Applied Software Engineering SIT725

T2 2024

Valeh Moghaddam

Lecture 4
Good Software- Software Engineering Ethics

Outline

- Good Software Ethics

➢Why we need?

➢Principles?

➢Software quality characteristics

➢Relation with the design concept

Good Software

Why do we write good software?

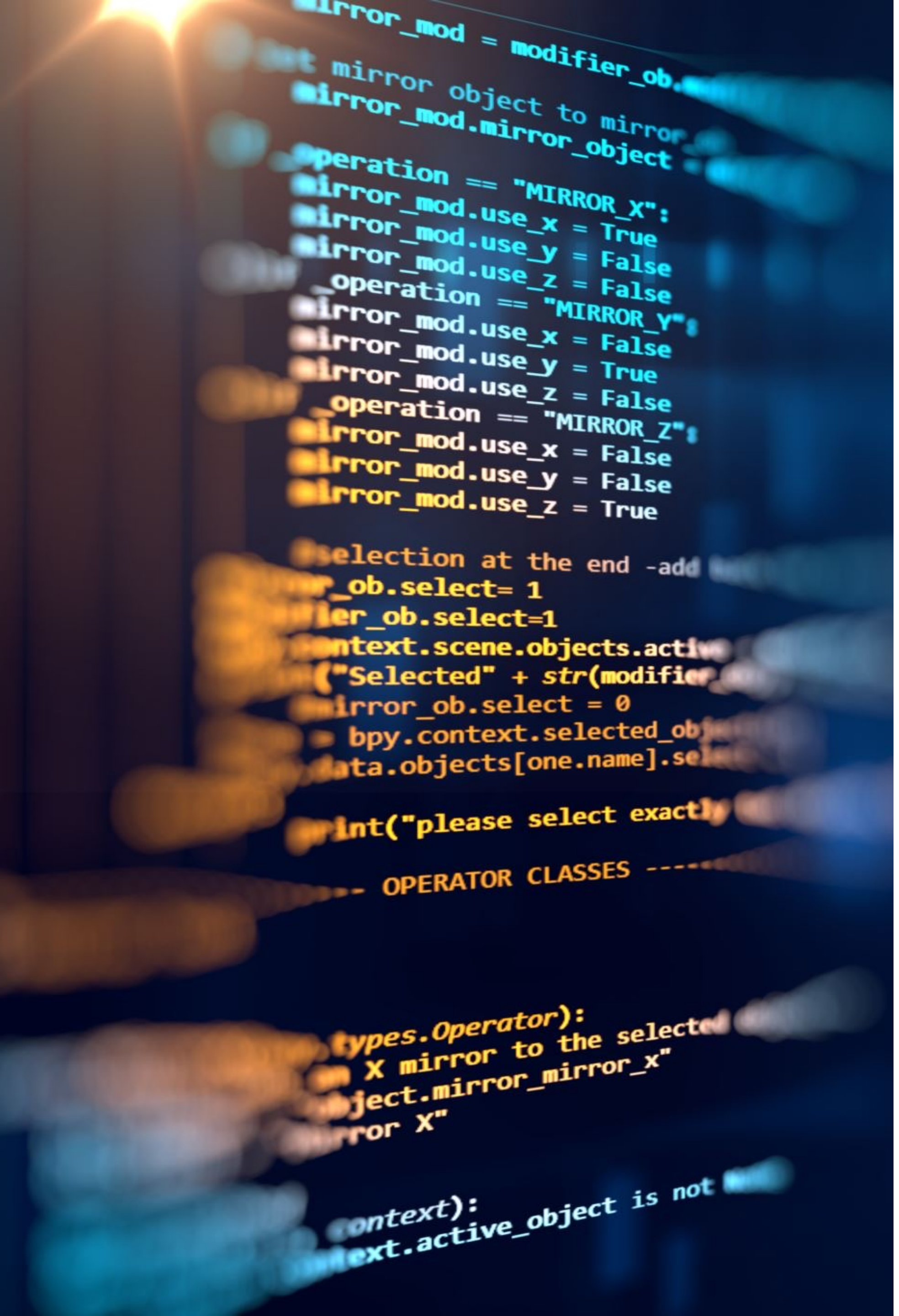
Prevalence of software in society provide significant opportunities to do good or cause harm.

