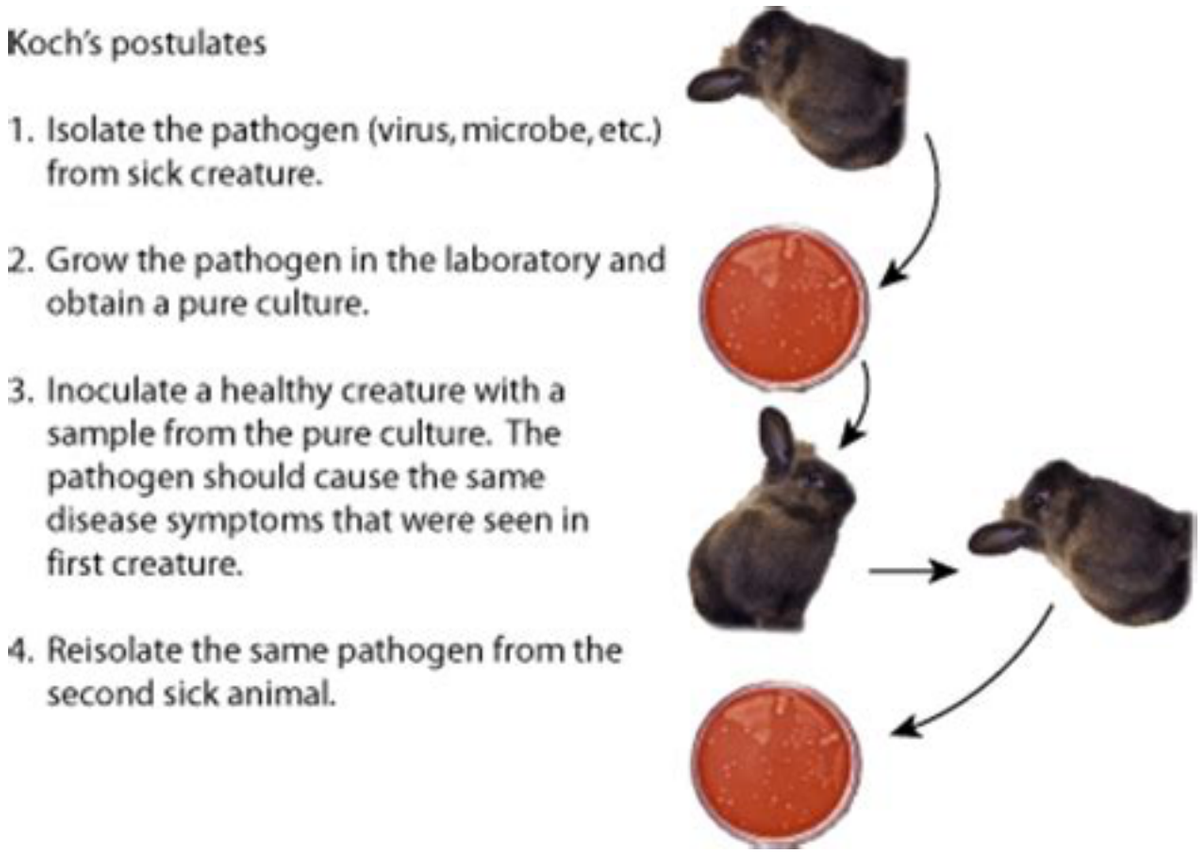


- Raise notice, print statements
- Debuggers
- Linear process, so slow
- Should usually be the last resort

