Software Engineering

Introduction

When computer software succeeds?

When it meets the needs of the people who use it, when it performs flawlessly over a long period of time, when it is easy to modify and even easier to use - it can and does change things for the better.

When software fails?

When its users are dissatisfied, when it is error prone, when it is difficult to change and even harder to use - bad things can and do happen.

We all want to build software that makes things better.

To succeed, we need discipline when software is designed and built.

We need an *engineering approach* - we need Software Engineering.

What is Software Engineering?

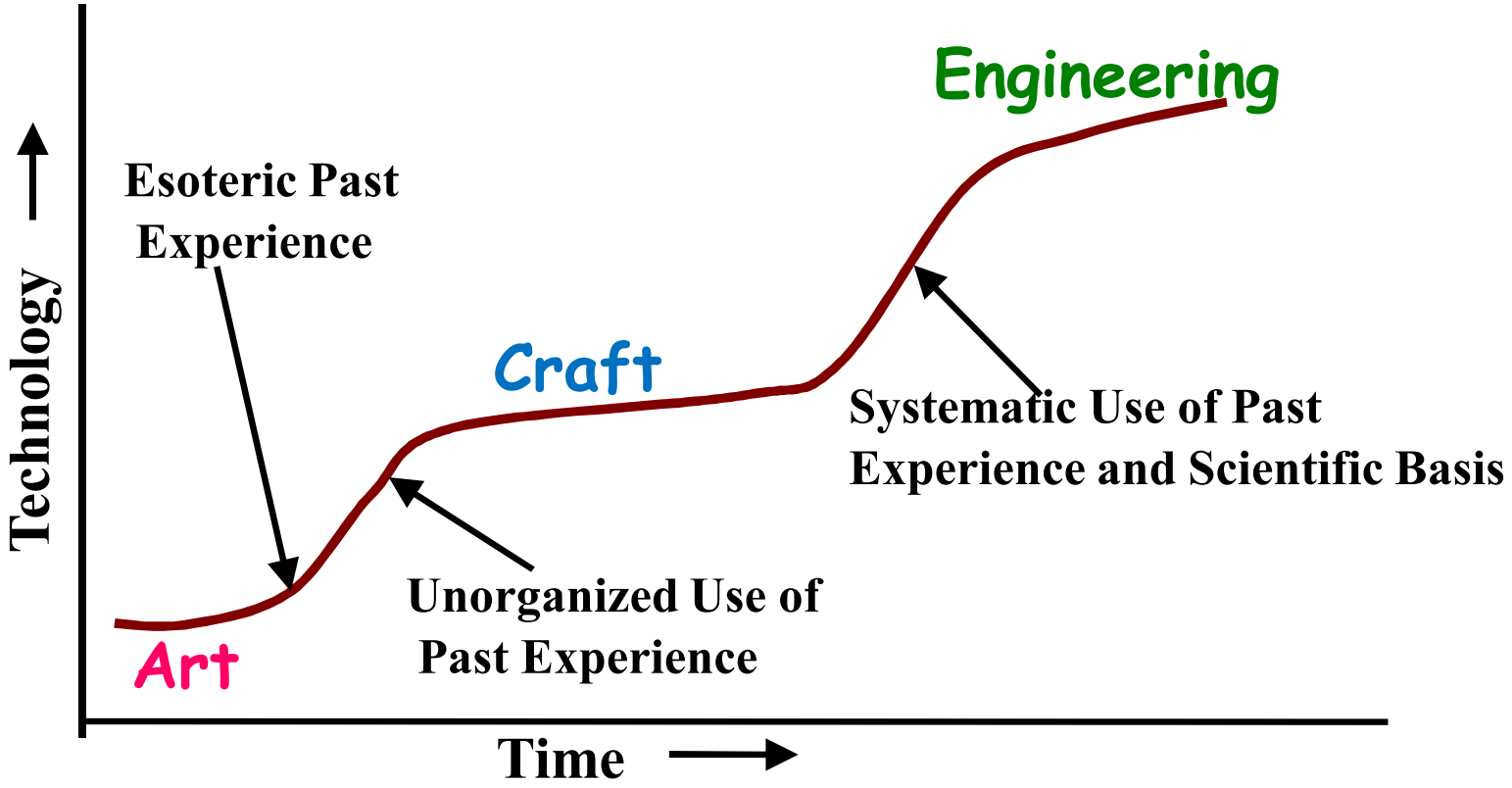
Engineering approach to develop software.