WALLY, I DISCOVERED A DEADLY SAFETY FLAW IN OUR PRODUCT. WHO SHOULD I INFORM?
NO ONE. THE STOCK WOULD PLUNGE AND WE'D HAVE MASSIVE LAYOFFS. YOUR CAREER WOULD BE RUINED.
BUT MY NEGLIGENCE COULD CAUSE THE DEATHS OF A DOZEN CUSTOMERS.
THE FIRST DOZEN IS ALWAYS THE HARDEST.
www.dilbert.com
scottadams@aol.com
© 2004 Scott Adams, Inc./Dist. by UFS, Inc.

- As a software engineer, what you do matters, wrong code doesn't mean "software crash"

- **It could mean**
- Plane crashes
- Car doesn't brake
- Life Sustaining system fails

- Can you recall any of these events in the past?



Good Software

- *WHAT DRIVES US?*

Software Engineers are those who contribute by direct participation or by teaching, to the analysis, specification, design, development, certification and testing of software Systems.

- **8 key principles of Software Engineering Ethics**

➢ Public
➢ Client and Employer
➢ Product
➢ Judgment
➢ Management
➢ Profession
➢ Collogues
➢ Self

Good Software

- **Principle 1: Public**

Software engineers shall act consistently with the public interest.

USE CASE

- Idea behind making such software is for public welfare.
- Develop the software which is safe ,meet the standards ,and passes all tests.
- Software should not decrease the quality of life and privacy as it provides high level of safety as well as security

Good Software

- **Principle 2: Client and Employer**

Software engineers shall act in a manner that is in the best interests of their client and employer, consistent with the public interest.

USE CASE

- All resources required for the development of the software is authentically approved from client or employer.
- Software engineers should be informative.
- Honor confidentiality of information (private information of client or employer should not be compromised) .

Good Software

- **Principle 3: Product**

- Software engineers shall maintain integrity and independence in their professional judgment.

USE CASE

- Understand specifications fully by applying certain methodologies such as feasibility study.
- Ensure adequate testing, debugging and review of software related documents.
- Software should be up to the standard.
- Software scope should be well defined

Good Software

- **Principle 4: Judgment**
- Software engineers shall maintain integrity and independence in their professional judgment.

USE CASE

- Maintain integrity and independency .
- Avoid conflicting financial interests like bribery, bauble billing .
- Temper technology by keeping ethics in mind.

Good Software

- **Principle 5: Management**
- Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.

USE CASE

- Ensure good management for any project on which they work, including effective procedures for promotion of quality and reduction of risks.
- Software engineers should be informed about standards.
- Software engineers should know the employer's policies and procedure.

Good Software

- **Principle 6: Profession**

- Software engineers shall advance the integrity and reputation of the profession consistent with the public interest

USE CASE

- Extend software engineering knowledge by participation in professional meetings, organizations and publications.
- Software engineers should be accurate about the characteristics of software on which they are working.
- Avoid association with businesses and organizations which conflict with the system.
- Act responsibly for detecting, correcting and reporting errors in software

Good Software

- **Principle 7: Colleagues**

- Software engineers should be fair and supportive of their colleagues

USE CASE

- Assist colleagues in professional development .
- Give a fair hearing to the opinion, concerns or complaints of a colleague.
- Encourage colleagues to adhere to this code.
- Review the work of others in an objective and properly documented way.
- In situation outside of their own area of competence , call other professionals who have competence in that area.