Hairy Eyeball

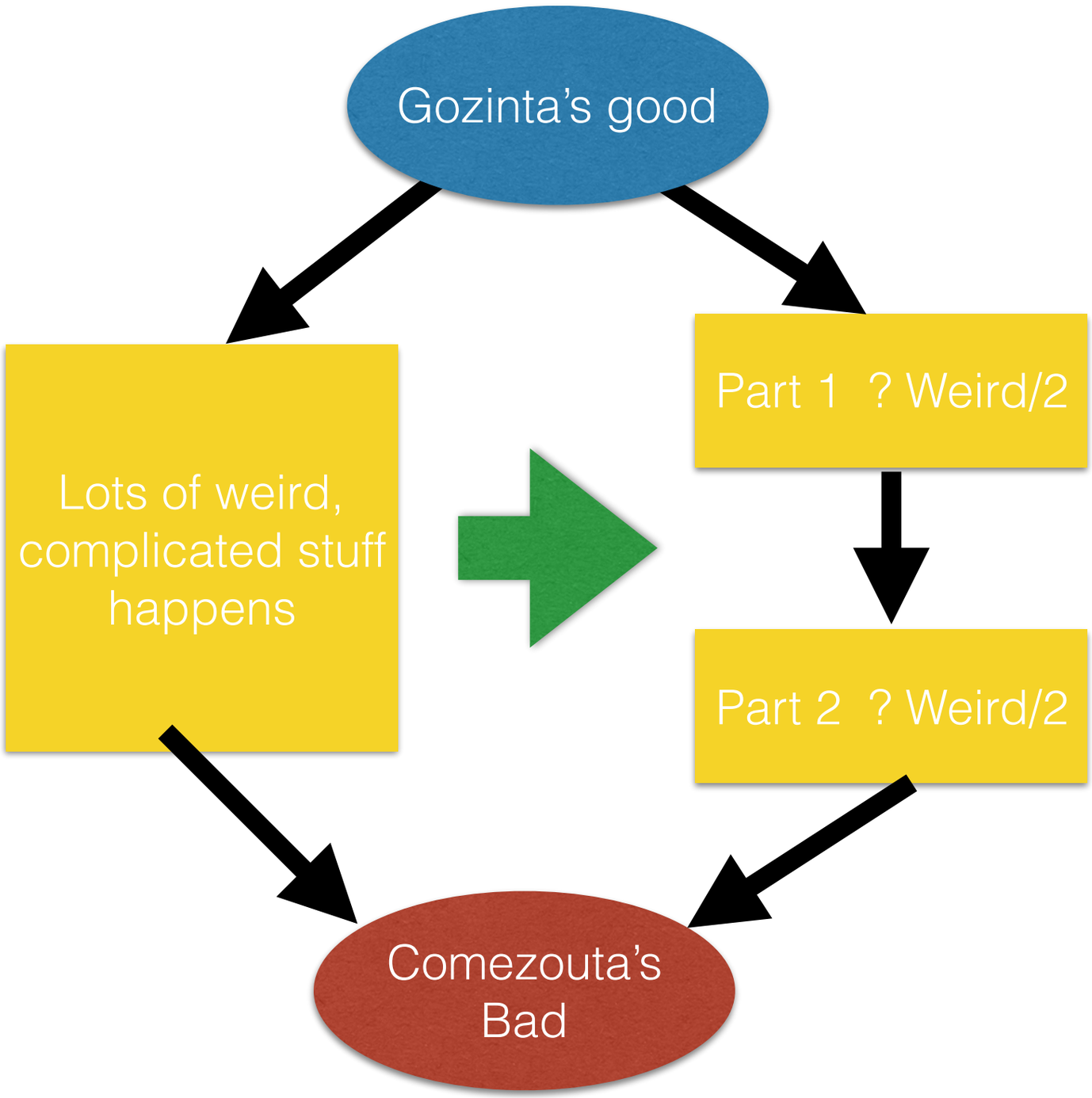
- Sight check
- Don’t compile till you think it will work
- If you can’t check by sight, the module is too big
- Get a 2nd pair of eyes on the problem

Scientific approach



- Why seen here?
- **But** not here?
- Can we reproduce the bug at will?
- Can we isolate the bug?
- What is the minimal test case?