Systematic collection of past experience:

- Techniques
- Methodologies
- Guidelines

Technology Development Pattern

Saree designing analogy



What is software?

- Computer programs and associated documentation
  - Documentation: instruction manuals, specification, algorithms, test cases, …
  - May need user-interface, design may involve both software and hardware

- May be developed for various ends
  - For a particular customer / client
  - For general market
  - For in-house use by the company itself
  - Prototype for larger systems

Software application domains

- Broad categories
  - System software
  - Application software
    - Engineering/scientific software
    - Embedded software
    - Web applications

Software crisis

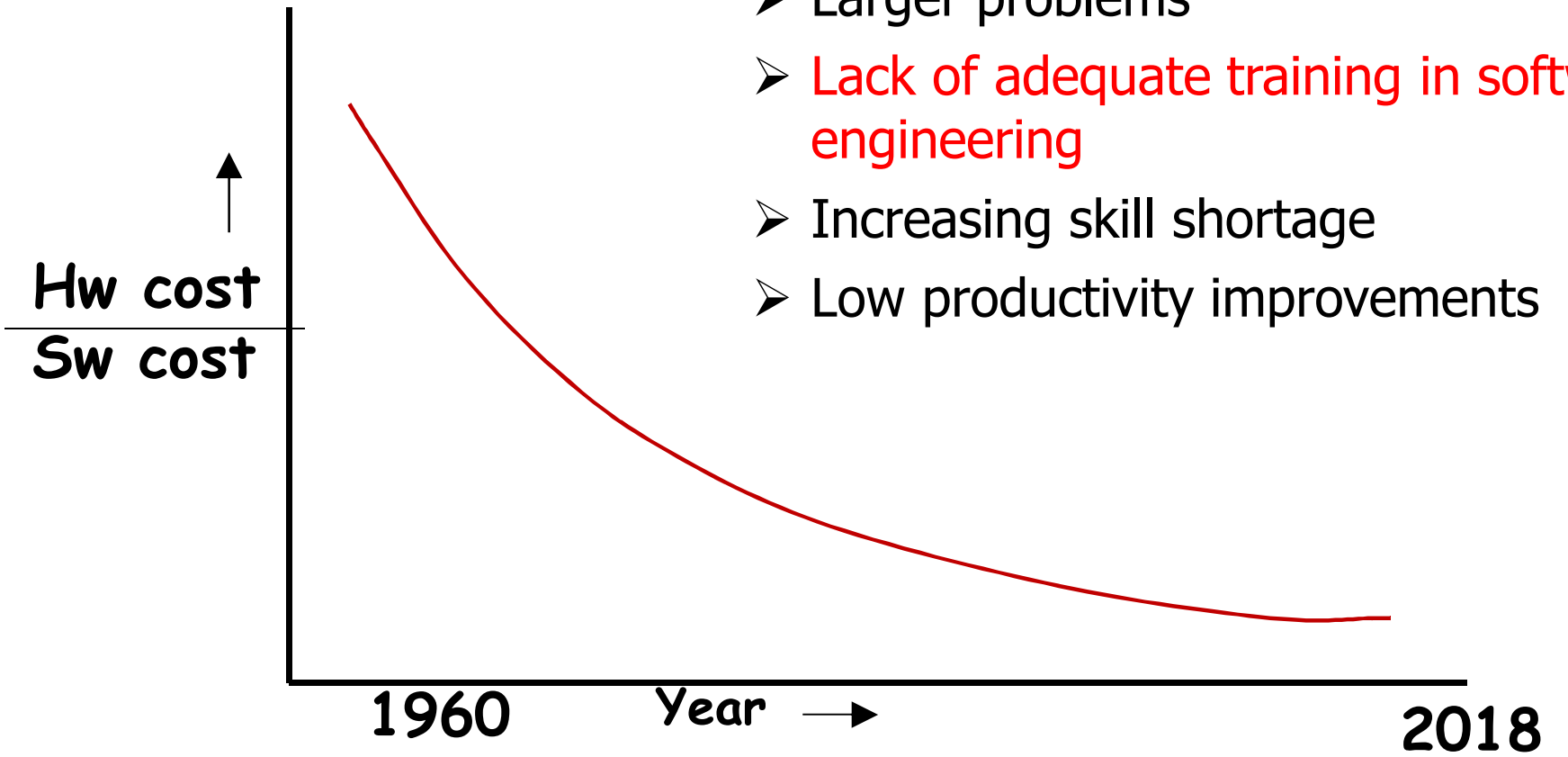
- As software became more complex, software products
  - Failed to meet user requirements
  - Became too expensive
  - Frequently crash
  - Difficult to modify, debug, enhance
  - Often delivered late
  - Used resources non-optimally