Good Software

- **Principle 8: Self**

Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

Use Case

- Further their knowledge of development in the analysis , design , development , maintenance and testing of software and related documents.
- Improve their ability to create safe , reliable and quality software at reasonable cost within reasonable time.
- Improve their ability to produce accurate and well-written documentation.
- Not influence others to undertake any action that involve a breach of the code

Good Software

Software Quality Characteristics

Good Software

- **How to define good software? Software Quality**

KEY CHARACTERISTICS

- Maintainability

  The ease with which changes can be made to satisfy new requirements or to correct deficiencies.
- Correctness

  The degree with which software adheres to its specified requirements
- Reusability

  The ease with which software can be reused in developing other software
- Reliability

  The frequency and criticality of software failure
- Portability

  The ease with which software can be used on computer configurations other than its current one
- Efficiency

  The degree with which software fulfills its purpose without waste of resources.
-

DEAKIN UNIVERSITY

Good Software

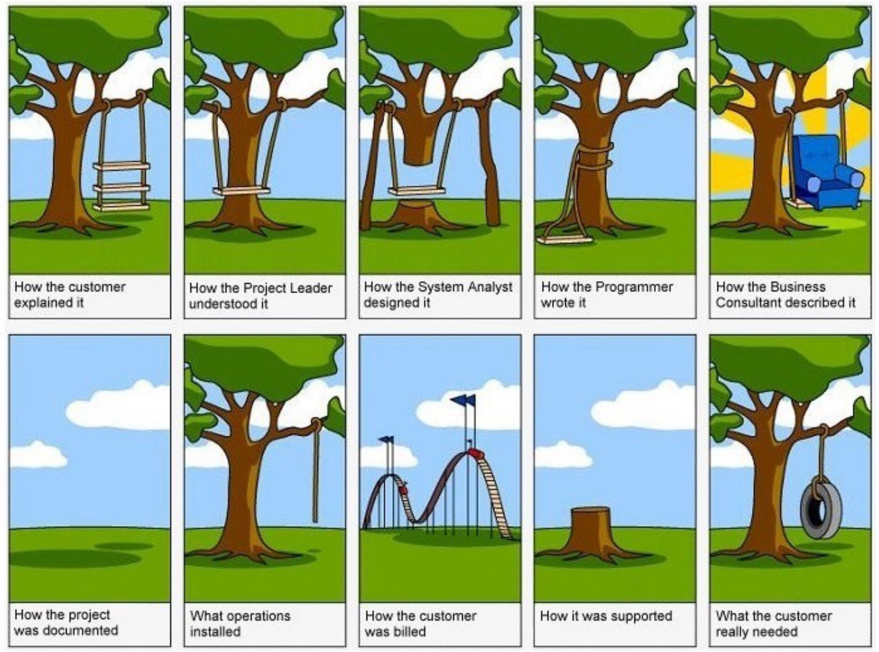
Maintainability

is "the ease with which changes can be made to satisfy new requirements or to correct deficiencies“.

