
Coca-Cola


pepsi

emacs

vi

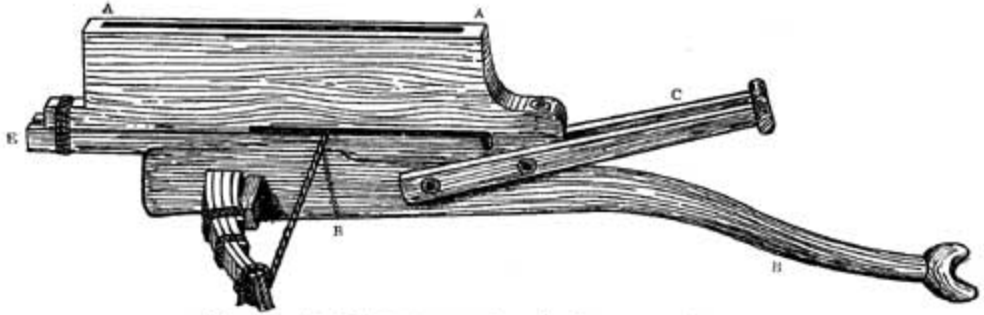
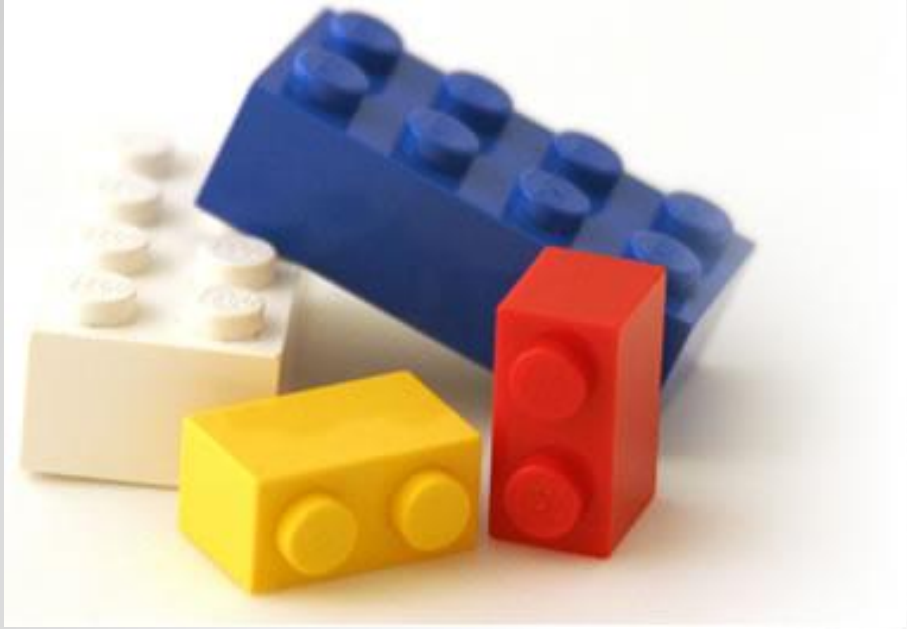


FIG. 171.—SIDE VIEW OF THE CHINESE REPEATING CROSSBOW.




MMIX

CODING

SOFTWARE ENGINEERING

SCIENCE

ART

WORK

FUN

Overview of Topics

Business Case, Planning, Requirements

Process, Roles and Rules

Design, Estimation, Scope, Scale

Documentation, Implementation and Coding

Tools, Testing and Iteration

Delivery, Support and Life Cycle

Software Engineering

One Definition

Principles, techniques, and habits that enable us to consistently create software solutions that:

- Satisfy the needs and wants of the user.
- Are easy to correctly modify to satisfy the evolving needs and wants of the user.

Analysis of the Definition

Important words:

- “Consistently”
  - Once is not enough
  - Once in a while is not enough
  - Luck is not enough
  - Success through heroics is not enough

Analysis of the Definition

“Consistent” implies “sustainable”

- “Death Marches” are not acceptable
- Overtime is the exception, not the rule
- “Showstopper” and “Shipstopper”

Analysis of the Definition

Software Solutions

- What is the difference between “Software” and “Software Solutions?”

