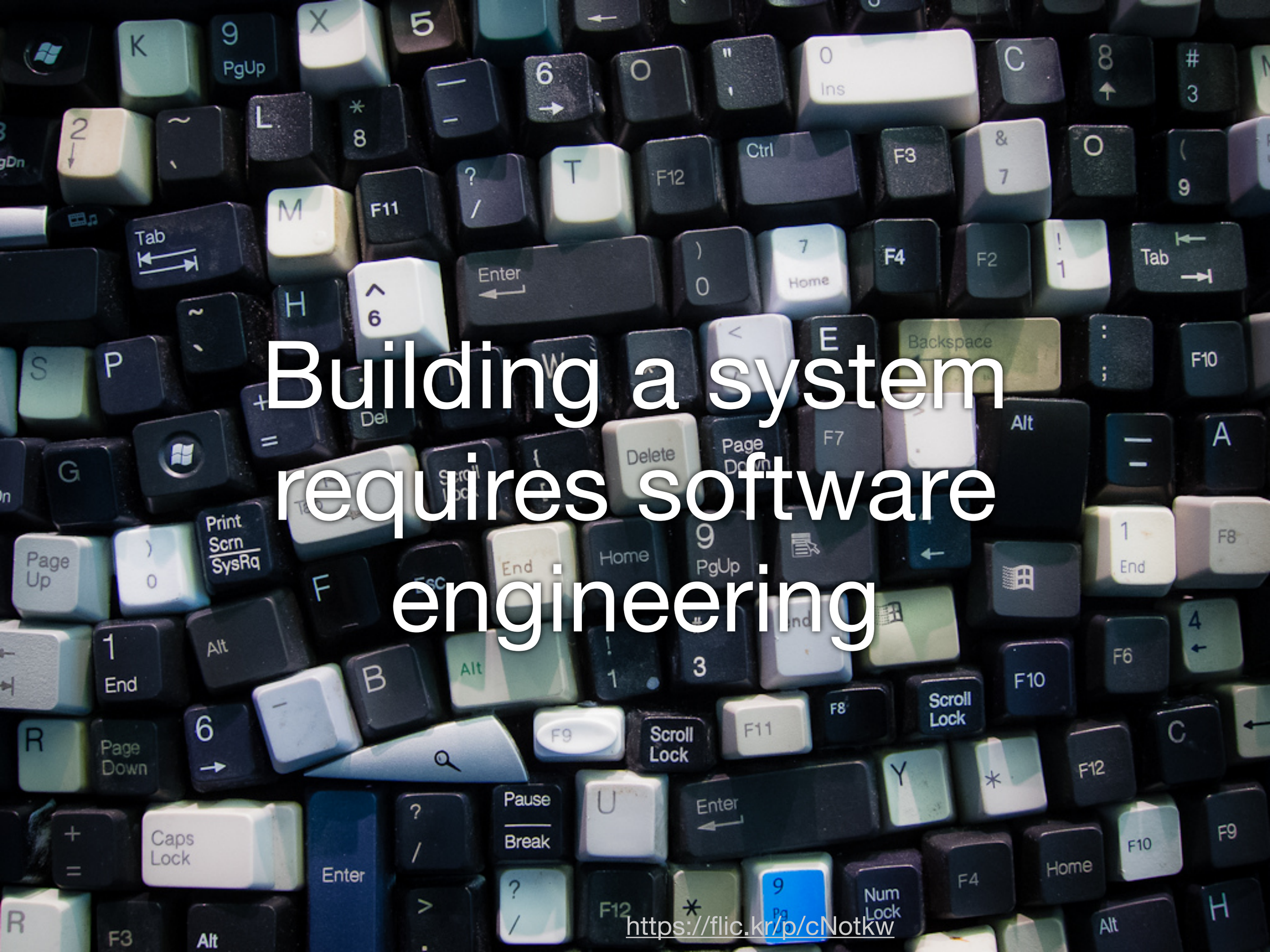
Engineering Software

Software Engineering
CS 130

Donald J. Patterson

Content adapted from Essentials of Software Engineering 3rd edition by Tsui, Karam, Bernal
Jones and Bartlett Learning