Good Software

Correctness

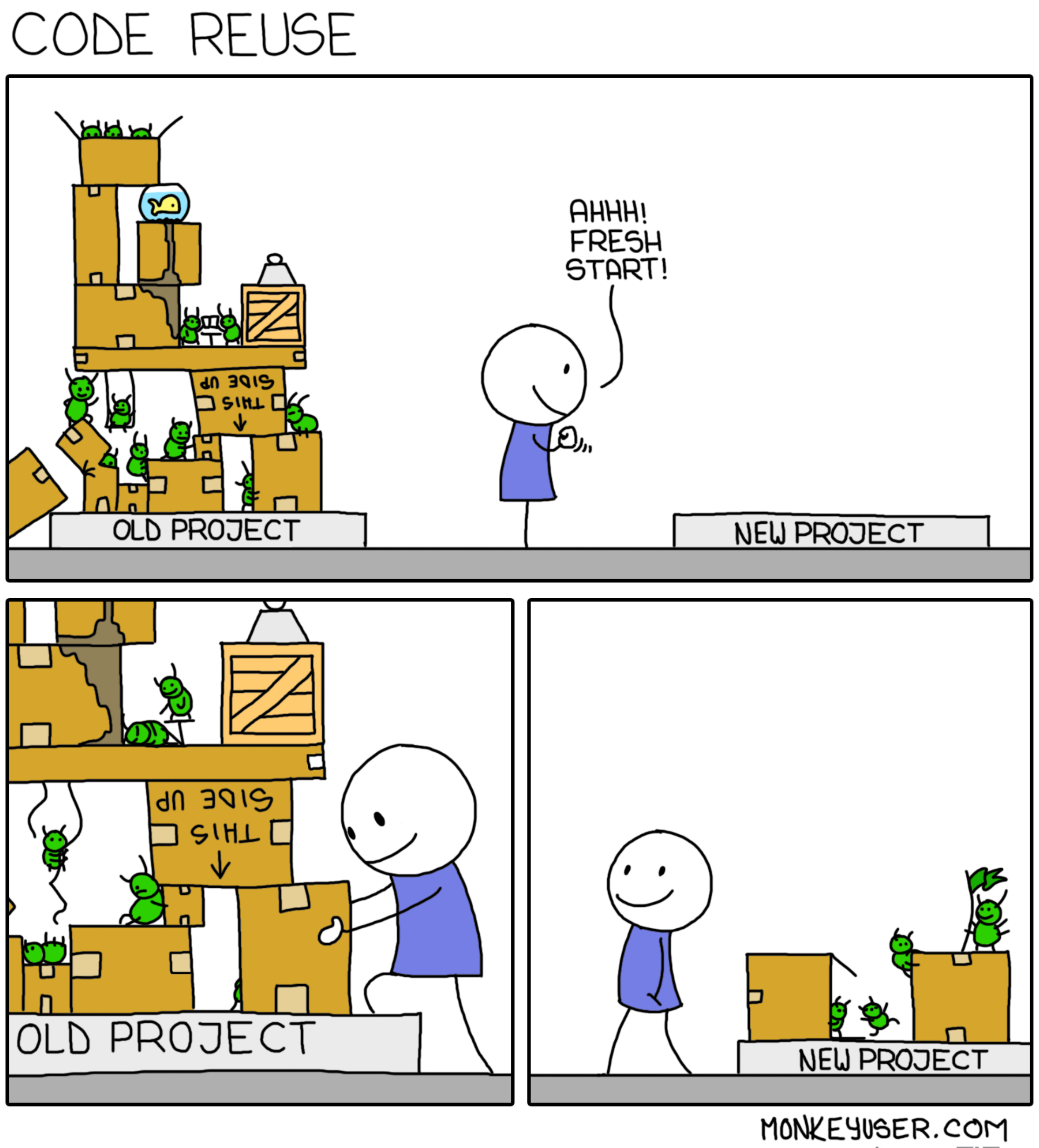
is "the degree with which software adheres to its specified requirements".


How the customer explained it
How the Project Leader understood it
How the System Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Good Software