- Software Engineering aims to eliminate software crisis

Software Crisis

Factors Contributing to the Software Crisis

- Larger problems
- Lack of adequate training in software engineering
- Increasing skill shortage
- Low productivity improvements



Relative Cost of Hardware and Software

Hardware vs software development

- Customary procedure if fault discovered
  - Hardware: repair/replace/return products
  - Software: wait / ask for patch/ debugging, …
- Lifetime
  - Hardware: finite lifetime after which it will be discarded
  - Software: expected to perform same forever
- Prototype available for most hardware systems, usually meant for mass production

Principles Used by Software Engineering

Mainly two important principles are deployed:

- Abstraction: Simplify a problem by omitting unnecessary details.
  - Focus attention on only one aspect of the problem and ignore irrelevant details.
- Decomposition: Decompose a problem into many small independent parts.
  - The small parts are then taken up one by one and solved separately.
  - The idea is that each small part would be easy to grasp and can be easily solved.
  - The full problem is solved when all the parts are solved.

Software Engineering

Definition by Prof. Mary Shaw from Carnegie Mellon Univ.

- Software Engineering is
  - the branch of computer science
  - that creates practical, cost-effective solutions to computing and information processing problems,
  - preferentially by applying scientific knowledge
  - developing software systems in the service of mankind.

- Simpler definition:
  - Systematic and cost-effective approach to develop software with available resources

Why Study Software Engineering?

- To acquire skills to develop large programs.
  - Exponential growth in complexity and difficulty level with size.
  - The ad hoc approach breaks down when size of software increases
- Ability to solve complex programming problems:
  - How to break large projects into smaller and manageable parts?
  - How to use abstraction?
- Also learn techniques of:
  - Specification, design, user interface development, testing, project management, etc.
- To acquire skills to be a better programmer:
  - Higher Productivity
  - Better Quality Programs

Features of Software Engineering

- General principles
  - Abstraction and clarity: simplify problem by omitting unnecessary details, clearly understand requirements
  - Decomposition: decompose problem into small manageable parts, solve individually and then combine solutions
  - Reuse: code, documentation, design patterns
  - Automation and tools

- Systematic collection of past experience
  - Methodologies, guidelines, thumb-rules, quantitative techniques

Features of SE

- Error prevention instead of error correction
  - Detect errors as close as possible to point of introduction
- Coding is only a small part of software development
  - Emphasis on requirement specification, design stages
- During every stage
  - Periodic reviews
  - Systematic testing & debugging techniques
  - Better visibility of design and code – consistent and standard documentation produced

Attributes of good software

- The software should deliver the required functionality and performance to the user and should be
  - Maintainable: can evolve to meet changing needs
  - Dependable: must be trustworthy
  - Efficient: should not make wasteful use of system resources
  - Usable: usable by the users for which it was designed, accompanied by good-quality manuals, documentation, ...

Resource and References

BOOK: Fundamentals of Software Engineering

By Rajib Mall