“There is no code so big, twisted, or complex that maintenance can't make it worse.”

- Gerald M. Weinberg
Just put the technical debt on my credit card

Moving Fast and Breaking Things
Fragile Development Guide

@ThePracticalDev

© 2017 Philip Koopman
Software Maintenance

- **Anti-Patterns:**
  - Informal bug tracking
  - Not allocating post-release staffing
    - Bad prior release distracts team
  - Not paying off technical debt

- **Code maintenance during and after development**
  - You need a process to identify bugs and track to resolution
  - Most software is an update, not a clean-slate project
  - Ongoing effort is required to repay “technical debt”
Managing Bugs

- Map reported issue to an actual bug
  - L1/L2/L3 support to capture bug report
  - Sorting out duplicate reports takes effort

- Prioritize the bug fix (e.g., risk table)
  - Combination of frequency, business cost

- Find someone with right skills to fix it
  - Does this derail new development tasks?
  - Quick and dirty? Or a solid re-engineer fix?

- Validate the fix
  - Did you inject a new fault with the fix?

- Package the fix and deploy it
  - Hot patch? Defer to future schedule release?

- Risk table example:
  - High consequence defect
  - With low probability of occurrence
    → Medium risk / medium priority bug
Most SW work is on existing code, not a clean slate
- “Clean slate” often works with COTS components

60/60 rule [Glass, IEEE Software May 2001]
- Maintenance can average 60% of lifecycle cost
- About 60% of maintenance is adding new features

Maintenance is harder than development
- Need to understand existing system
  - Motivation for keeping entire V document chain up to date
  - Optimized code is more painful to maintain
- Need to modify system without breaking things
  - Complete rewrite usually impractical – and might be worse

Maintenance Matters Most

https://goo.gl/1CqN9i
Technical debt: messy code/design/architecture that hasn’t been cleaned up

- Some signs of debt:
  - Degraded code quality (spaghetti code, globals, warnings, …)
  - Skipped process steps (missing peer reviews, unit tests, …)
  - High fault reinjection ratio (new bugs when fixing old bugs)

- You incur debt by taking a shortcut
  - Short-term debt can be useful (e.g., meet a deadline)

- Repay debt by refactoring the system

Technical debt incurs interest

- Shortcuts often lead to bugs, fragility
- Accumulated debt becomes unsustainable

Use the right amount of debt

- It’s like using a credit card responsibly
- Devote part of each development cycle to repaying technical debt

Managing Technical Debt

https://goo.gl/cFXrD9
Best Practices for Maintenance

- Most development is maintenance
  - Plan for and staff maintenance
    - Most development is on the next revision
    - Plan for high priority emergency fixes
  - Keep up with technical debt payments

- Maintenance pitfalls
  - Not allocating time for bugs, maintenance & technical debt
    - For example, need perhaps 10% budget for technical debt repayment
    - Leave slack in deadlines for fixing urgent previous-version bugs
  - Evaluating programmers only for clean-sheet development skills

https://goo.gl/DDZfcY
CHANGES IN VERSION 10.17:
THE CPU NO LONGER OVERHEATS
WHEN YOU HOLD DOWN SPACEBAR.

COMMENTS:

LONGTIME_USER4 WRITES:
THIS UPDATE BROKE MY WORKFLOW!
MY CONTROL KEY IS HARD TO REACH,
SO I HOLD SPACEBAR INSTEAD, AND I
CONFIGURED EMACS TO INTERPRET A
RAPID TEMPERATURE RISE AS "CONTROL".

ADMIN WRITES:
THAT'S HORRIFYING.

LONGTIME_USER4 WRITES:
LOOK, MY SETUP WORKS FOR ME.
JUST ADD AN OPTION TO REENABLE
SPACEBAR HEATING.

EVERY CHANGE BREAKS SOMEONE'S WORKFLOW.

---

EVERY NOW AND THEN I REALIZE I'M MAINTAINING A
HUGE CHAIN OF TECHNOLOGY SOLELY TO SUPPORT ITSELF.