Reusability

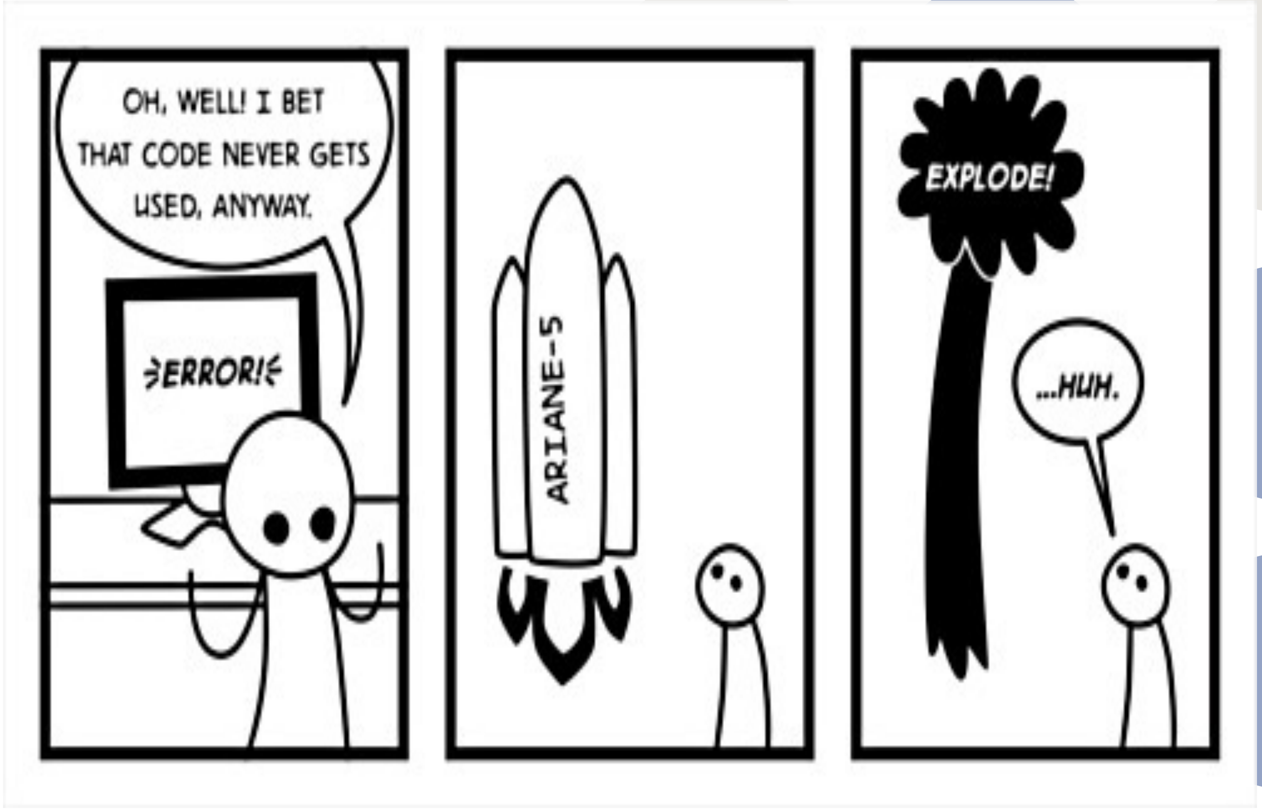
- is "the ease with which software can be reused in developing other software" [Balci 1997].
- By reusing existing software, developers can create more complex software in a shorter amount of time. Reuse is already a common technique employed in other engineering disciplines. For example, when a house is constructed, the trusses which support the roof are typically purchased preassembled.



Good Software

Reliability

is "the frequency and criticality of software failure, where failure is an unacceptable effect or behavior occurring under permissible operating conditions" [Balci 1997]. The frequency of software failure is measured by the average time between failures.