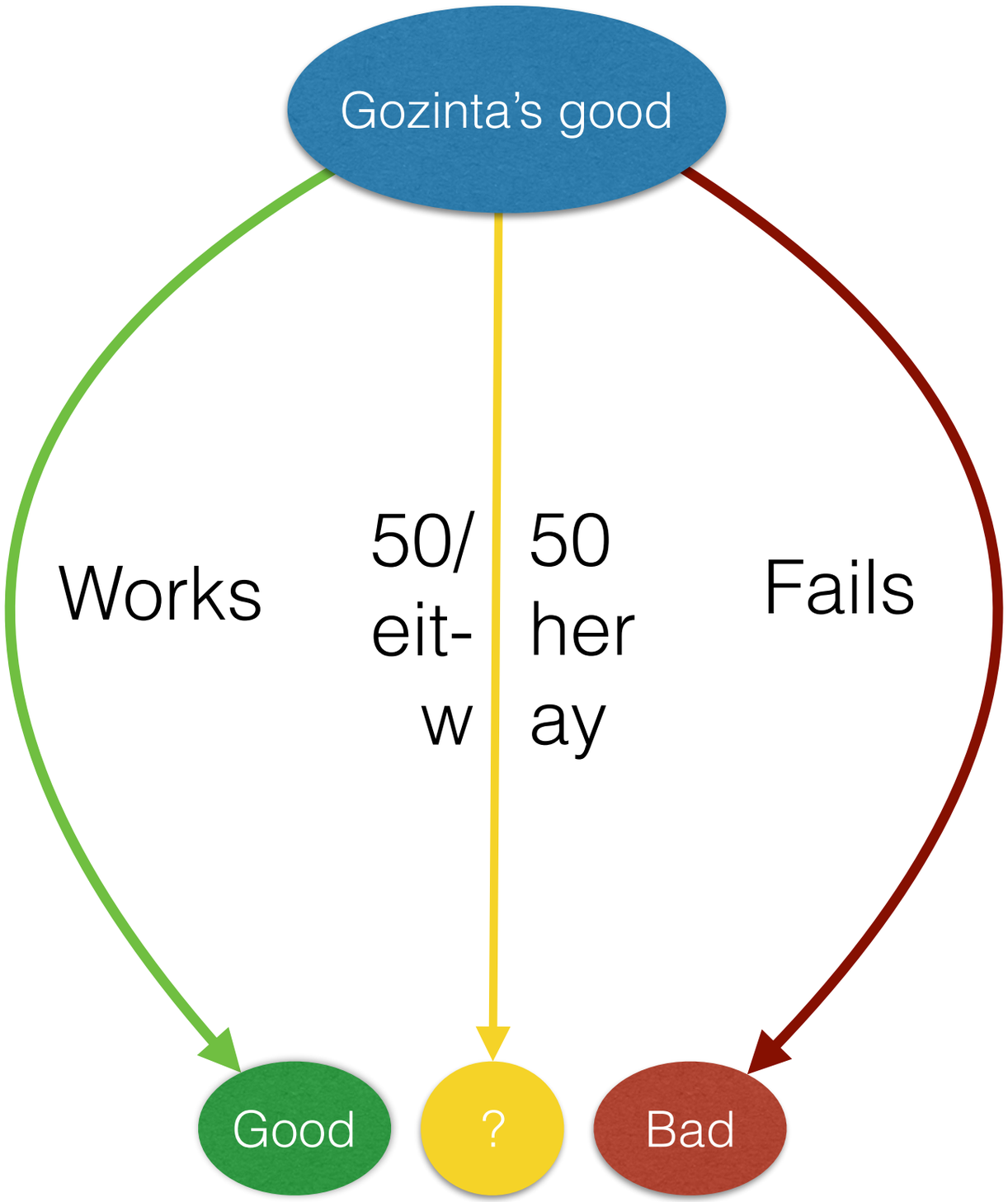
Gozinta's good, Comezouta's Bad



- Double check the Gozinta's anyway
- Do basic cleanup first: FKs, types, check constraints, …
- Don't guess, instrument
- Check entrance values and returns
- Prefer **binary** search to linear; keep cutting the hiding space for bugs in half

