SCS1302: Introduction to Software Engineering

Bachelor of Computer Science

1st Year

Semester 1

Prof. K. P. Hewagamage

UCSC

Ground Rules…

- Attendance – Theory and Practical
- UGVLE Course Page
  - Add a profile picture (Get to know members in the course)
  - Course Related Announcements
  - Participate activities
- How to ask questions
  - Direct Interaction (F2F)
  - slido.com (online interaction)

UCSC

Course Aim:

- To provide a broad understanding of the software engineering process, concepts and the systematic development and management of software projects.
- explain the software engineering principles and techniques that are used in developing quality software products
- apply software engineering principles and techniques appropriately to develop a moderately complex software system

UCSC

Learning Outcomes

After successfully completing this subject, students should be able to:

- **LO1**. Explain the software engineering principles and techniques in developing quality software products, such as the software development lifecycle, software quality, and software testing
- **LO2**. Apply software engineering principles and techniques appropriately in developing a moderately complex software system
- **LO3:** Use software engineering tools and technologies to support the development process
- **LO4:** Apply ethical and professional principles to software engineering practice.

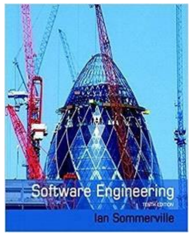
Outline of Syllabus

1. Introduction
2. Software Processes
3. Requirement Engineering
4. Agile Software Development
5. System modeling
6. Architectural design
7. Design and implementation
8. Software Testing
9. Software evolution

Software engineering

- The economies of ALL developed nations are dependent on software.
- More and more systems are software controlled
- Software engineering is concerned with theories, methods and tools for professional software development.
- Expenditure on software represents a significant fraction of GNP in all developed countries.

Main References

1.  Software Engineering by Ian Sommerville, 10th edition, Addison-Wesley, 2015.

2. Software Engineering: A practitioner's approach by Roger S. Pressman, 9th edition, McGraw-Hill International edition, 2020.

Software Engineering

1. Introduction to Software and Software Engineering

Intended Learning Outcomes

- Identify the problems associated with developing software
- Describe the need for a managed approach to software development
- Be able to define the term ‘Software Engineering’
- Identify software quality attributes and their classification

What is Software?

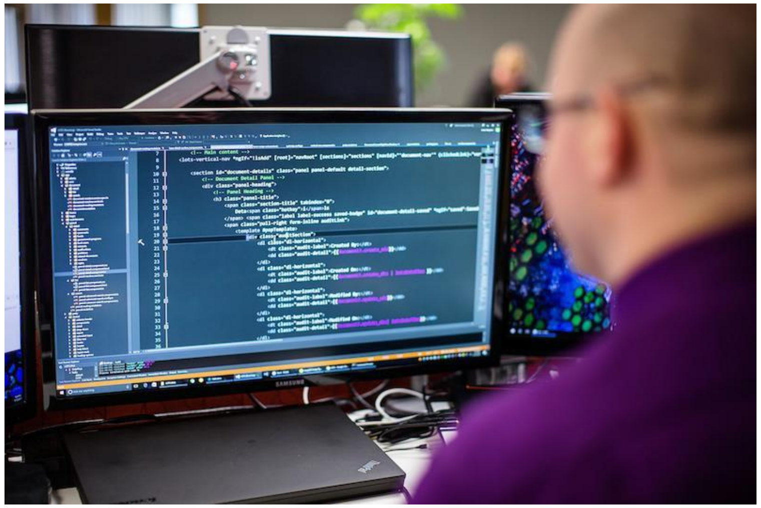
- Software is any set of machine-readable instructions to the computer's processor to perform specific operations.
- As a product, a Software system is a collection of intercommunicating components, programs, configuration files, system documentation and user documentation
- Software products may be developed for a particular customer or may be developed for a general market.
  - **Generic** - developed to be sold to a range of different customers
  - **Bespoke (custom)** - developed for a single customer according to their specification

Define - Software

Software is:

1) *Instructions (computer programs) that when executed provide desired features, function, and performance;*
2) *Data structures that enable the programs to adequately manipulate information.*
3) *Documentation that describes the operation and use of the programs.*

Software Projects vs. Other Projects



Software Projects vs. Other Projects

- The main difference in software engineering compared to other engineering disciplines are as follows;
  - It is difficult for a customer to specify requirements completely.
  - It is difficult for the developer to understand fully the customer needs.
  - Software requirements change regularly.
  - The environments to which software is made change regularly.
  - Software is primarily intangible; much of the process of creating software is also intangible, involving experience, thought and imagination (a brain product).
  - Because of intangibility it is difficult to specify software.
  - Customers and developers have communication gaps.