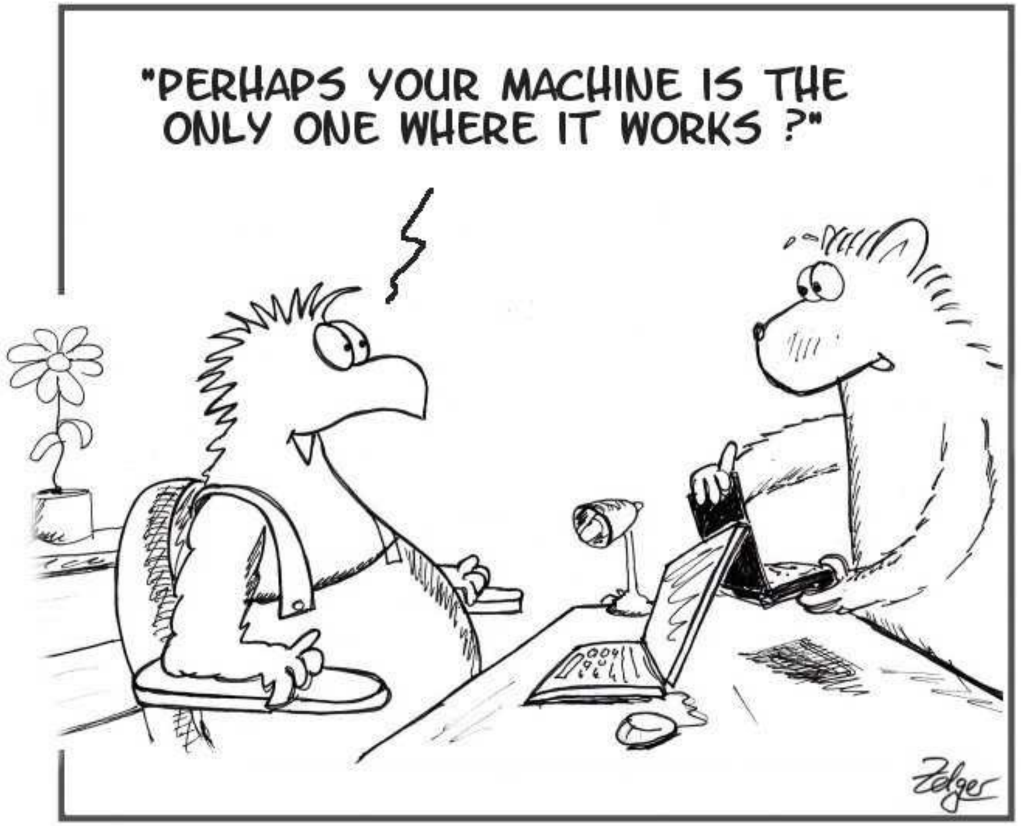
https://www.csfieldguide.org.nz/en/chapters/softwareengineering/

DEAKIN UNIVERSITY

Good Software

- **Portability**

is "the ease with which software can be used on computer configurations other than its current one" [Balci 1997]. Porting software to other computer configurations is important for several reasons.