Building a system requires software engineering



https://flic.kr/p/cNotkw

Engineering Software

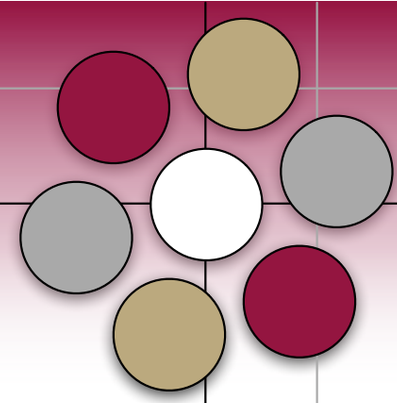
As size and complexity increased so did the failure rate

increases in size and complexity

increased effort due to size and complexity

increases in size and complexity

Engineering Software



“Engineering” Software was thought to be the cure

- Put some discipline into “programming” !
- Do more than just coding/programming!
- ‘Study’ (model/measure),
- ‘Understand’ (analyze),
- ‘Improve’ (change) this field!

Engineering Software

Chaos Report and Software Project Success/ Failures

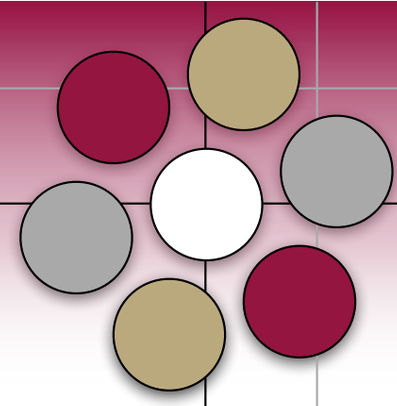
- Chaos Report (1995) sampled some 300 software projects and reported that only about 16% of those projects “completed,” “on-time,” and “within-budget” !

84% of projects failed

Chaos Report and Software Project Success/ Failures

- Chaos Report (2009) stated that software projects have improved with 32% “completed,” “on-time,” and “within-budget.”

68% of projects still failed



Software Project Success & Failure Factors (Chaos Report)

Attributes of projects that “succeeded”

- User Involvement
- Executive Management Support
- Clear Requirements
- Proper Planning

Engineering Software

Software Project Success & Failure Factors (Chaos Report)

Attributes of projects that were “challenged”*

- Lack of user input
- Incomplete user requirements and specification
- Changing requirements and specifications