Software Projects vs. Other Projects

- It is difficult to test software exhaustively.
- Not like other engineering disciplines that have narrow scope, software can be developed in diverse contexts ranging from embedded systems to giant manufacturing systems.
- The replication of software is trivial and as a consequence software is elastic and gradually becomes more complex as changes are made over time.
- A small change to a few lines of code could do a big change in the overall behavior of the system.
- Finally, as a discipline, software development is so young that measurable, effective techniques are not yet available, and those that are available are not well-calibrated.

Questions about software

- Why does it take so long to get software finished?
- Why are development costs so high?
- Why can't we find all errors before we give the software to our customers?
- Why do we spend so much time and effort maintaining existing programs?
- Why do we continue to have difficulty in measuring progress as software is being developed and maintained?

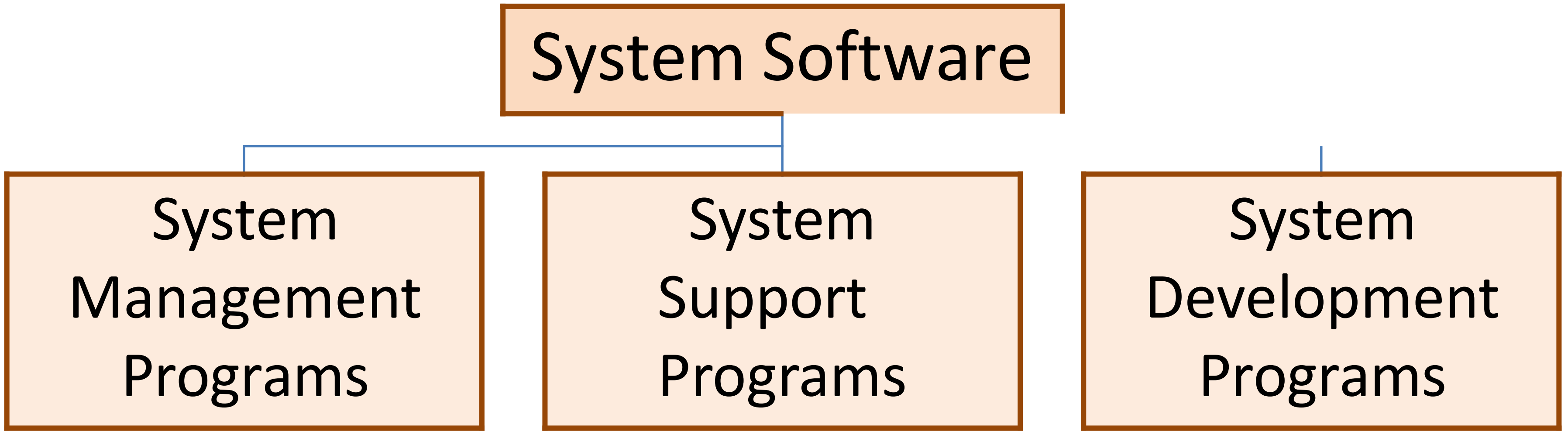
Main Types of Software

- There are two main types of software;

1. System Software
   - computer software designed to operate and control the computer hardware and to provide a platform for running application software