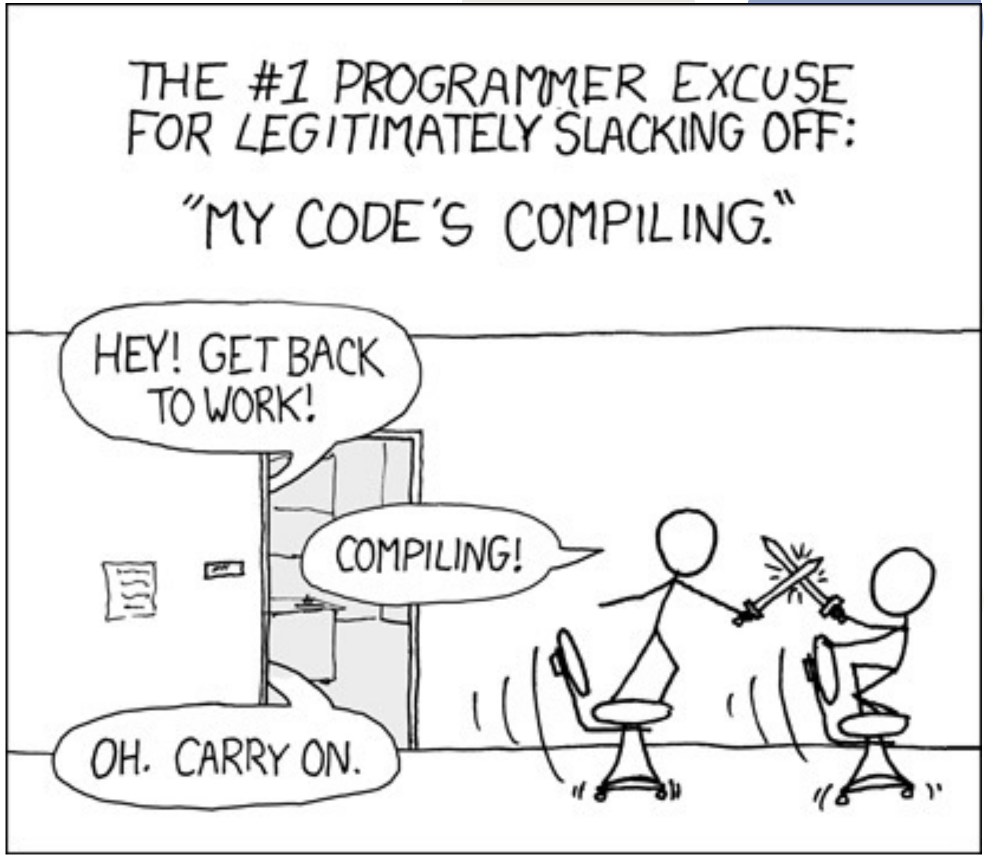
It works on my machine

https://www.architect.io/blog/2020-03-24/the-importance-of-portability/

Good Software

- **Efficiency**

is "the degree with which software fulfills its purpose without waste of resources" [Balci 1997]. Efficiency is really a multifaceted quality characteristic and must be assessed with respect to a particular resource such as execution time or storage space



XKCD Comics for Programmers

Design Concept

Good Software

What are the design concepts?

- Abstraction
- Refinement
- Architecture
- Modularity
- Information hiding
- Reflecting
- Structural partitioning

I'M BACK FROM TRAINING.
S. Adams E-mail: SCOTTADAMS@AOL.COM
I GOT A BIG BINDER.
THE TRAINING IS ALREADY FORGOTTEN, BUT THE BINDER WILL LAST FOREVER.
A LIVING MONUMENT TO TEMPORARY KNOWLEDGE!

Good Software

- **Abstraction**

Abstraction allows designers to focus on solving a problem without being concerned

about irrelevant lower level details.

- **Refinement**

a process of elaboration. We begin with a statement of function (or description of information)

that is defined at a high level of abstraction and reach at the lower level of abstraction. In each step (of the refinement), one or several instructions of the given program are

decomposed into more detailed instructions.

Good Software

Architecture

- *Structural models – architecture as organized collection of*
- *components*
- *Framework models – attempt to identify repeatable architectural*
- *patterns*
- *Dynamic models – indicate how program structure changes as a*
- *function of external events*
- *Process models – focus on the design of the business or technical*
- *process that system must accommodate*
- *Functional models – used to represent system functional hierarchy*

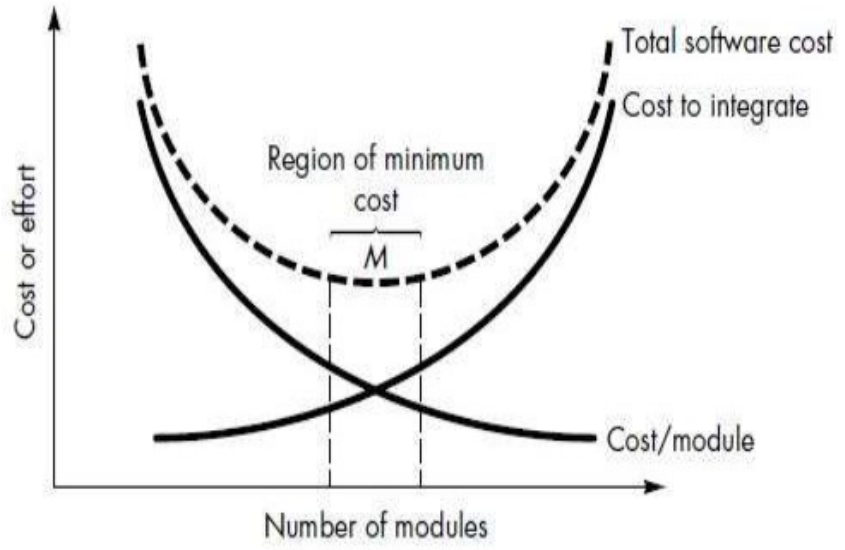
Good Software

- **Modularity**

"divide and conquer" conclusion—it's easier to solve a complex problem when you break it into manageable pieces.