*completed and operational but over-budget or over-time

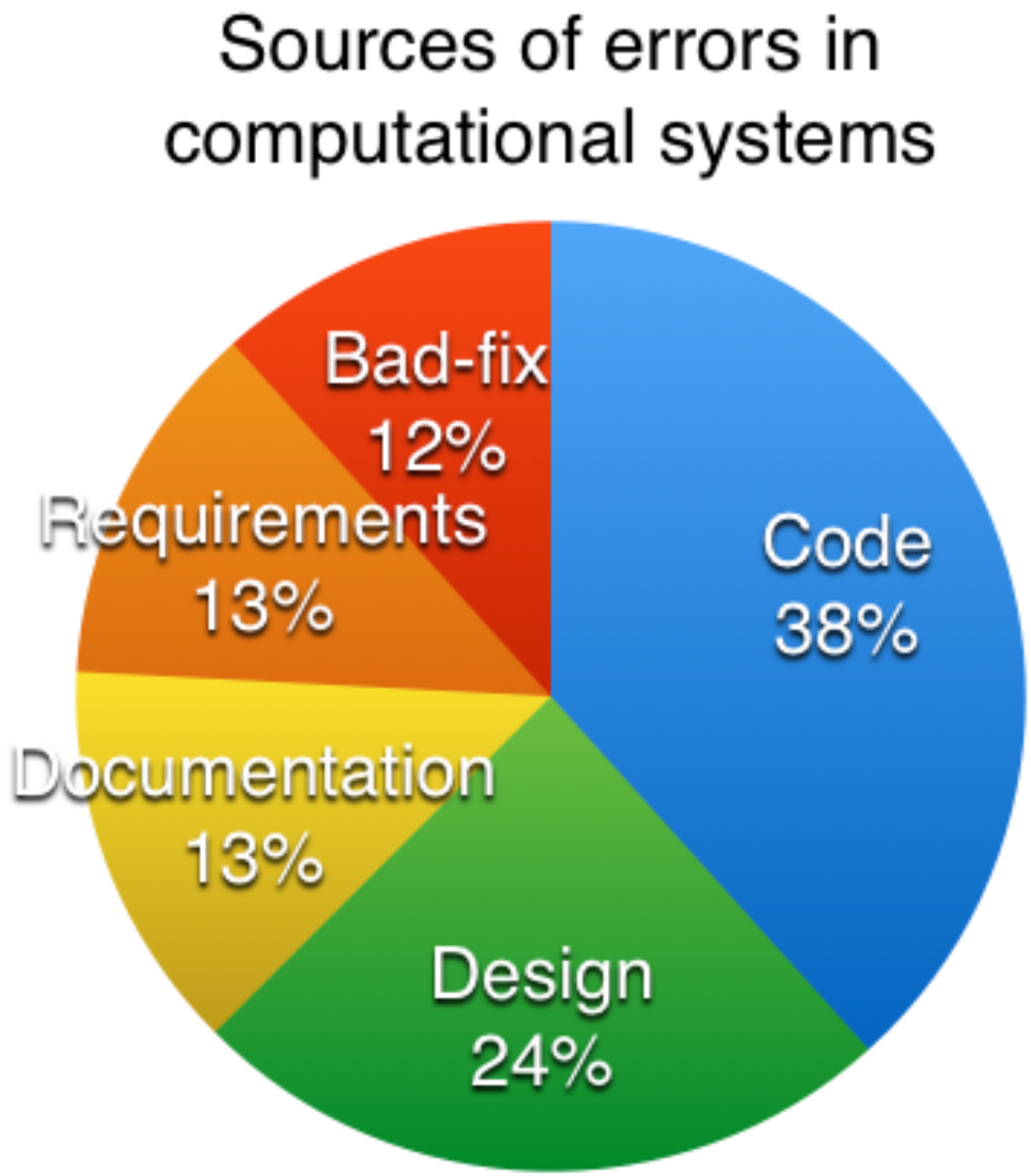
Engineering Software

Software Project Success & Failure Factors (Chaos Report)

Attributes of projects that were “impaired and ultimately cancelled”*

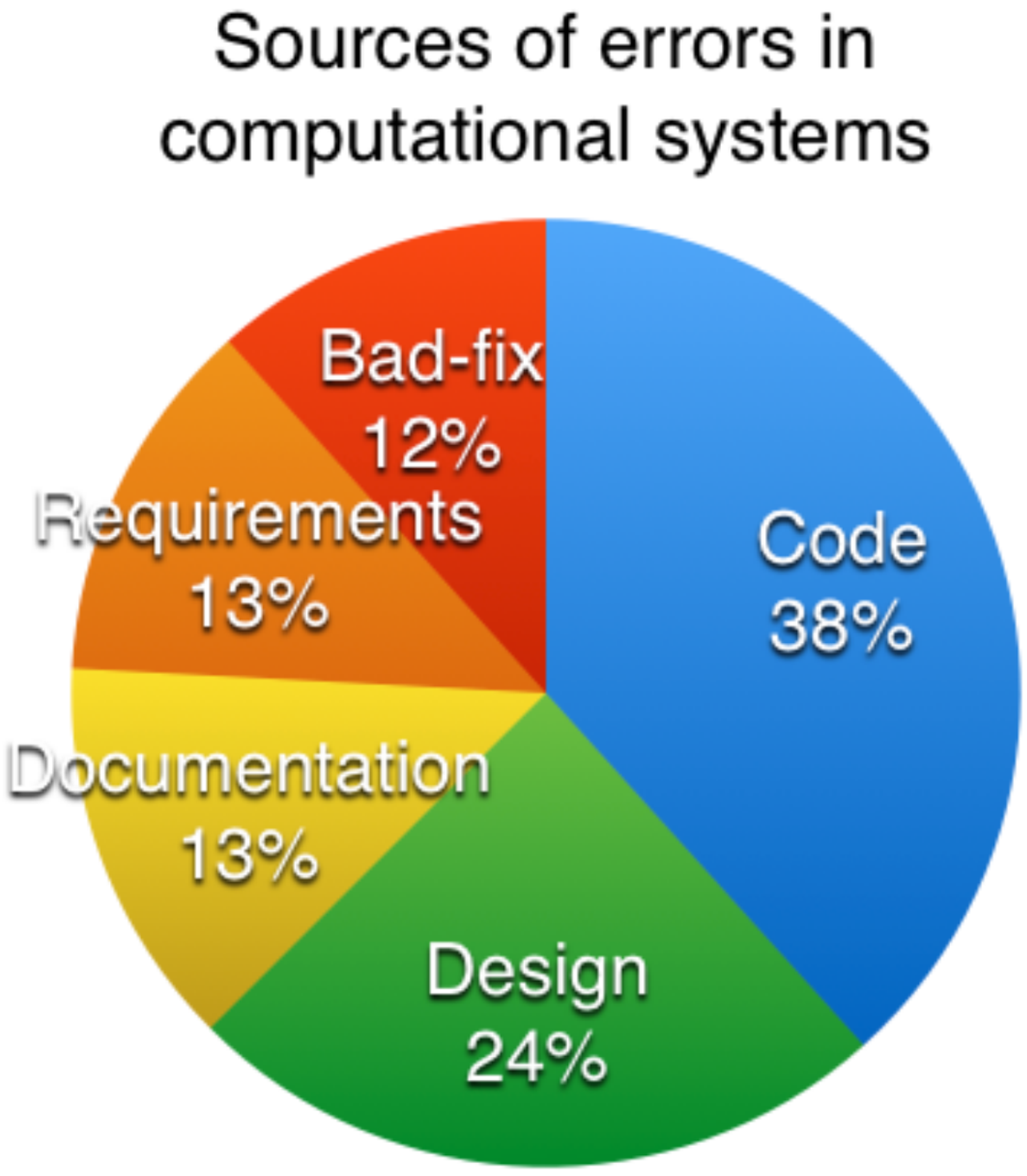
- Incomplete requirements
- Lack of user involvement
- Lack of resources