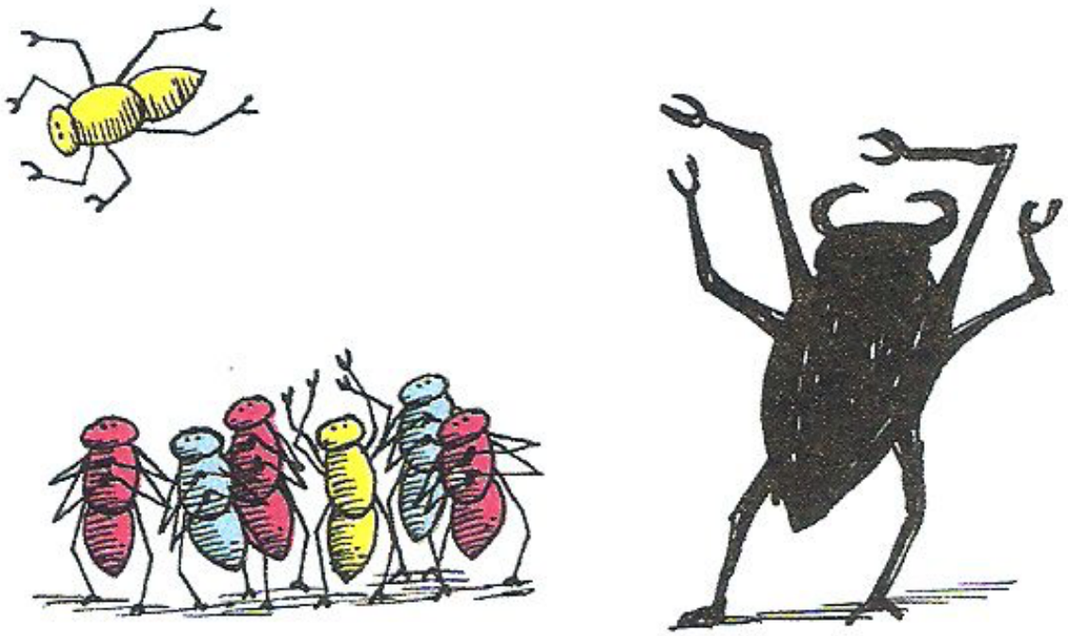
Left side Works …

…right side Fails



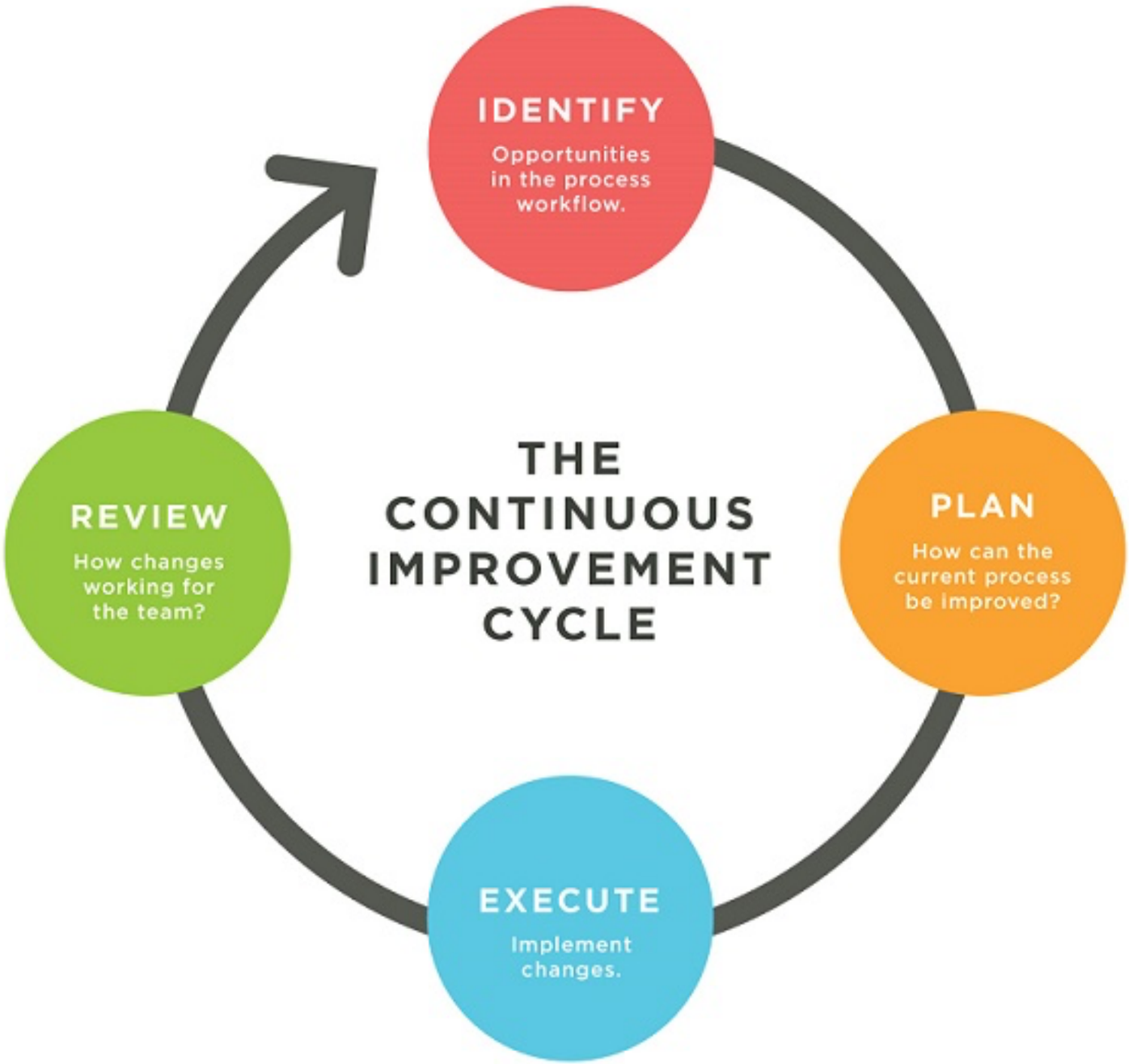
- Find one case that fails
- And one that works
- Walk them towards each other
- Using **binary** search (roughly 50% chance of bug in each half)