2. Application Software
   - set of one or more programs designed to carry out operations for a specific application

System Software

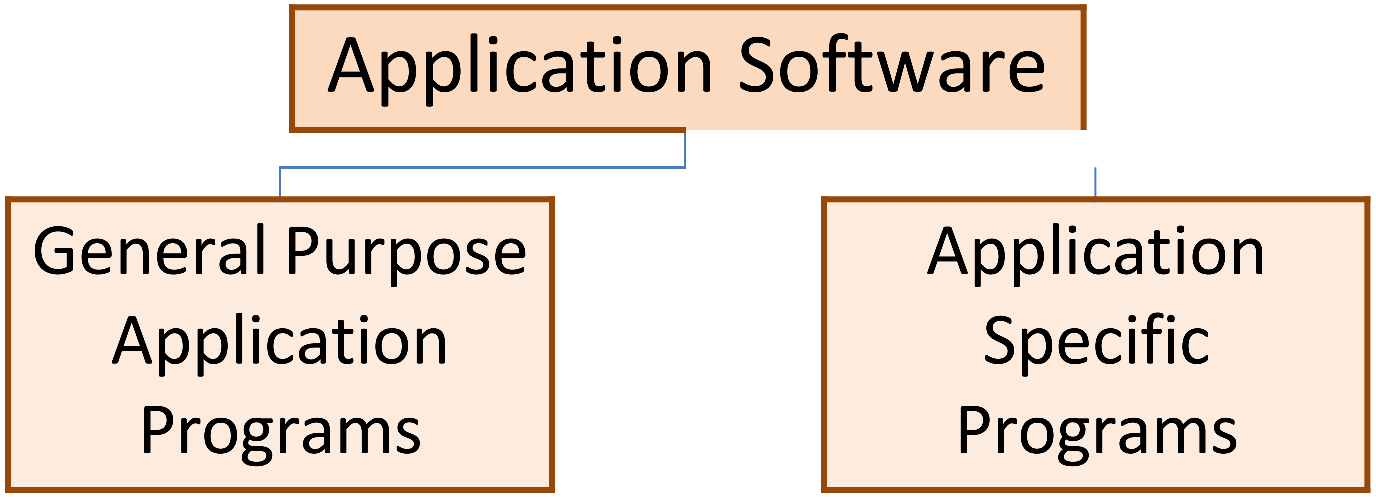


- Operating Systems
- Operating Environments
- Virtualization Tools
- Docker

- System Utilities
- Performance Monitors
- Security Monitors
- Database Management Systems Utilities

- Programming Language Translators
- Programming Environments
- Computer Aided Software Engineering (CASE) Packages

Application Software


Application Software
General Purpose Application Programs
Application Specific Programs

- Word Processing
- Electronic Spread Sheets
- Database Managers
- Graphics Software
- Integrated Packages

- Accounting, General Legers etc.
- Marketing-Sales Analysis etc.
- Manufacturing-Production Control etc.
- Finance-Capital Budgeting etc.

Common Software Types

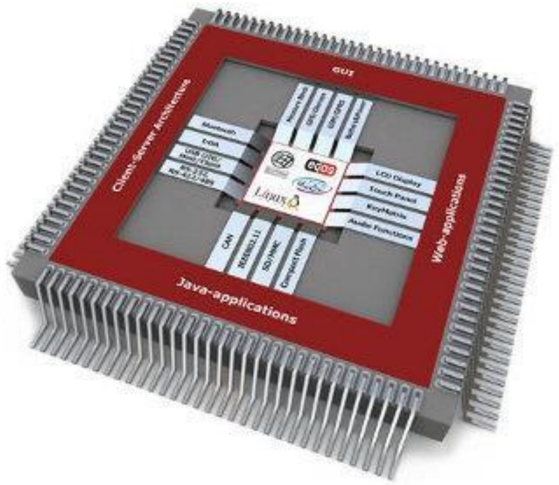
- **System Software:**
  - System software is a collection of programs is written to service the other programs.

Eg: Operating system component, drivers, telecommunication proces s

Common Software Types

- **Business software:**

  management information system software that access one or more large database containing business information



- **Embedded software:**

  Embedded software resides in read-only memory and is used to control product and system for the customer and industrial markets

Common Software Types

- **Web-based software:**

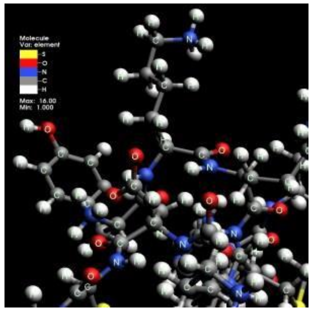
  The network becomes a massive computer providing an almost unlimited software resources that can be accessed by anyone with a modem.

- **Artificial intelligence software:**

  AI software makes use of non-numerical algorithms to solve the complex problems that are not amenable to computing or straightforward analysis.

UCSC

Common Software Types



- **Personal computer software:**

  Personal computer software market has burgeoned over the past two decades.

