Software Engineering

Organization of this Lecture:

- Nature of Software
- Programs vs. Software Products
- What is Software Engineering?
- Why study Software Engineering?
- Evolution of Software Engineering
- Types of Software
- Stakeholders in Software Engineering
- Software Quality
- Software Crises
- Summary

Programs versus Software Products

• Usually small in size	• Large
• Author himself is sole user	• Large number of users
• Single developer	• Team of developers
• Lacks proper user interface	• Well-designed interface
• Lacks proper documentation	Well documented & user-manualprepared
• Ad hoc development.	• Systematic development

The Nature of Software...

Software is intangible

- Hard to understand development effort

Software is easy to reproduce

- Cost is in its *development*
  - —in other engineering products, manufacturing is the costly stage

The industry is labor-intensive

- Hard to automate

The Nature of Software ...

Untrained people can hack something together

- Quality problems are hard to notice

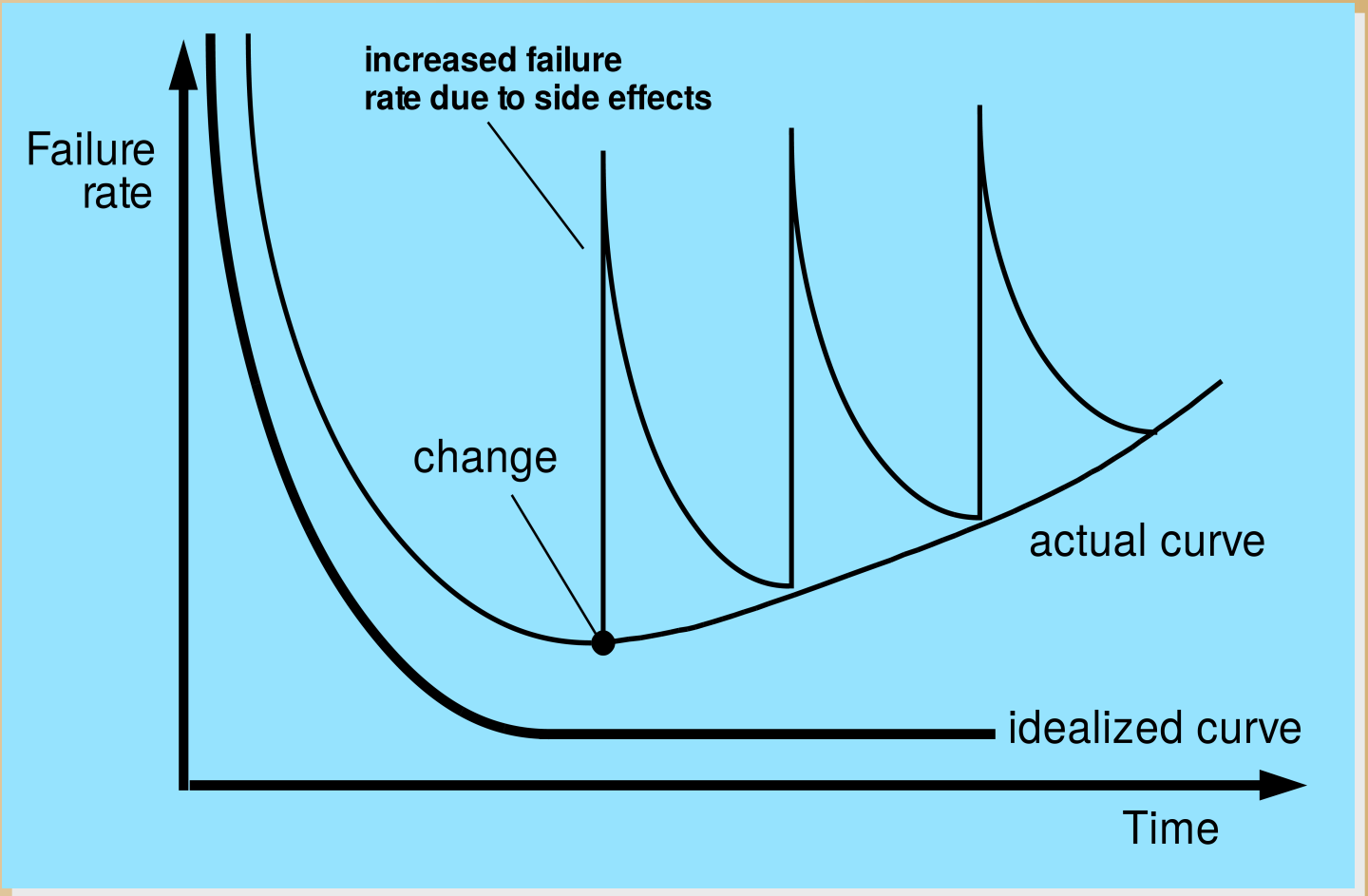
Software is easy to modify

- People make changes without fully understanding it

Software does not ‘wear out’

- It *deteriorates* by having its design changed:
  - —erroneously, or
  - —in ways that were not anticipated, thus making it complex

Nature of Software
Wear vs. Deterioration



What is Software Engineering?

The process of solving customers' problems by the systematic development and evolution of large, high-quality software systems within cost, time and other constraints

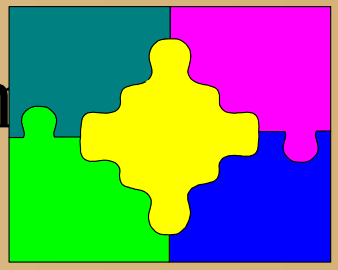
Other definitions:

- IEEE: the application of a systematic, disciplined, quantifiable approach to the development, operation, maintenance of software; that is, the application of engineering to software.

Why Study Software Engineering? (1)

- 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:

Why Study Software Engineering? (2)

