Implementation Issues

Topics

1) Programming is complex; how can we combat this?
2) Can we find bugs by reading each other's code?
3) Do different coding style help?
4) Can software reuse solve our problems?

Limiting Software Complexity

Limiting Software Complexity

- Writing software involves..

  (McConnel: Code Complete 2, 2004)
  - Developer must reason about..

- Beyond human competency
  - Humans cannot cope with these 10 orders of magnitude all at once.
  - An Analogy:
    think about a scientist trying to work
    with subatomic particles and galaxies
    in one calculation.

Analogy: not same orders of magnitude, but you get the idea.

Limiting Software Complexity

- (McConnel 2004)
  Software's Primary Technical Imperative:
  ..
  - We must simplify the problems in order to be able to think about them.
- Use encapsulation to reduce cognitive load
  - A good design allows you to..
  - A bad design requires you to work at low and high levels simultaneously, across multiple modules.

Complexity Example

- Compare the levels of abstraction in the following two competing interface designs to control SkyTrain:

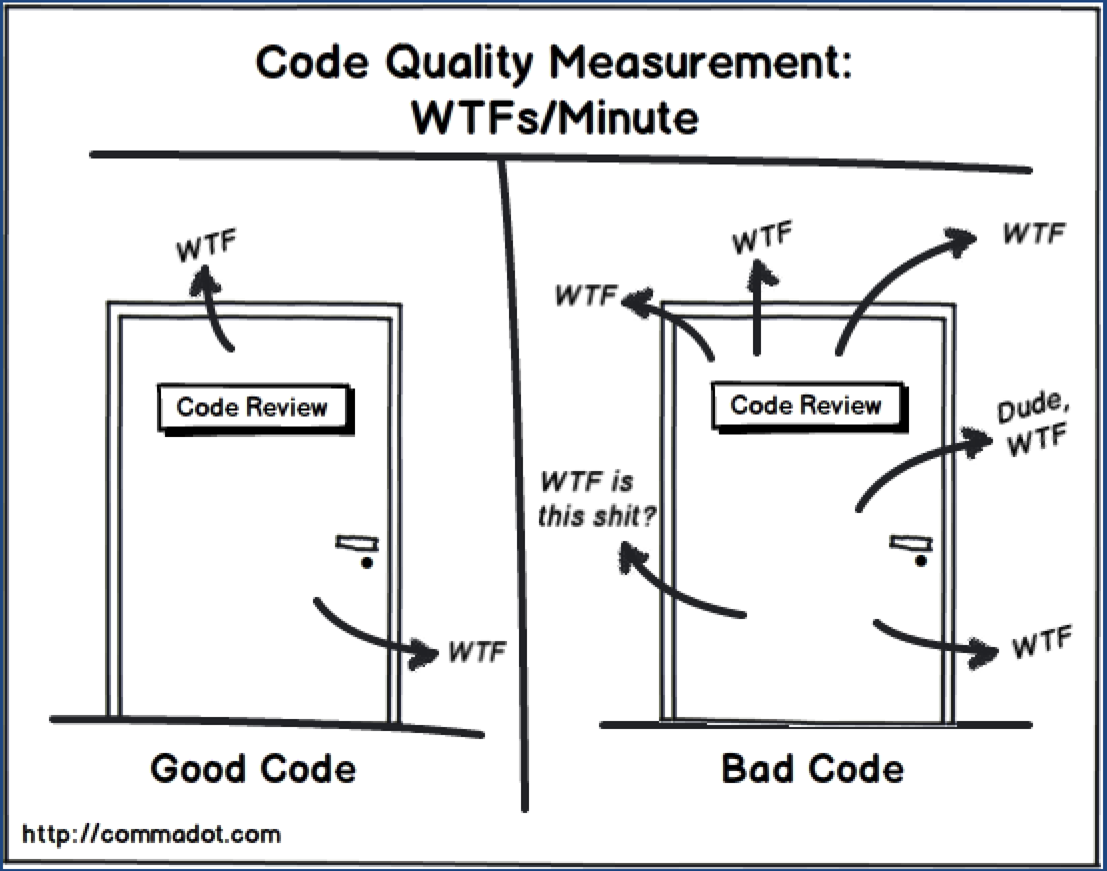
A

```
int isSpeedReadingValid();
long getSpeedSensorReading();
void setBrakeBits(long brakeBitMask);
void setMotorRPM(long rpm);
```