  Word processing, spreadsheets, computer graphic, multimedia management

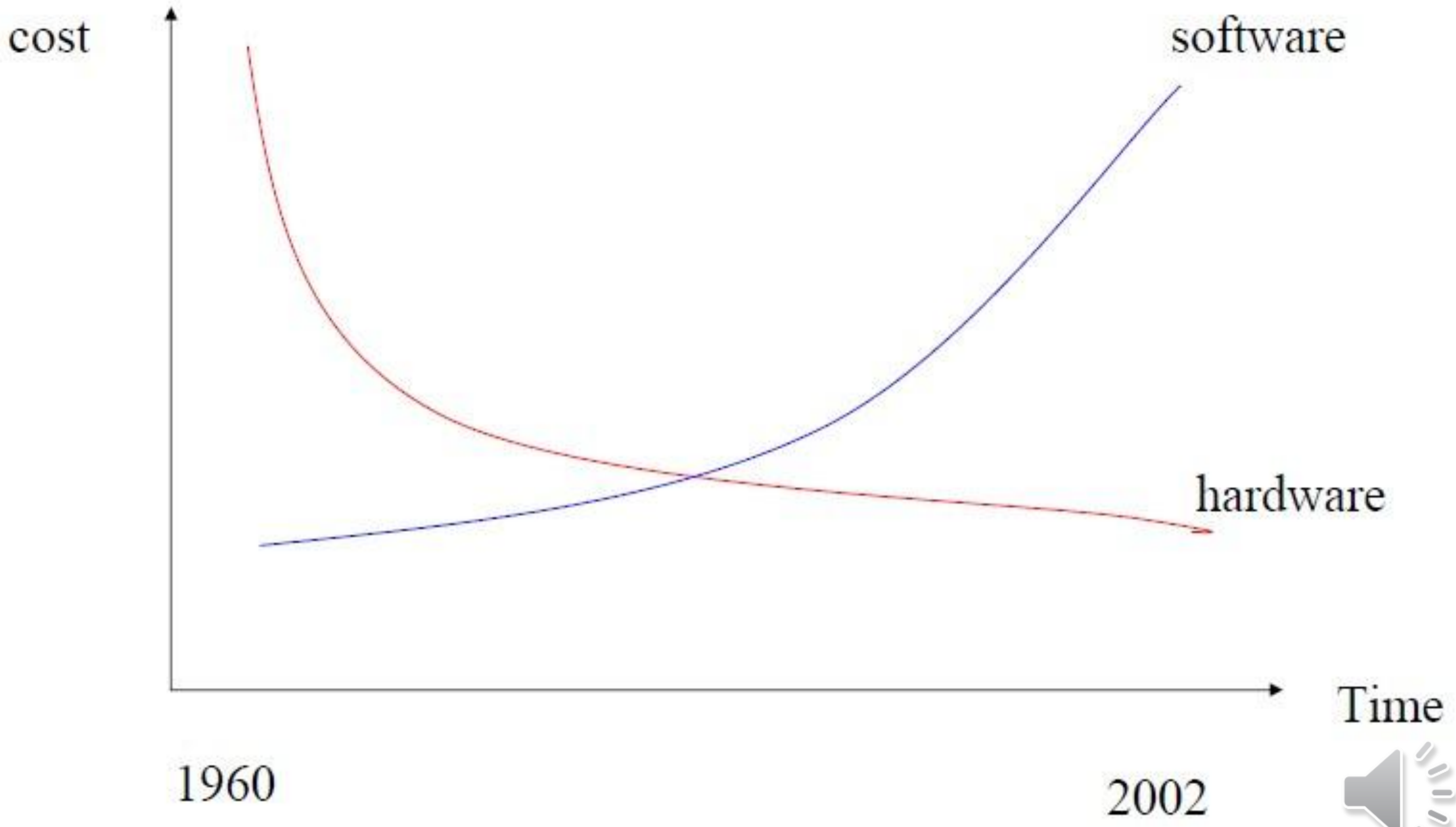
Software Application Domains

- System software.
- Application software.
- Engineering/Scientific software.
- Embedded software.
- Product-line software.
- Web/Mobile applications.
- AI software (robotics, neural nets, game playing).

What is the legacy software?