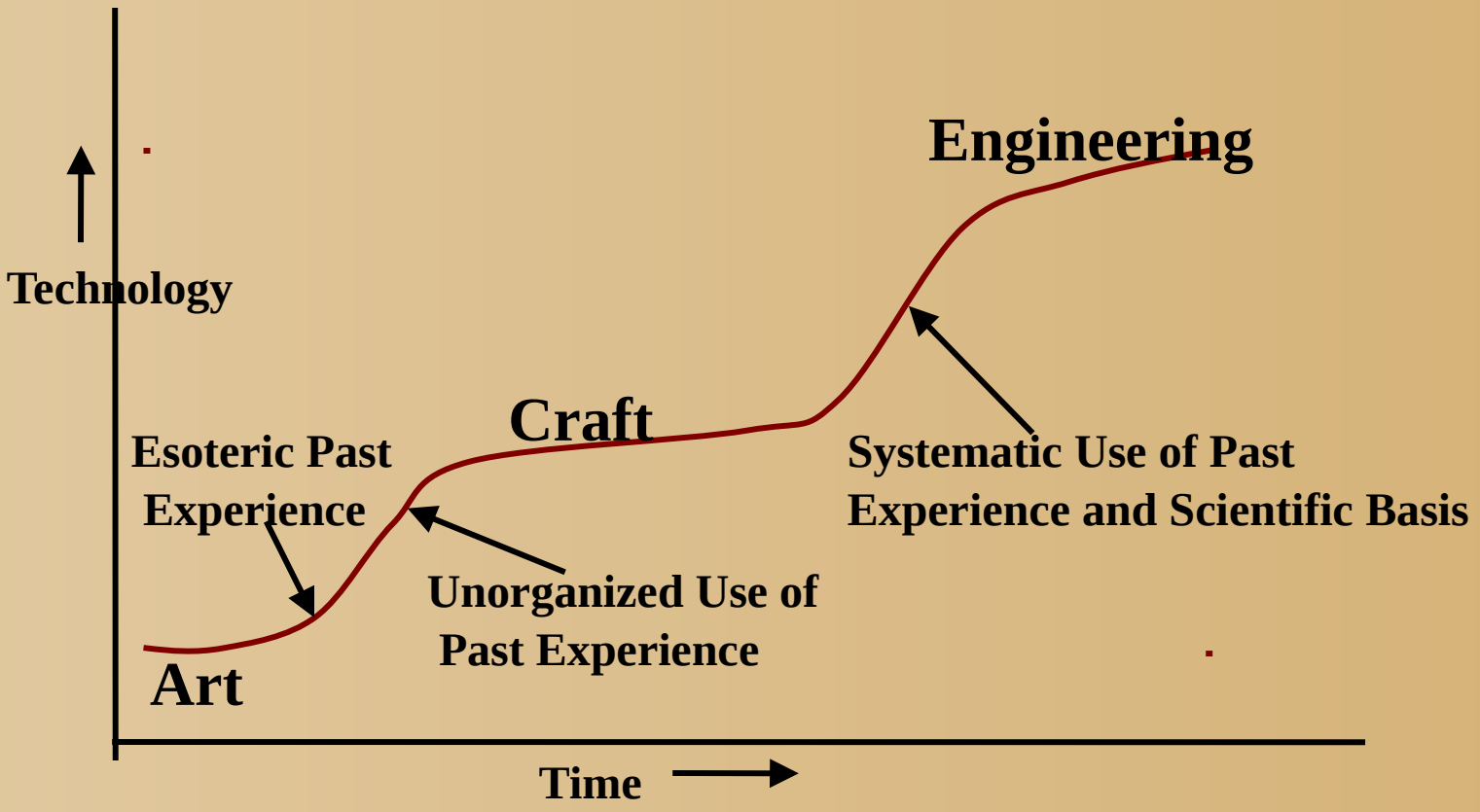


- Ability to solve complex programming problems:
  - How to break large projects into smaller and manageable parts?
- Learn techniques of:
  - specification, design, interface development, testing, project management, etc.

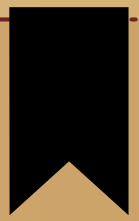
Introduction: Software is Complex

- Complex ≠ complicated
- Complex = composed of many simple parts related to one another
- Complicated = not well understood, or explained

Technology Development Pattern



Types of Software...



Custom

- For a specific customer

Generic

- Sold on open market
- Often called
  - —COTS (Commercial Off The Shelf)
  - —Shrink-wrapped

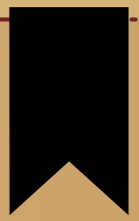
Embedded

- Built into hardware
- Hard to change

Types of Software

Differences	Custom	Generic	Embedded
Number of copies in use	Low	Medium	High
Total processing power devoted to running this type of software	Low	High	Medium
Worldwide annual development effort	High	Medium	Medium

Types of Software



Real time software

- E.g. control and monitoring systems
- Must react immediately
- Safety often a concern

Data processing software

- Used to run businesses
- Accuracy and security of data are key

Some software has both aspects

Exercise 1.1

Q1. What are the most important differences between generic software product development and custom software development? What might this mean in practice for users of generic software products?

GENERIC	CUSTOM
1. The specification is owned by the **product developer**.	The specification is owned and controlled by the **customer as the domain, requirement and envirnment being unique to the customer**.
2. The developer **can quickly decide to change** the specification in response to some external change.	changes have to be negotiated between the **customer** and the developer and may have contractual implications.

USERS:

For users of generic products, this means they have no control over the software specification so cannot control the evolution of the product. The developer may decide to include/exclude features and change the user interface. This could have implications for the user's business processes and add extra training costs when new versions of the system are installed. It also may limit the customer's flexibility to change their own business processes.

E 1.1

Classify the following software according to whether it is likely to be custom, generic or embedded (or some combination); and whether it is data processing or real-time.

(a) A system to control the reaction rate in a nuclear reactor.

(b) A program that runs inside badges worn by nuclear plant workers that monitors radiation exposure.

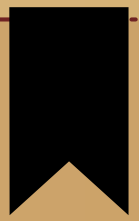
(c) A program used by administrative assistants at the nuclear plant to write letters.

(d) A system that logs all activities of the reactor and its employees so that investigators can later uncover the cause of any accident.

(e) A program used to generate annual summaries of the radiation exposure experienced by workers.

(f) An educational web site containing a Flash animation describing how the nuclear plant works.