Software Product Failures (Capers Jones Study)



Engineering Software

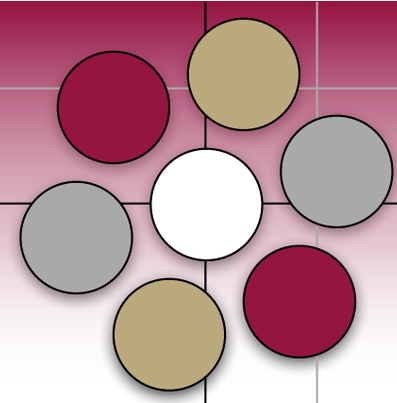
Software Product Failures (Capers Jones Study)



All errors can be serious and costly

Should we worry about coding or requirement errors more? why?

Why can requirement errors be so costly if not caught?



Coordination & Non-Technical Concerns

- As software projects grew in size and complexity, problems went beyond just code and software. Other “non-technical” issues became apparent:
  - Executive commitment and leadership
  - Thorough planning of both business and technical processes
  - Skilled and experienced work-force
  - Management focus and project monitoring
  - Willingness to make changes and adjustments

Engineering Software

US General Accounting Office Report to US Senate (2004)

- 3 “Key” strategies to ensuring delivery of:
  - high quality software
  - on-time and
  - within-budget:
- Focused attention on software development environment (people/tools/management/etc.)
- “Disciplined” development process
- Methodical use of metrics to gauge cost, schedule, and functional performance targets

Engineering Software

“Birth” of Software Engineering

- The early experiences of writing difficult but “small programs” did NOT provide us with the road map when we started to build “large” operating system, database, commercial system, etc.

Engineering Software

“Birth” of Software Engineering

- What is needed to develop large and complex software “products” and what is needed to control such projects ?
  - More “discipline” is needed in this field:
  - “SOFTWARE ENGINEERING”
    - (NATO conference - 1968)

Well that’s nice, but what is Software Engineering?

Engineering Software

What is Software Engineering?

- “multi-person construction of multi-version software”
  - David Parnas

Engineering Software

What is Software Engineering?

- “an engineering discipline whose focus is the cost-effective development of high quality software system”
  - Sommerville

Engineering Software

What is Software Engineering?

- application of computing tools to solving problems
  - Pfleeger

Engineering Software

What is Software Engineering?

- “form of engineering that applies the principles of computer science and mathematics to achieving cost-effective solutions to software problems”
- CMU/SEI-90-TR-003

Engineering Software

What is Software Engineering?