Remediation



- 50% of bug fixes create a new bug
- Old code often conceals critical business logic
- Laparoscopy
- Madame La Guillotine
- Hybrid approaches

Kaizen (Continuous Improvement)



- Find root cause, not just immediate trigger
- Bugs travel in packs: look for similar problems
- Report earlier
- Prevent entirely
- Eliminate the possibility

Maintenance

- Reporting
- Monitoring
- Technical debt
- Remediation

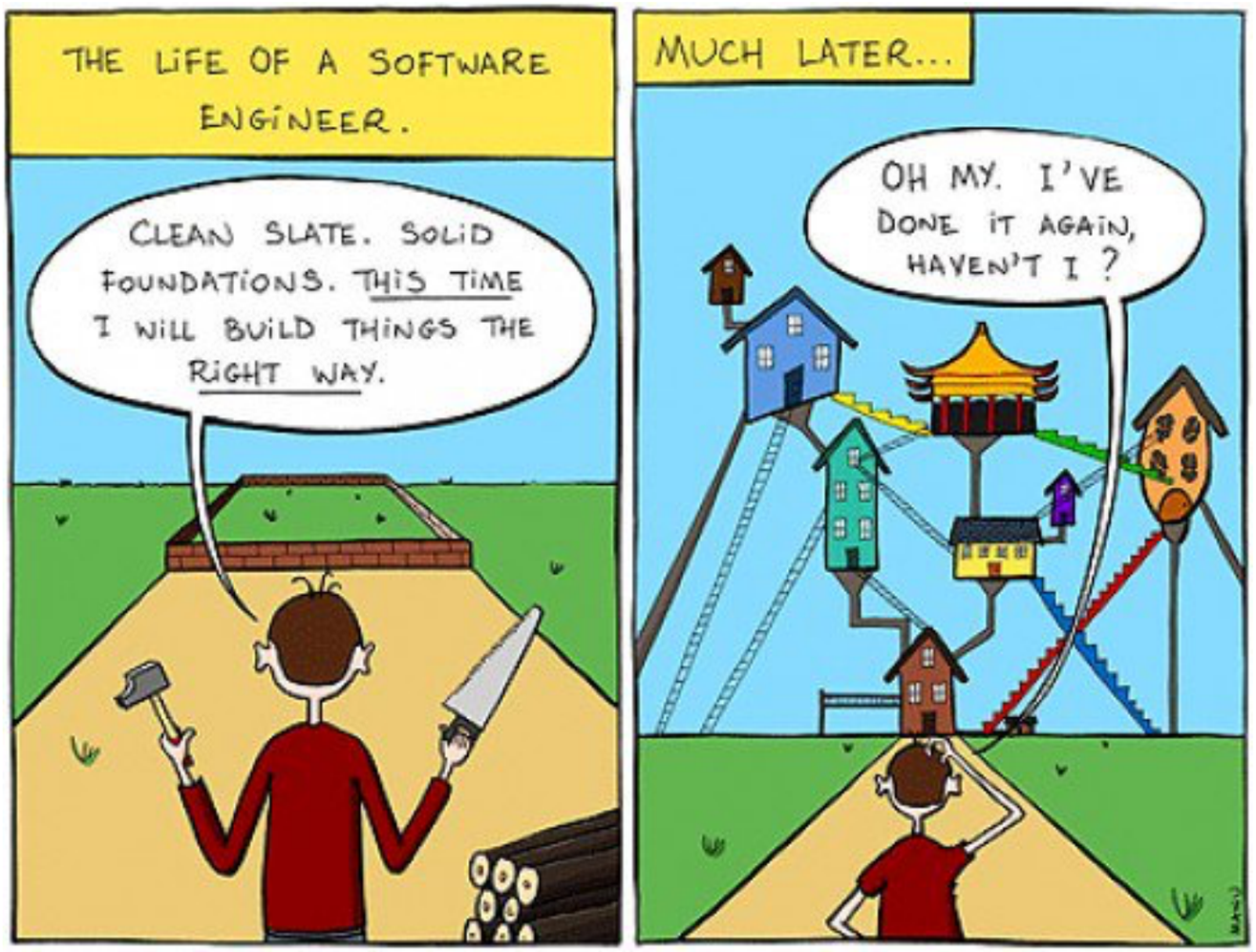
Reporting the Next Bug

- Exception handling
- Error logs & tables
- PGBadger & other tools
- Helping users report errors
- Walking the beat

Monitoring

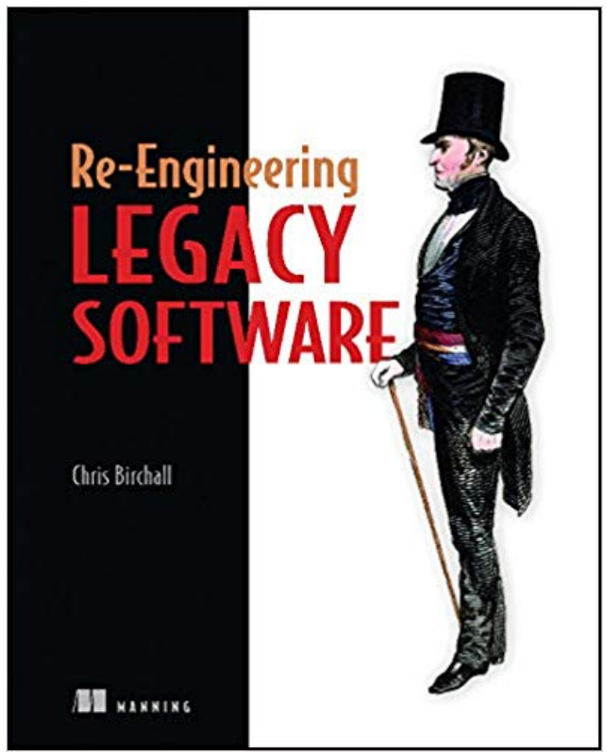
- pg_stat_activity, pg_stat_statements, …
- PGBadger & other tools
- Purpose-built
- Momjian's blog https://momjian.us