Stakeholders in Software Engineering



1. Users
   - Those who use the software
2. Customers
   - Those who pay for the software
3. Software developers
4. Development Managers

All four roles can be fulfilled by the same person

ATM stakeholders

- Bank customers
- Representatives of other banks
- Bank managers
- Counter staff
- Database administrators
- Security managers
- Marketing department
- Hardware and software maintenance engineers
- Banking regulators

Main Stake Holders of Indian Rail way Issuing System

1. Indian Railway Department
2. Passangers
3. Banks
4. Mobile Service Provider
5. Software Developers
6. Project Manager

1.5 Software Quality...

Usability

- Users can learn it and fast and get their job done easily

Efficiency

- It doesn’t waste resources such as CPU time and memory

Reliability

- It does what it is required to do without failing

Maintainability

- It can be easily changed

Reusability

- Its parts can be used in other projects, so reprogramming is not needed

E1.2

For each of the following systems, which attributes of quality do you think would be the most important and the least important?

(a) A web-based banking system, enabling the user to do all aspects of banking on-line.

(b) An air traffic control system.

(c) A program that will enable users to view digital images or movies stored in all known formats.

(d) A system to manage the work schedule of nurses that respects all the constraints and regulations in force at a particular hospital.

(e) An application that allows you to purchase any item seen while watching TV.

Software Quality and the Stakeholders

Customer:
solves problems at
an acceptable cost in
terms of money paid and
resources used

User:

easy to learn;
efficient to use;
helps get work done

Developer:

easy to maintain;
easy to design;
easy to reuse its parts

Development manager:

pleases customers
sells more and
while costing less
to develop and maintain

Software Quality: Conflicts and Objectives

The different qualities can conflict

- Increasing efficiency can reduce maintainability or reusability
- Increasing usability can reduce efficiency

Setting objectives for quality is a key engineering activity

- You then design to meet the objectives
- Avoids ‘over-engineering’ which wastes money

Optimizing is also sometimes necessary

- E.g. obtain the highest possible reliability using a fixed budget

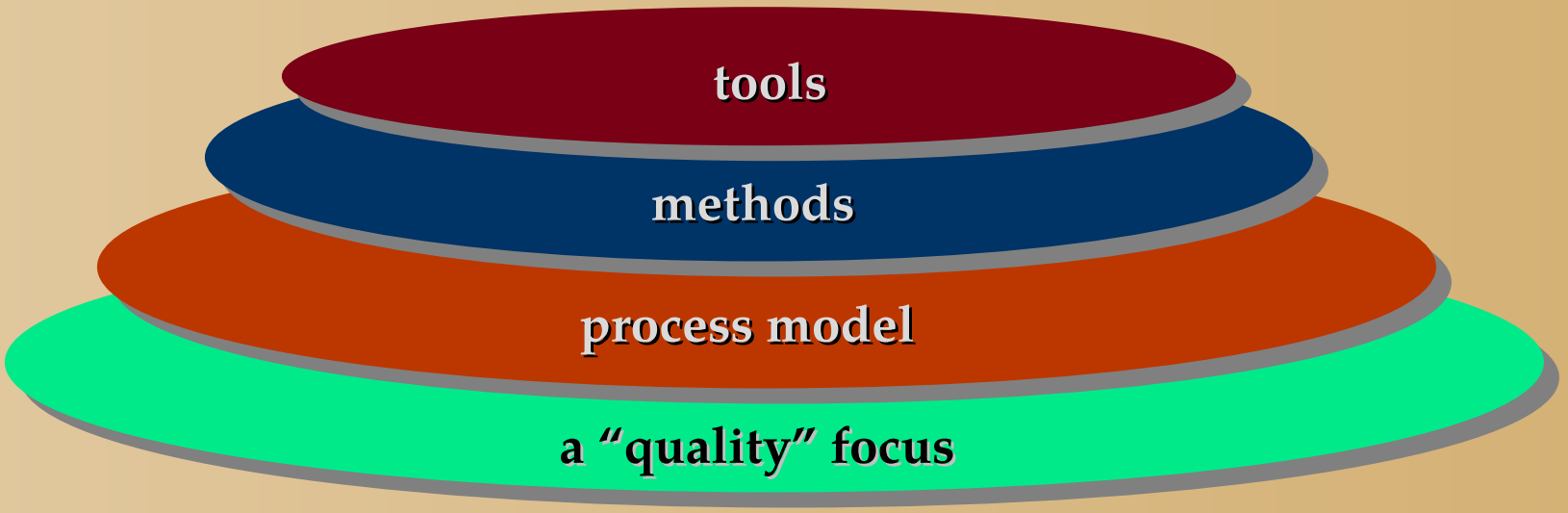
E 1.3

How do you think each of the four types of stakeholders described above would react in each of the following situations?

