“If you torture the data long enough, it will confess to anything.”

– Ronald Coase
Key Embedded Software Metrics

- **Anti-Patterns:**
  - Development effort > validation effort
  - Too many lines of code per hour
  - Peer review finds <50% of all bugs

- **Healthy project metrics:**
  - About 2-3 hours of validation effort per hour development
    - Tester:Developer head count ratio is about 1 to 1
  - Productivity of 1-2 lines of code per hour for solid software
    - This includes entire process (requirements through acceptance test)
  - Peer review should be finding >50% of all defects
Software = Design + Testing

System level
- Safety requirements specification

Software level
- SW safety requirements specification
- SW architecture specification

Validation
- Test concept: SW system tests, integration tests
- Module test concept

Module Design
- Module testing

Integration testing
- Validation testing

Validated software

DESIGNERS

Figure H.1 – V-Model for the software life cycle

IEC 60730 Appliance Safety

IEC 60730
Tester to Developer ratio varies depending on situation

- Web development: 1 tester per 5-10 developers
- Microsoft: 1 tester per 1 developer
- Aircraft controls: ~5 testers per 1 developer

EMBEDDED SW PROJECT EFFORT

- 25% DEV: Peer Reviews & Unit Test
- 25% DEV: Design & Implementation
- 45% TEST: Integration Test, System Test, Regression Tests
- 5% SQA/PPQA

Typical Effort Distribution

- 50%/50% Head Count
- 25%/75% Effort
Code Productivity

- **Productivity 1-2 lines of code/hr (including testers)**
  - Perhaps 3 lines/hr with Agile, but that speed increases quality risk

- **High lines of code/hr ➔ cutting corners**
  - Partial requirements, no design?
  - No peer reviews?
  - Only system level testing?

- **$25-$75 / line of source code**
  - All-in cost, including entire V process, until field testing
  - “Maintenance” can cost more, but might count as new project
Peer Review Effectiveness

- Good peer reviews find 50%-70% of the defects
  - Fewer than 40%-50% of defects found in peer reviews mean they are BROKEN

- Peer Reviews cost perhaps 5%-10% of total project cost
  - Let’s do the math:
    - Peer reviews process about 100 lines of code per hour total
    - Three reviewers ➔ 33 lines of code per person-hr
      = 0.033 hours per line of code reviewed (2 minutes)
    - 0.033 hours review / .5 hours per LOC total = 6.7% for code review
    - Plus review requirements & design … but still a great ROI

- Are peer reviews finding half your bugs?
  - Are you spreading them out or bunching them together?
  - If they’re not finding bugs, consider improving review culture
Best Practices For Key Software Metrics

- 2-3 hours of validation for each 1 hour of development
  - Head count ratio generally 1 Tester to 1 Developer
  - About 5% of effort for SQA

- Code productivity of about 1 to 3 lines per hour
  - At or above 3 lines/hr, you probably are cutting corners

- Peer reviews should find 50% (or more) of defects
  - At about 5%-10% of total project cost

- Metric Pitfalls
  - Use only metrics that provide value – don’t go crazy with metrics!
  - Gaming the metric doesn’t improve software quality
  - Reward/punish based on metric values will render metric useless
It would be a pure function if not for the side effects on your sanity

GOOD
FAST
CHEAP
(Pick Any Two)
What were things like in the old days? I hear you had to... compile things for different processors?

Yeah.

To compile your code, you had to mail it to IBM. Took 4-6 weeks.

Before garbage collection, data would pile up until the computer got full and you had to throw it away.

Early compilers could handle code fine, but comments had to be written in assembly.

C could only be written on punch cards. You had to pick a compact font, or you'd only fit a few characters per card.

C++ was big because it supported floppy disks. It still just punched holes in them, but it was a start.

Woo.

https://xkcd.com/1755/