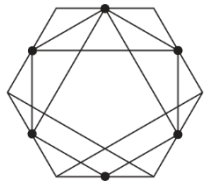
THE COMPANY KILLED BY THE CODE (ACCORDING TO UNCLE BOB)

- 1988 Sword Inc.
- Commercially successful C-debugger.
- Many companies moved to C++
- Sward Inc. Promised C++ debuger

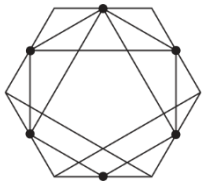
C Debugger



THE COMPANY KILLED BY THE CODE (ACCORDING TO UNCLE BOB)

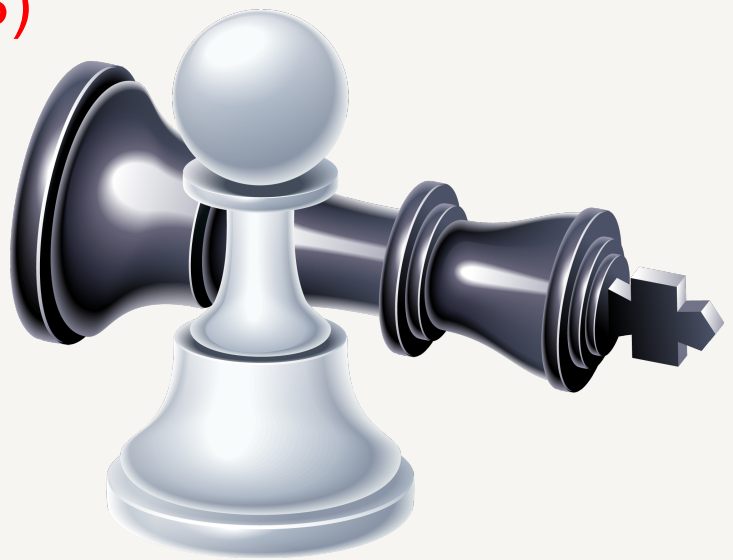
- 1990 Sword Inc delivered a C++ debugger
- Loading time: 45 Min, debugger crashed
- Release 2: six months later
- Still with the same bug

C++ Debugger



THE COMPANY KILLED BY THE CODE (ACCORDING TO UNCLE BOB)

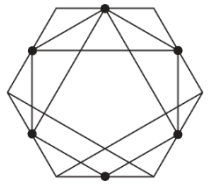
- Most customers canelled their contracts
- Some even sued Sword Inc.
- Sword Inc.