(a) You study a proposal for a new system that will completely automate the work of one individual in the customer's company. You discover that the cost of developing the system would be far more than the cost of continuing to do the work manually, so you recommend against proceeding with the project.

(b) You implement a system according to the precise specifications of a customer. However, when the software is put into use, the users find it does not solve their problem.

Software Engineering: A Layered Technology

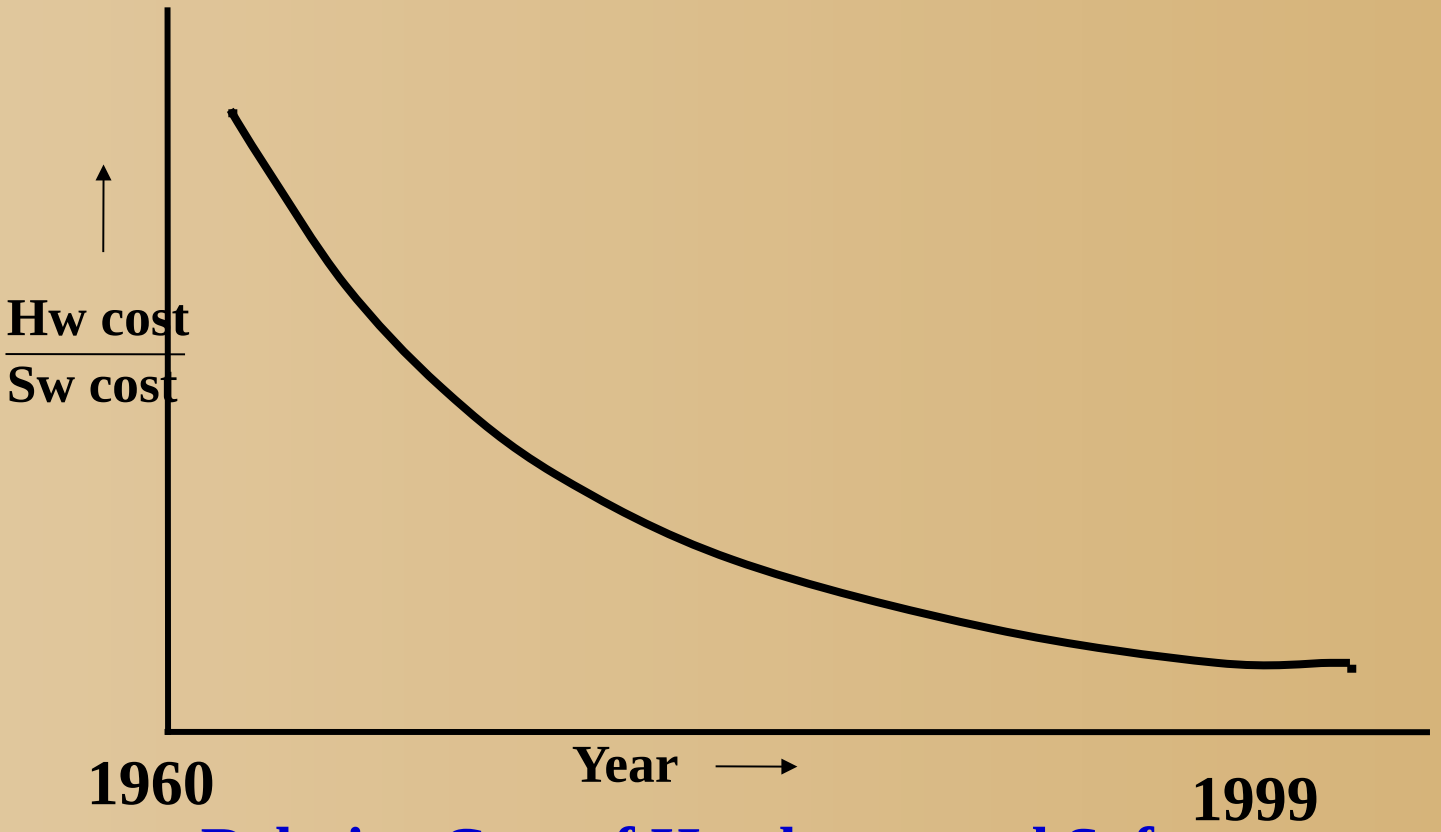


Software Engineering

Software Crisis

- Software products:
  - fail to meet user requirements.
  - frequently crash.
  - expensive.
  - difficult to alter, debug, and enhance.
  - often delivered late.
  - use resources non-optimally.