B

```
double getSpeedInMps();
void emergencyStop();

// May speed up or slow down
void accelerateToNewSpeedInMps(double speedInMps);
```



Code Reviews

Code Reviews

- A code review is having..
  - Could be a walk-through of the code by the author to show colleagues how code works
  - Usually it’s code reviewing a merge request (MR) or pull request (PR)
- Possible MR Code Review Process
  - Each MR reviewed by 1 (or more) developers
  - Add comments to the MR in GitLab
  - When ready, +1 it
  - Repo Manager accepts MR’s with +1

Code Review Tips

- ..
  - 200-400 lines of code (LoC)
  - < 60 minutes
- Use a checklist (next slide)
- Code review is positive!
  - ..
    (I had one MR with ~5 rounds of changes)
  - Knowing it will be code reviewed gives..
  - # defects found does not reflect badly on coder
- Fix bugs before MR accepted (in most cases)

Code Review Checklist

- During a code review look for
  - logic errors (logic backwards, missing else, ...)
  - poor error handling
  - poor security (buffer overrun)
  - poor readability/comments
  - common errors (== vs =, null ptr, memory leak)
  - requirements misunderstanding
- Can do a “code review” on design, test plans, test code, deployment scripts, etc.
  - Not just for shippable code.

Benefits of Code Reviews

- Code Review Effectiveness (Jones 1996, in McConnel 2004)
  - Informal code reviews catch.. of defects
  - Formal code reviews catch.. of defects
  - Unit testing catches.. of defects
- Code reviews benefits
  - Have a different person reading the code
    - Different way of thinking; validate requirements
  - Share knowledge between developers
    - Ex: suggest calling an existing function
  - Can suggest how to..
    whereas unit tests just test behaviour

Style Guide

Coding Style

- Coding is hard!
  - Developers must actively think about:
    - (design patterns, classes)
    - (algorithms)
    - (data types)
    - (spaces, naming, brackets)
- Syntactic concerns are often "religious" issues
  - Devs feel passionate about tab size (2, 3, 4, 8)
  - Not usually possible to “convert” someone to a new style without a lot of effort.

Code Style Example

- Linux kernel style guide:
  - Tabs are 8 characters, and thus indentations are also 8 characters. There are heretic movements that try to make indentations 4 (or even 2!) characters deep, and that is akin to trying to define the value of PI to be 3.

    (some text omitted...)
  - Now, some people will claim that having 8-character indentations makes the code move too far to the right, and makes it hard to read on a 80-character terminal screen. The answer to that is that if you need more than 3 levels of indentation, you're screwed anyway, and should fix your program.

    (some text omitted...)

Style Guide

- A style guide..
  - Consistent code style across project makes it faster to read and modify code.
  - Instead of syntactic disagreements, devs can think of..
- Can address some common issues in a language:

  -
    ```
    int x = 0;
    print(x?x++:++x);
    ```

  -
    ```
    int y = 100;
    if (y < 5 && y > 0 && y % 2 == 1) y--;
        y = 10;
    print(y);
    ```
- Example style guide (CMPT 213 w/ Java)

Code Reuse

Reuse cost

- Reusing well tested component can..

- But, it's not free
  - Must find and evaluate existing components.
  - Must spend time to integrate into new system.
- Reuse can cause errors
  - Some disasters caused by reusing software which had an unknown bug.
  - We tend not to test them well enough because..

Caution on reuse

https://www.youtube.com/watch?v=5tJPXYA0Nec

- Ariane 5 rocket: Initial test flight...
  - Reused a module from Ariane 4 which converted horizontal velocity (floating point) to a 16bit integer.
  - Ariane 4 rocket never encountered an error.
  - Exception handling was turned off for efficiency.
  - Both primary and backup computers encountered the error at the same time and shutdown.
  - *Code was only needed while on launch pad :(*

Caution on reuse

- Therac-25 (1982): Canadian made radiation therapy machine. ...
- Reused buggy software that relied on hardware safeties, which were left out in the later version.
- Actually numerous independent bugs; each was fatal
- ..

Idea behind one bug

```
unsigned char count = 1;
while (count != 0) {
        if (check_hardware_ready()) {
                count = 0;
        } else {
                count++;
                display_progress(count);
        }
}
turn_on_radiation();
```

Summary

- Primary technical imperative: manage complexity
- Code reviews effective complement unit testing for finding defects; improves code quality
- Use a style guide to free developer from syntactic decisions
  - Can instead focus on higher-level issues
- Consider possible reuse of existing software
  - Beware of over confidence.