INTERVIEW WITH A FORMER SWORD INC. EMPLOYEE

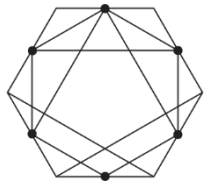
We rushed…

… and made a huge mess in the code

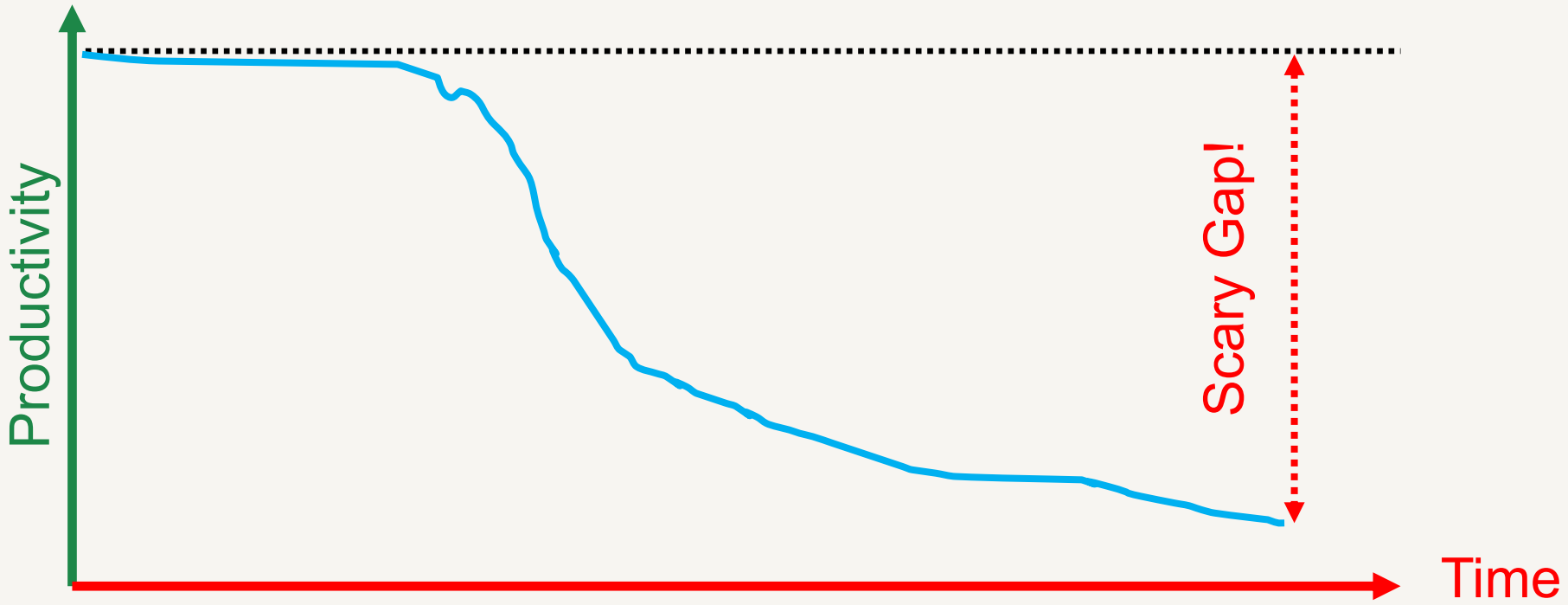


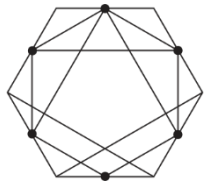
WHAT HAPPENED?

- ❑ Marketing: We need to be the first!
- ❑ Continuous bug fixing, high-priority features
- ❑ No time for refactoring, clean up
- ❑ Lots of work with little to show for it
- ❑ Code that can‘t even parse C++ with no way to fix it



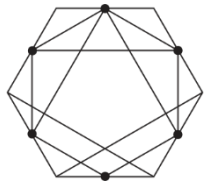
THE PRODUCTIVITY TRAP





THE PRODUCTIVITY TRAP

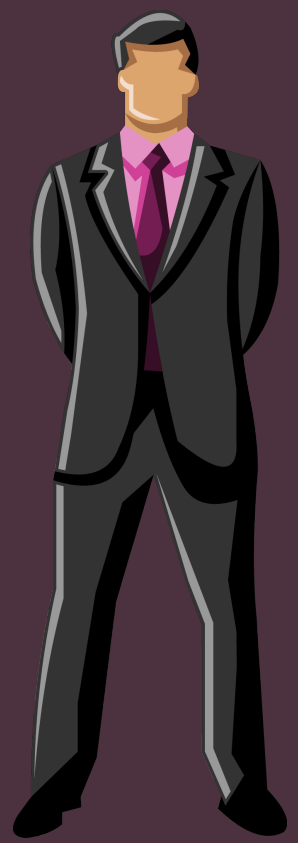
- Developers
  - Can initially make fast progress
  - Get slower over time
- Manager
  - Plan with initial velocity
  - Promise unrealistic results and completion times



THE PRODUCTIVITY TRAP

- Developers
  - Need to work faster and faster, and longer and longer hours
  - Have no time for refactoring
- Code base
  - Becomes rigid, fragile, inseparable, opaque
- Everybody
  - Is unhappy

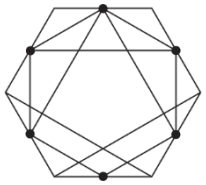
WHAT CAN MANAGEMENT DO?