Software Crisis (cont.)

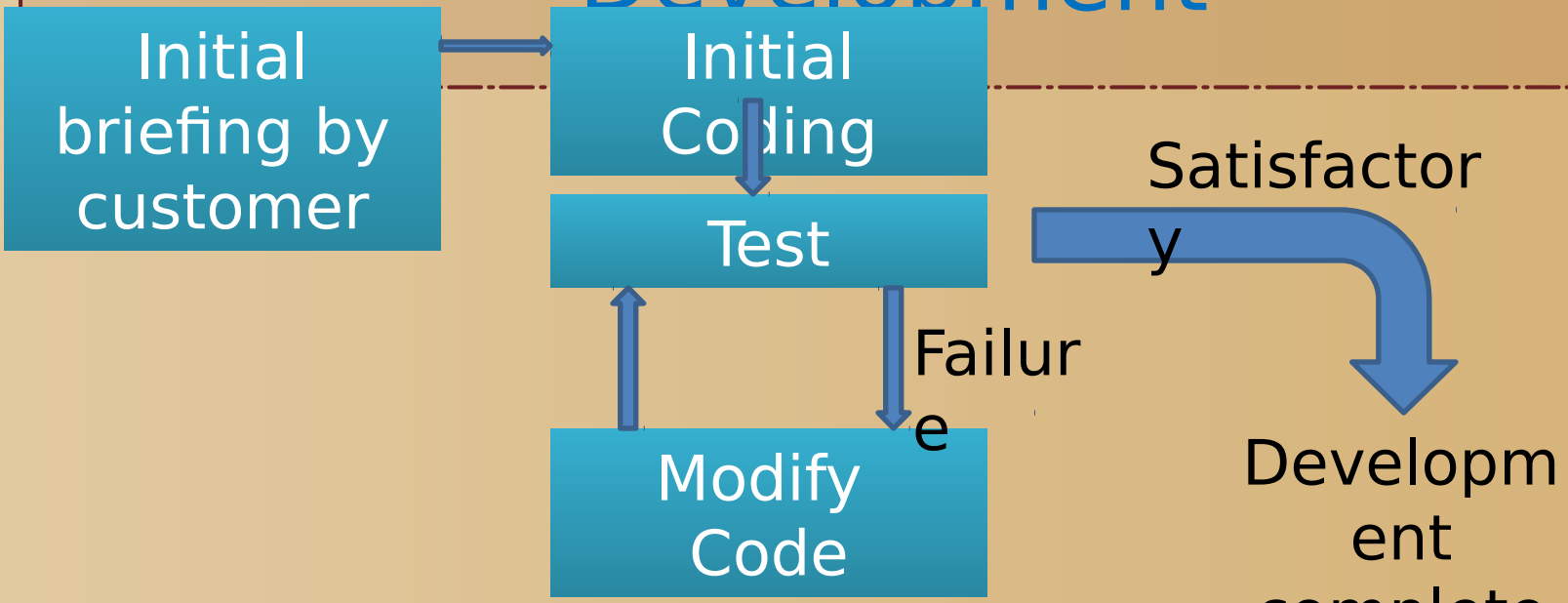


Relative Cost of Hardware and Software

Factors contributing to the software crisis

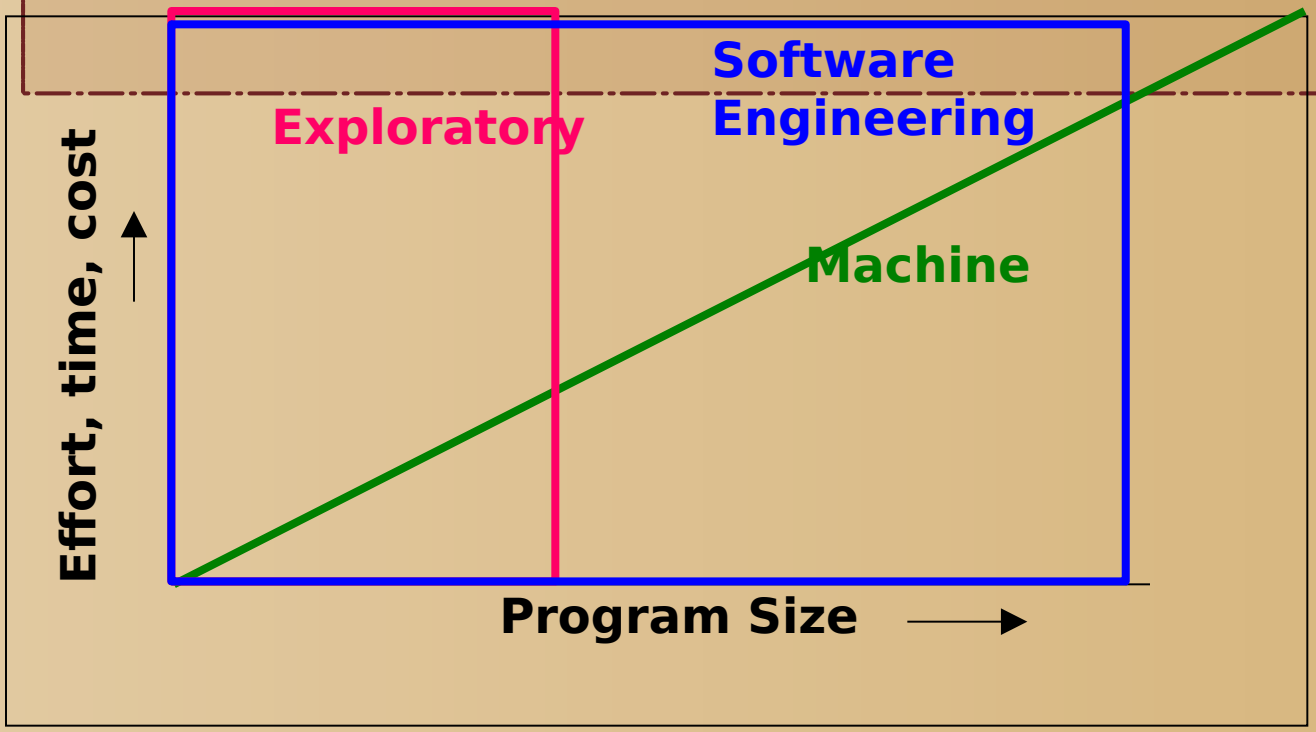
- Larger problems,
- Lack of adequate training in software engineering,
- Increasing skill shortage,
- Low productivity improvements.

Exploratory style of S/W Development



- Also called the “build and fix” style, normally a `dirty' program is quickly developed.
- The different imperfections that are subsequently noticed are fixed.
- This style usually results in un-maintainable code.
- It becomes very difficult to use this style in a

Exploratory style



- This style causes the perceived difficulty of a problem to grow exponentially due to human cognitive limitations.

How to Overcome Human Cognitive Limitations ?

- Mainly two important principles are deployed:
  - Abstraction
  - Decomposition

Abstraction

- Simplify a problem by omitting unnecessary details.
  - Focus attention on only one aspect of the problem and ignore irrelevant details.
- Case Study: Suppose you are asked to develop an overall understanding of some country.
  - One would meet all the citizens of the country, visit every house, and examine every tree of the country, etc. (unlikely)
  - One would possibly refer to various types of maps for that country. (likely)
- A map, in fact, is an abstract representation of a country.
  - Several abstractions are possible for the same object (country) like political map and physical map