- “application of a systematic, disciplined, quantifiable approach to the i) development of, ii) operation of, and iii) maintenance of software”
- IEEE std 610-1990

MULTI-PERSON

MULTI-VERSION

→ COST CONTROL

QUALITY

→ REAL-WORLD PROBLEMS

REAL-WORLD USE

FORMAL TRAINING

LIFECYCLE

PLANNING

COMMUNICATION

PURPOSE-DRIVEN

→ PURSUIT OF COST-EFFECTIVE

BUSINESS NEEDS SIDE

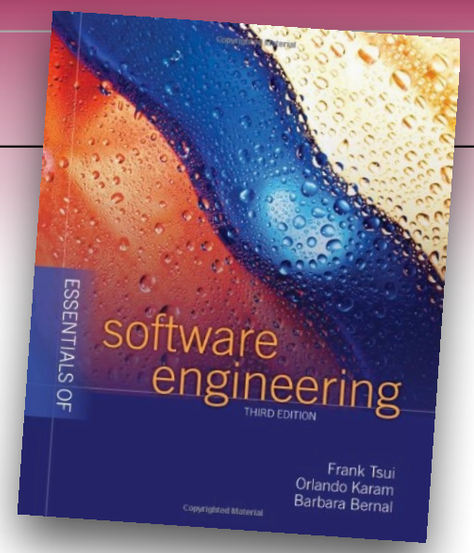
PURSUIT OF

COST-EFFECTIVE

TIME-EFFECTIVE

BUG-LESS

Engineering Software



What is Software Engineering?

- Software Engineering is a broad field that touches upon all aspects of developing and supporting a software system, spanning across the following key areas:
  - Technical and business processes
  - Specific methodologies and techniques
  - Product characterization and metrics for measurements
  - People skills and team work
  - Project coordination and management

Engineering Software

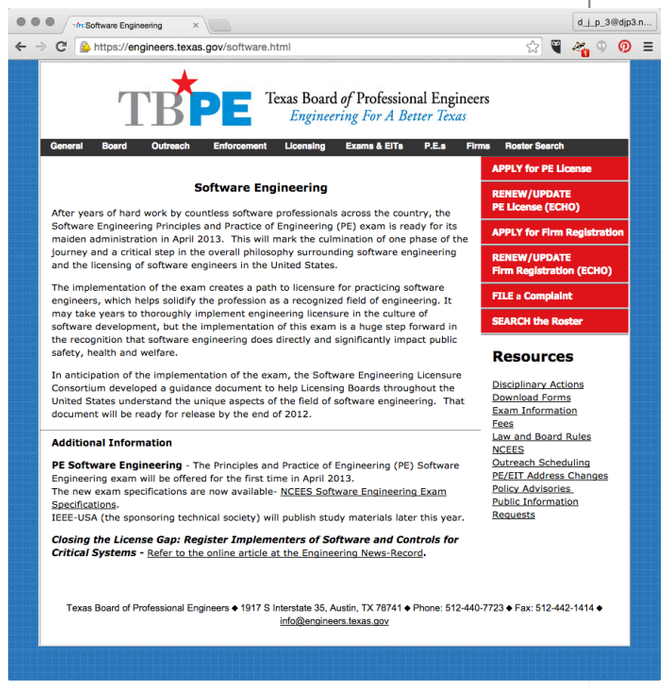
Relevancy of Software Engineering

- Software is a serious business
  - Reached $180 billion in 2000
  - It is ubiquitous across multiple industries
- The business of software has graduated from a “garage” operation to an “enterprise” profession ----- including recent “Facebook”
- There is a move to treat software engineering as an engineering profession
- 15 universities have received accreditation (2009) from accreditation board of engineering and technology for SE specifically

Engineering Software

Software Engineering “Professionals”

- There is no equivalent “professional engineer” (PE) designation for software engineers, yet.
  - Except in Texas where the board of professional engineers adopted software engineering as a specific discipline under which en engineering license may be issued.



So -- how do we become more “professional” in Software Engineering

Any More Guidance?

Engineering Software

IEEE-CS/ACM Version 5.2 Report

- 8 principles** for ethics and professional practices in software engineering
  - Software engineers shall act consistently with the public interest
  - Software engineers shall act in a manner that is in the best interest of their client and employer, consistent with the public interest
  - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible

Engineering Software

IEEE-CS/ACM Version 5.2 Report

- 8 principles** for ethics and professional practices in software engineering
  - Software engineers shall maintain integrity and independence in their professional judgment
- Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance
- Software engineers shall advance the integrity and reputation of the profession consistent with the public interest

Engineering Software

IEEE-CS/ACM Version 5.2 Report

- 8 principles** for ethics and professional practices in software engineering
  - Software engineers shall be fair to and supportive of their colleagues
- 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.