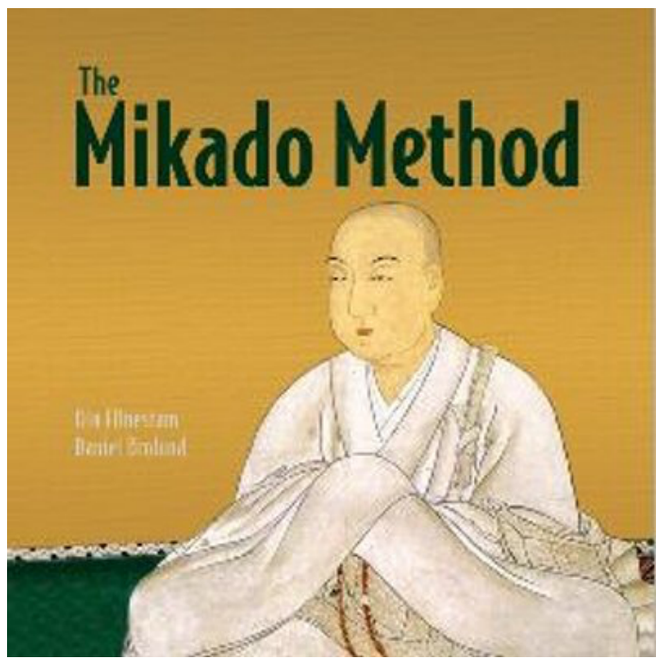
Technical Debt

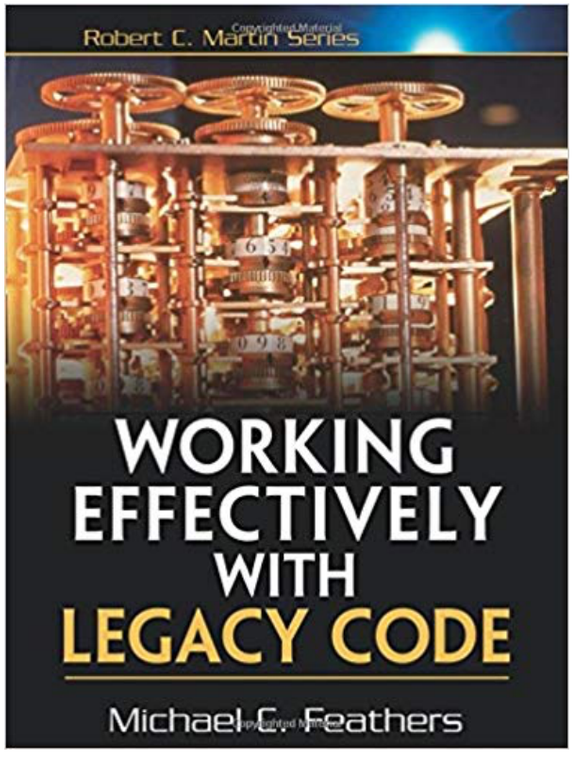


- Overcoming code fear
- Stylistic analysis
- Laparoscopy
- "Watch out for stobor!"