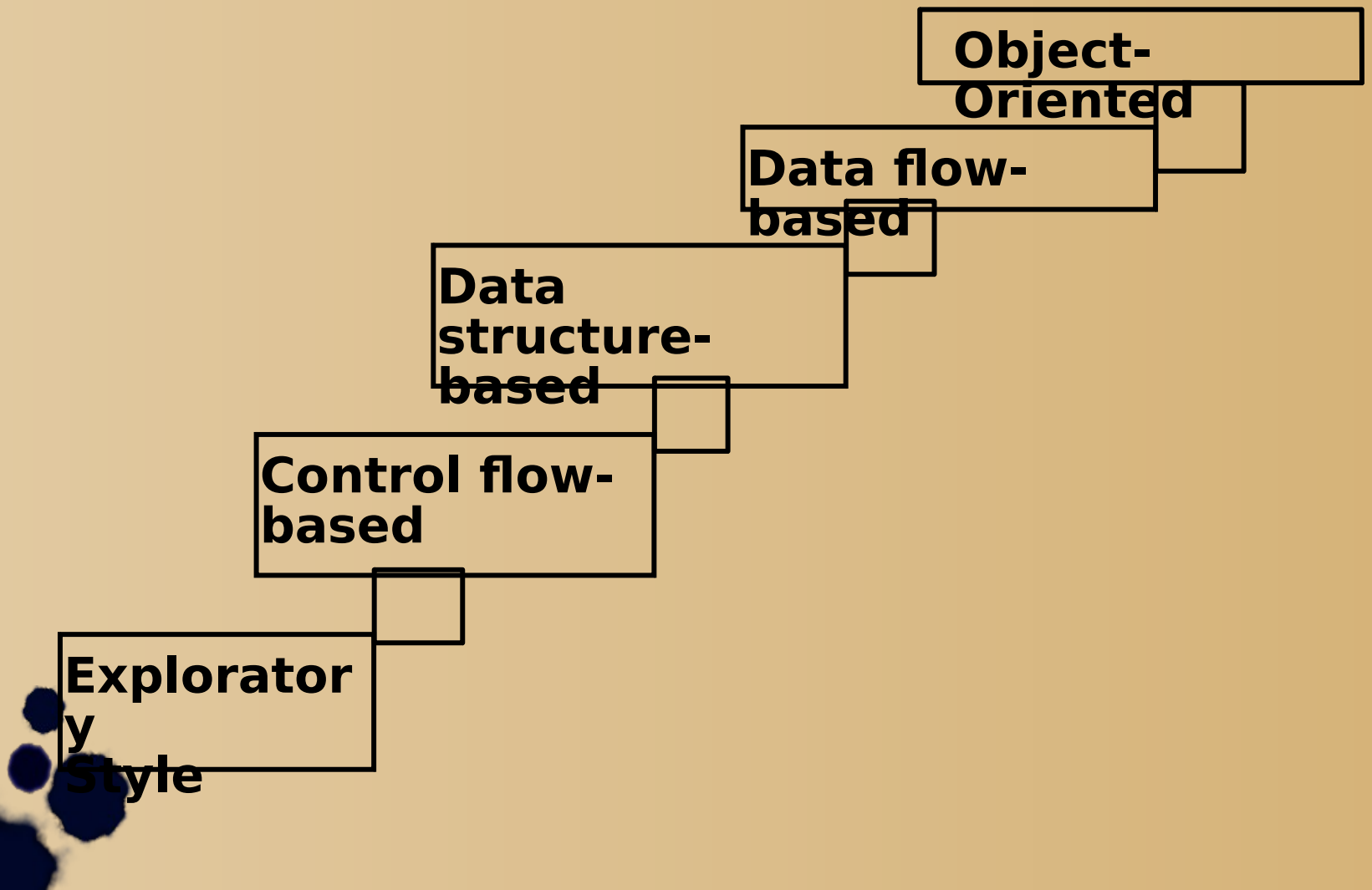
Decomposition

- 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.
- Example
  - Try to break a bunch of sticks tied together versus breaking them individually.

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?
- Learn techniques of:
  - specification, design, interface development, testing, project management, etc.

Evolution of Design Techniques



Modern S/W Development Practices

- Use of Life Cycle Models
  - Software is developed through several well-defined stages:
    - requirements analysis and specification,
    - design,
    - coding,
    - testing, etc.
- Emphasis has shifted from error correction to error prevention.
- Modern practices emphasize on detection of errors as close to their point of introduction as possible.
- Unlike exploratory style, now coding is considered only a small part of program development effort.

Summary

- Software engineering is:
  - systematic collection of decades of programming experience
  - together with the innovations made by researchers.

Summary

- A fundamental necessity while developing any large software product:
  - adoption of a life cycle model.

Summary

- Adherence to a software life cycle model:
  - helps to do various development activities in a systematic and disciplined manner.
  - also makes it easier to manage a software development effort.