There’s more to a solution than just software

- Packaging
- Installation/usage instructions
- Usable software!

Analysis of the Definition

Needs and wants of the user

- Fulfills the requirements
- Free of bugs
- Does not have extra (unexpected) features
- Includes non-functional attributes, including:

  Reliability, Usability, Availability, Performance
- Important: the user needs the software to be completed on time!

Analysis of the Definition

Correctly Modify Software

- It’s easy to change software
- It’s challenging to change software correctly
- Keys to success:
  - Easy to understand source code
  - Changes have local -- not global -- implications

One Definition

Principles, techniques, and habits that enable us to consistently create software solutions that:

- Satisfy the needs and wants of the user.
- Are easy to correctly modify to satisfy the evolving needs and wants of the user.

Why Do We Care?

Software is a huge business, and is getting bigger all the time.

- One estimate: at current growth rates, every man, woman, and child will be needed to write software by the year 2050

Software is an important part of our lives.

- More and more life-critical systems
- More and more “way-of-life”-critical systems

History ...

1968, NATO conference on software, Garmisch, Germany

The software crisis

- Hardware was getting bigger, allowing much more complex software.
- The software craft was largely ad hoc.
  - The ad hoc approach would not scale up.
- Systems were late, over budget
- Systems were unreliable and difficult to maintain.
- Hardware costs dropping, software costs rising.

History ...

The term “software engineering:”

- Coined as an attempt to express a goal.
- Make software development more like the engineering disciplines:
  - Founded on principles
  - Produce reliable products
  - Have consistent performance
- An aspiration, not a reality.

How Are We Doing?

How Are We Doing?

- Projects are still late, over budget.
  - In some cases, we are getting better: Well-defined enhancements to existing systems.
- Reliability has improved, but still a big problem.
- Usability:
  - Some improvement.
  - Users have grown used to systems that are hard to use.

Your Life Is On The Line

You write software. Would you trust your life to software?

But you do…

- Automobiles and traffic semaphores
- Airplanes and air traffic control
- Nuclear reactors
- Hospital equipment

Your Money Is On The Line

- Your bank
- The stock market
- Your credit

- Accuracy
- Security

Your Identity Is On The Line

But that’s a discussion for another day

At The End of This Course ...

- You should have an increased awareness of your responsibility to others.
- You should understand things you can do to improve the situation.
- You should have a commitment to carry through on your responsibility.
- You should be more competent.

What Will It Take To Succeed?

Two facets:

- Personal competence
- Team effectiveness

Personal Competence

- You must write good software!
  - Includes correctness, modifiability, and consistency.
- You must understand and follow appropriate development practices.
  - Includes supporting activities.
  - Includes development processes.

Team Effectiveness

- Software development is a team sport
- Therefore, you need practices that enable the team to function smoothly.
  - Everyone follows the same processes.
  - Well understood roles and responsibilities.
  - Communication and Collaboration.
  - Organizational learning.

During This Semester

We spend half the time on personal practices:

- Coding
- Design
- Documentation
- Etc.

The intent is for these to become habits.

During This Semester

We spend the other half on software engineering topics:

- Process
- Organizational dynamics
- The nature of software development

These will be introductory, to be studied in greater depth in future courses.

A Self Test

Write down some key words that represent the activities that go into software development:

ready, go

What Did You Write?

- Problem definition (business case)
- Requirements development
- Construction (i.e. development) planning
- Software architecture, or high-level design
- Detailed (functional) design
- Coding and debugging
- Unit testing
- Integration testing
- Integration
- System testing
- Maintenance

The Goal

Employers are looking for software professionals who can do more than write code:

- Design
- Software Architecture
- Technical Leadership

This course is a major step in this direction

- Become excellent individually and collectively
- Understand good engineering in the big picture
- Future courses further develop these disciplines