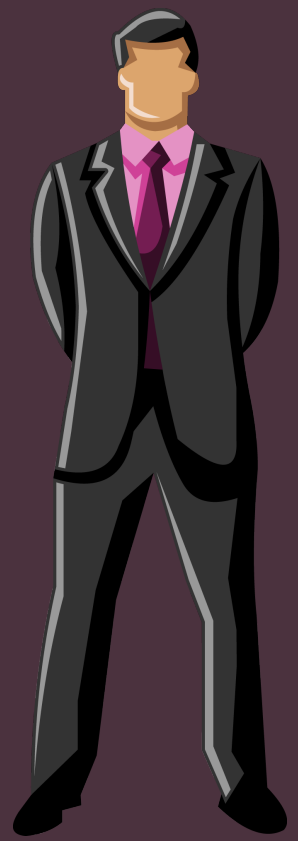
Initial Problem

Developers became slower and slower without any apparent reason.

What to do?

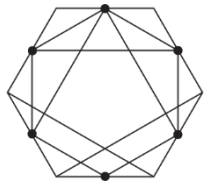


WHAT CAN MANAGEMENT DO?



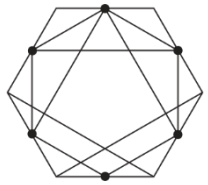
1) Remind developers about their responsibilities?
2) Introduce better working model?
3) Talk to developers to motivate them?
4) Shorten deadlines and order overtime?
5) Threaten to fire developers?

None of these solutions will solve the problem!



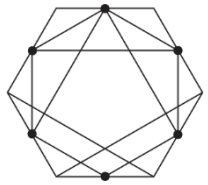
STRATEGIES TO RESOLVE THIS CRISIS

- Add more people
  - Brook’s law: Adding people to a late software project makes it later
- Big redesign
  - Stop development and spend some months/years cleaning up?
  - How do we know that this doesn’t break the code?



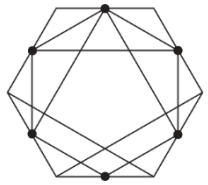
STRATEGIES TO RESOLVE THIS CRISIS

- Restart from scratch
  - Tiger team with the best developers
  - Now we have two competing teams and software versions
- Small, incremental redesign
  - Boy scout rule: slowly improve the code
  - Write new code using TDD and clean code