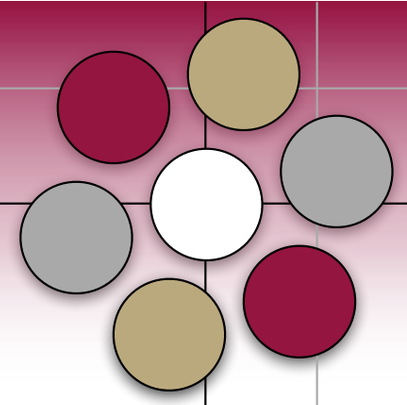
A “Simpler” Set of Behavioral Rules

- Respect others
- Strive for fairness
- Perform to one’s best capability
- Follow the law

Engineering Software

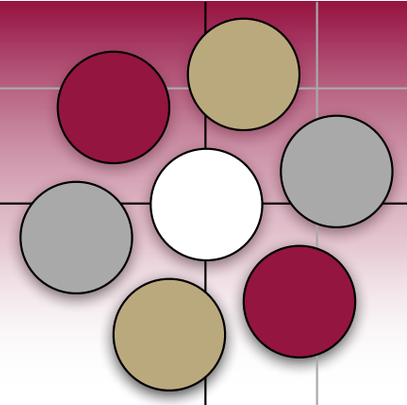
“General Principles”

- Different from other engineering disciplines such as civil or mechanical, there is no one set of “universal principles” in software engineering that is agreed to by everyone. Neither is there any “law” of software engineering such as Newton’s law of motion in physics



“General Principles”

- There are, however, several that are well received and respected.
  - Davis’s Principles
  - Royce’s Principles
  - Wasserman’s Concepts



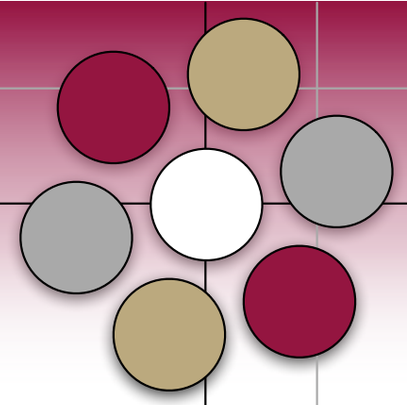
Davis's Early 15 principles

1. Make quality number 1
2. High quality software is possible
3. Give products to customers early
4. Determine the problem before writing the requirements
5. Evaluate design alternatives
6. Use an appropriate process model
7. Use different languages for different phases
8. Minimize intellectual distances
9. Put techniques before tools
10. Get it right before you make it faster
11. Inspect code
12. Good management is more important than good technology
13. People are the key to success
14. Follow with care
15. Take responsibility

Royce's More "Modern" set of Principles

1. Base the process on an architecture first approach
2. Establish iterative process --- address risk early
3. Emphasize component-based development to reduce effort
4. Establish change management
5. Use round-trip engineering – a form of iterative process
6. Use model-based and machine processable notations for design
7. Establish process for quality control and project assessment
8. Use approach that allows artifacts to be demonstrated early
9. Plan to have incremental releases
10. Establish a configurable process to suit the needs

Do you agree with these? Why?



Wasserman's Fundamental Concepts

1. Abstraction
2. Analysis and design methods and notation
3. User interface prototyping
4. Modularity and architecture
5. Reuse
6. Life cycle and process
7. Metrics
8. Tools and integrated environment

Important concepts -- how do they relate to earlier listed the principles from Davis or Royce?

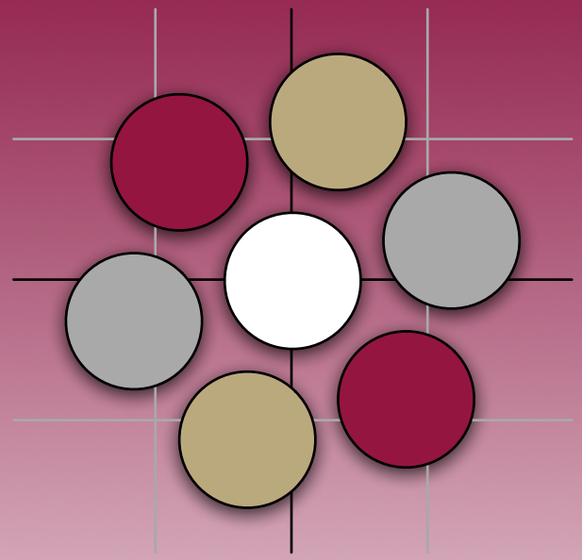
Engineering Software

Software Engineering is about identifying repeatable generalizable processes that can deliver

High-quality software

On time

Within budget



WESTMONT COMPUTER SCIENCE