Remediation

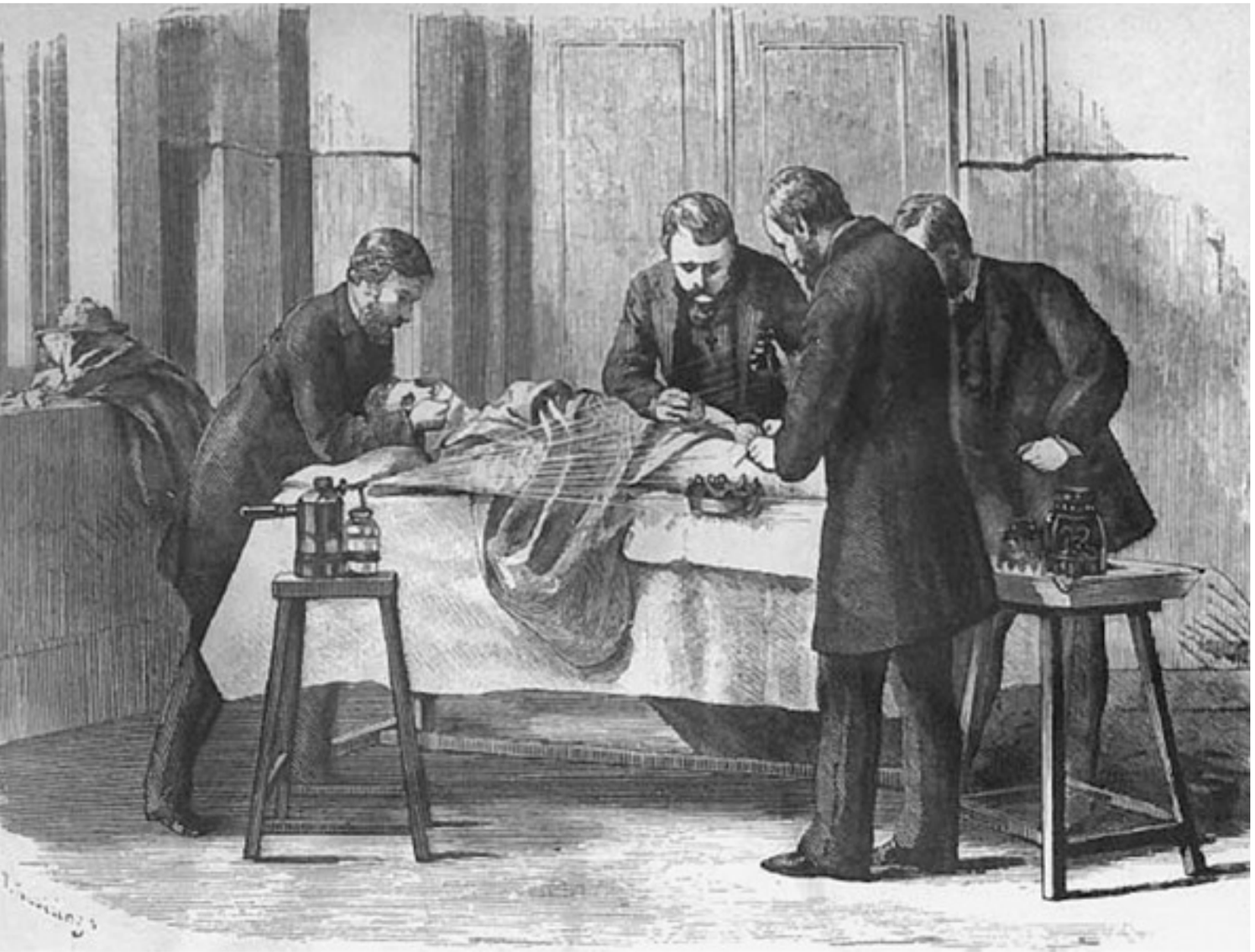






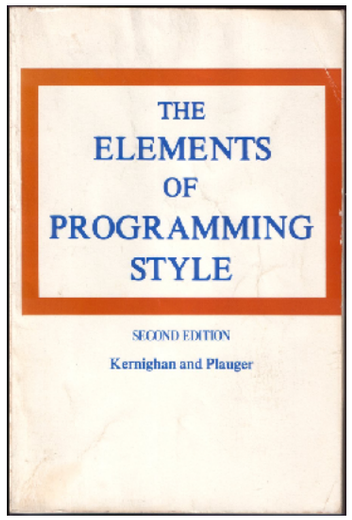
- Much of our lives but little of our literature
- Above all, do no harm
- Think globally, act locally
- When and how to refactor
- When it is time for "The Big Rewrite"

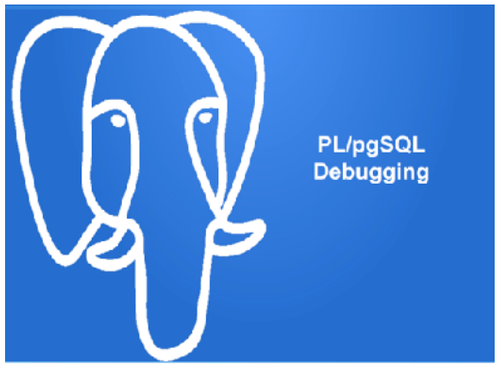
General Principles

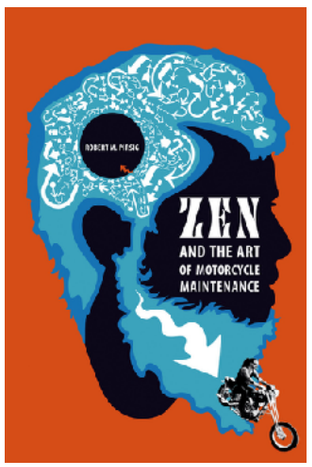


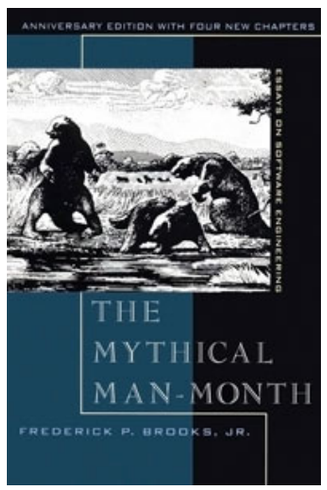
- Consider short & long term economics
- Take a systems level approach
- Work one step at a time
- Know your machine
- Know yourself

References









- RTFM
- Mlodginski - PL/pgSQL Debugging
- Kernighian, Ritchie - Elements of Programming Style
- Booth - The Mythical Man-month
- Pirzig - Zen and the Art of Motor cycle Maintenance