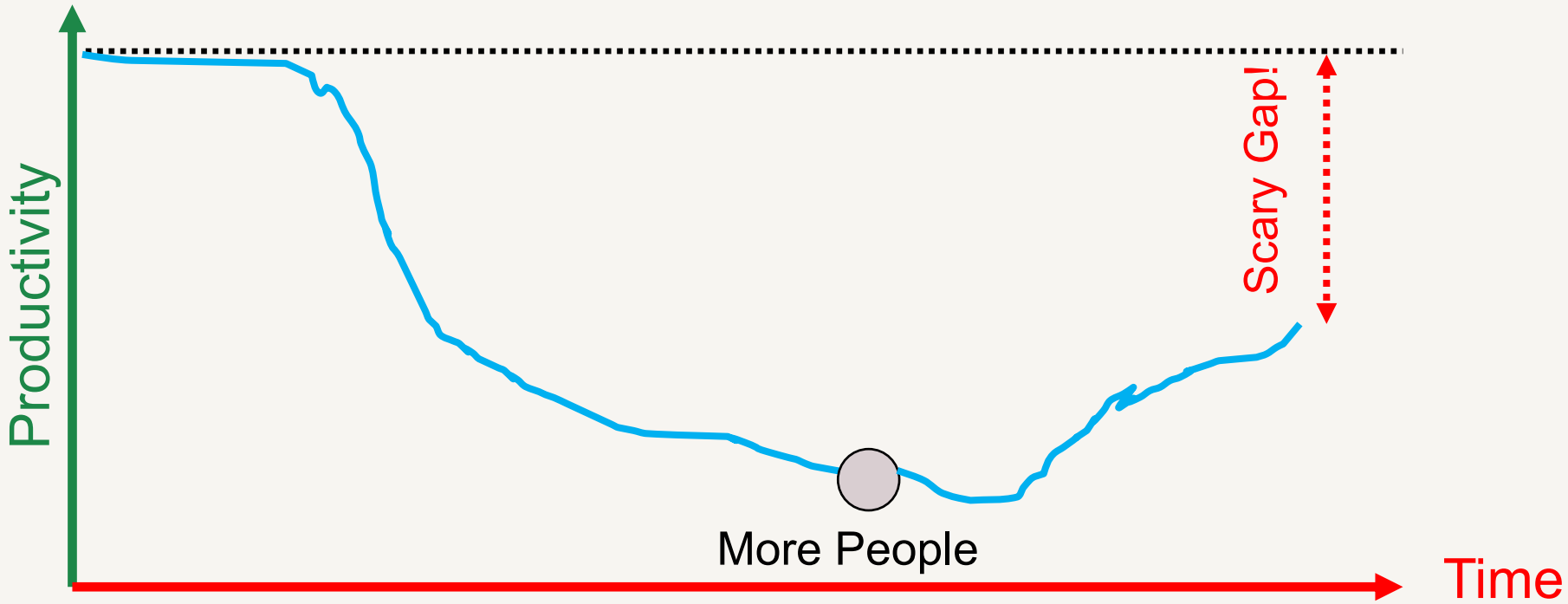


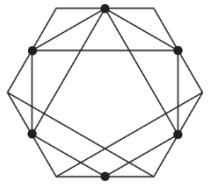
Krisen Strategien

- **Mehr Personal**
  - Ist mehr Personal = mehr Produktivität?
  - Brooks's Law: Adding manpower to a late software project make it later!
- **Big Redesign**
  - Entwicklungsstop und Clean-up für Wochen oder Monate?
  - Sehr gefährlich ohne Test Suite! Jede Änderung kann den Code brechen!
- **Neuer Anfang**
  - Tiger-Team von den besten Entwicklern alles neu machen?
  - Jahrelang zwei konkurrierende Teams und zwei SW Versionen!
- **Small Redesign**
  - Scout Rule: Den Code ein Tick besser einschicken als ausschicken
  - Neue Module mit Test, TDD und Clean Code Regeln entwickeln