It has been stated that "modularity is the single attribute of software that allows a program to be intellectually manageable“.

The effort (cost) to develop an individual software module does decrease as the total number of modules increases. Given the same set of requirements, more modules means smaller individual size. However, as the number of modules grows, the effort (cost) associated with integrating the modules also grows.



These characteristics lead to a total cost or effort curve shown in the figure. There is a number, M, of modules that would result in minimum development cost, but we do not have the necessary sophistication to predict M with assurance.

Good Software

- **Information Hiding**

Modules should be specified and designed so that information (procedure and data) contained within a module is inaccessible to other modules that have no need for such information.

Hiding implies that effective modularity can be achieved by defining a set of independent modules that communicate with one another only that information necessary to achieve software function.

OUR GREATEST STRENGTH IS OUR COMPANY CULTURE.
@ScottAdamsSays
Dilbert.com
WHAT'S OUR COMPANY CULTURE?
GIVE US A HINT.
© 2017 Scott Adams, Inc./Dist. by Andrews McMeel
4-25-17
WE'RE FIERCELY COMPETI-TIVE.
DO WE HIDE IT BECAUSE WE'RE ALSO MODEST?

Good Software

- **Refactoring**

Refactoring is a reorganization technique that simplifies the design (or internal code structure) of a component without changing its function or external behaviour.

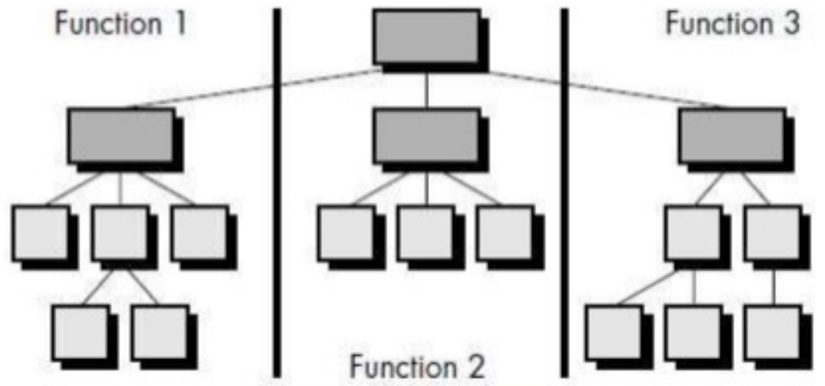
It removes redundancy, unused design elements, inefficient or unnecessary algorithms, poorly constructed or inappropriate data structures, or any other design failures

Good Software

- **Structural Partitioning**

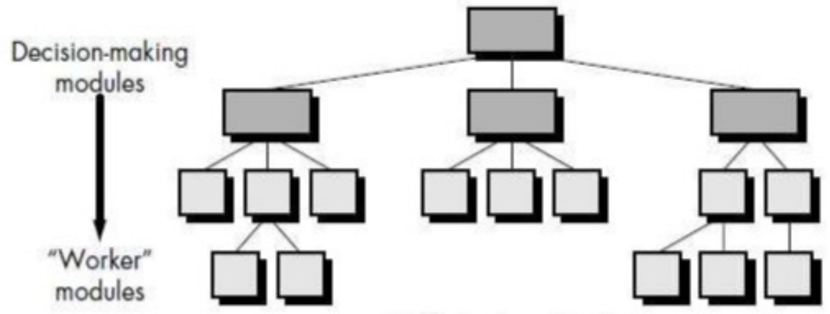
If the architectural style of a system is hierarchical, the program structure can be partitioned both horizontally and vertically.

Horizontal Partitioning
It defines separate branches of the modular hierarchy for each major program function.



(a) Horizontal partitioning

Vertical Partitioning
Also called factoring, suggests that control (decision making) and work should be distributed top-down in the program structure. Top level modules should perform control functions and do little actual processing work.



(b) Vertical partitioning

Summary

- Software Design Ethics

- Ethics Principles

- Software Quality Characteristics

Thank you

Question?

OnTrack check-in