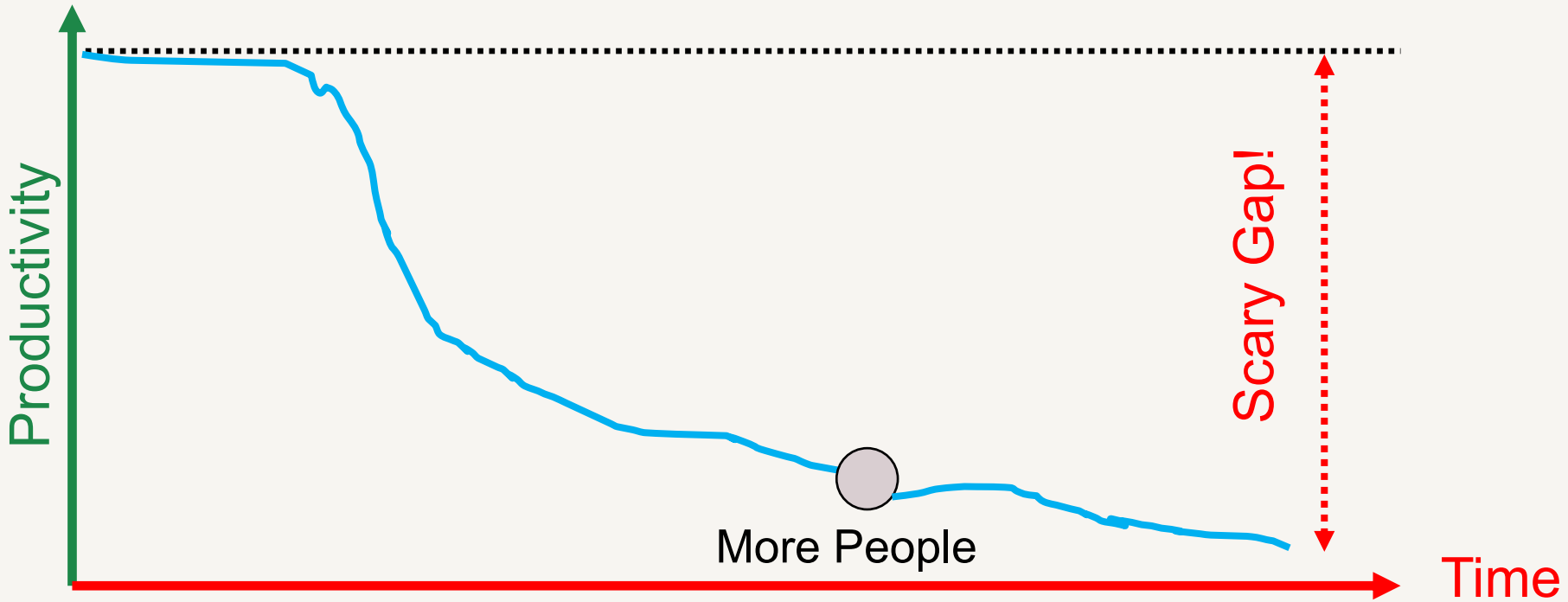


MORE PEOPLE: THEORY





MORE PEOPLE: REALITY



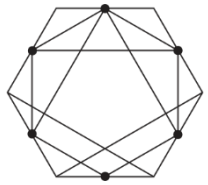
WHY DO WE WRITE BAD CODE?

- Stupid managers
- Impatient customers?
- Impossible timeframes?

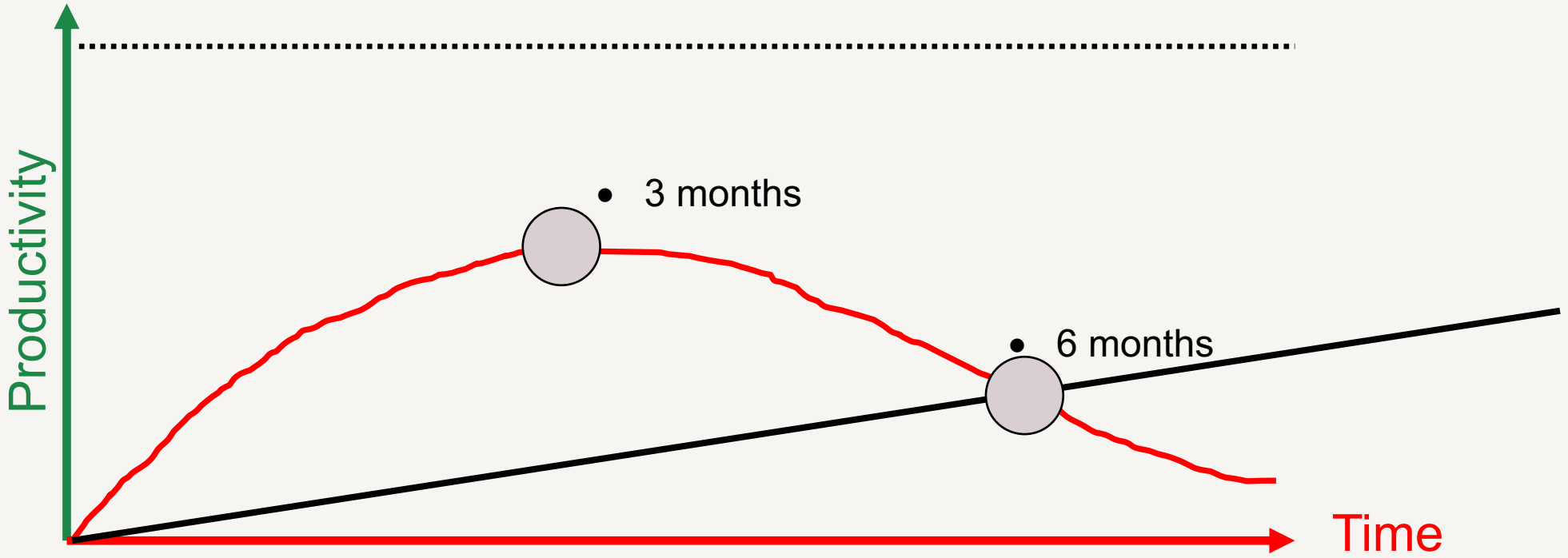
- These are (according to Uncle Bob) only pretexts.

WHY DO WE WRITE BAD CODE?

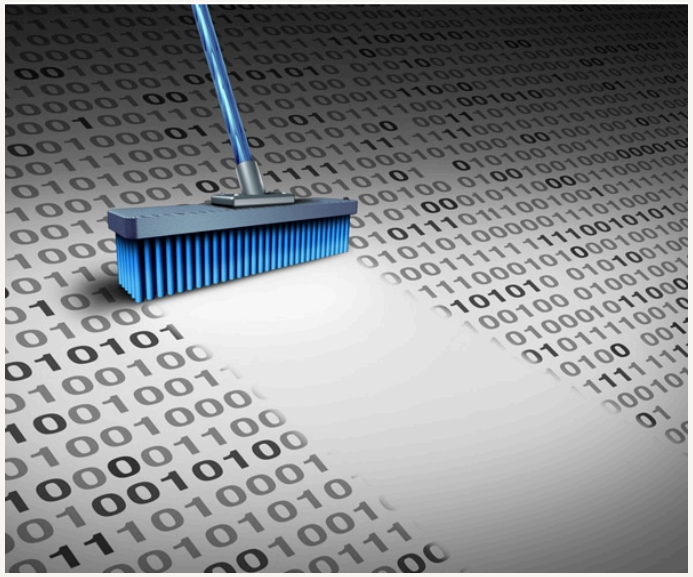
- Our hands are on the keyboard
- We are responsible for code rot
- Nobody else!
- Whenever you rush, you make a mess!



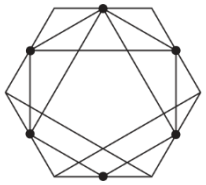
MORE PEOPLE: PRACTICE



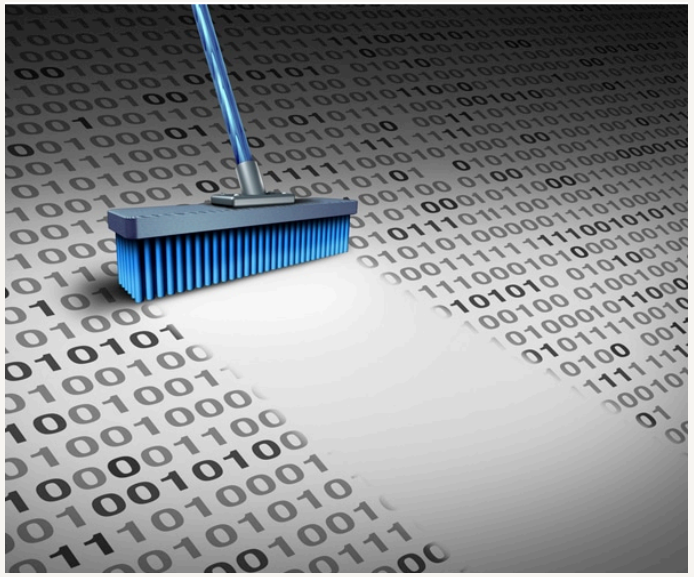
HOW TO DEAL WITH BAD CODE



- Don’t throw it away, improve it step-by-step
- As soon as you see bad code, clean it!
- Later = never.
- Leave the campground cleaner then you found it!
- Leave the world better than you found it!



IF EVERYBODY IMPROVES THE CODE A LITTLE BIT...



- No need for a big redesign! Steps that are doable.
- Introduce a better variable name
- Split a big function into smaller ones
- Remove duplicate code
- Remove some conditional statements

- If everybody leaves the code a little cleaner than they found it, it does not rot!