UCSC

Cost of Hardware vs. Software



What is the trend after 2020?

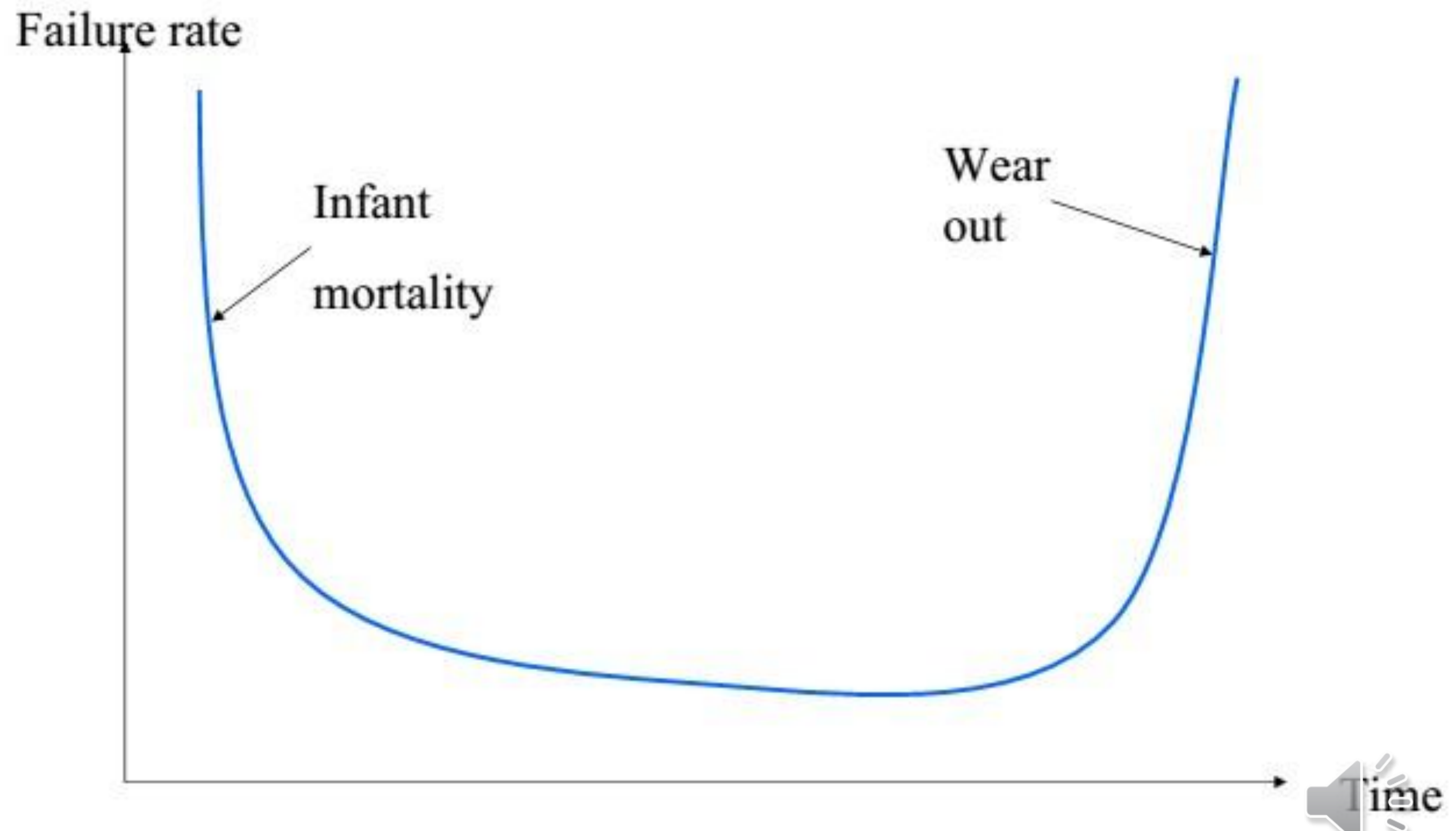
Software Costs

- Software costs often dominate system costs. The costs of software on a PC are often greater than the hardware cost
- Software costs more to maintain than it does to develop. For systems with a long life, maintenance costs may be several times development costs
- Software engineering is concerned with cost-effective software development

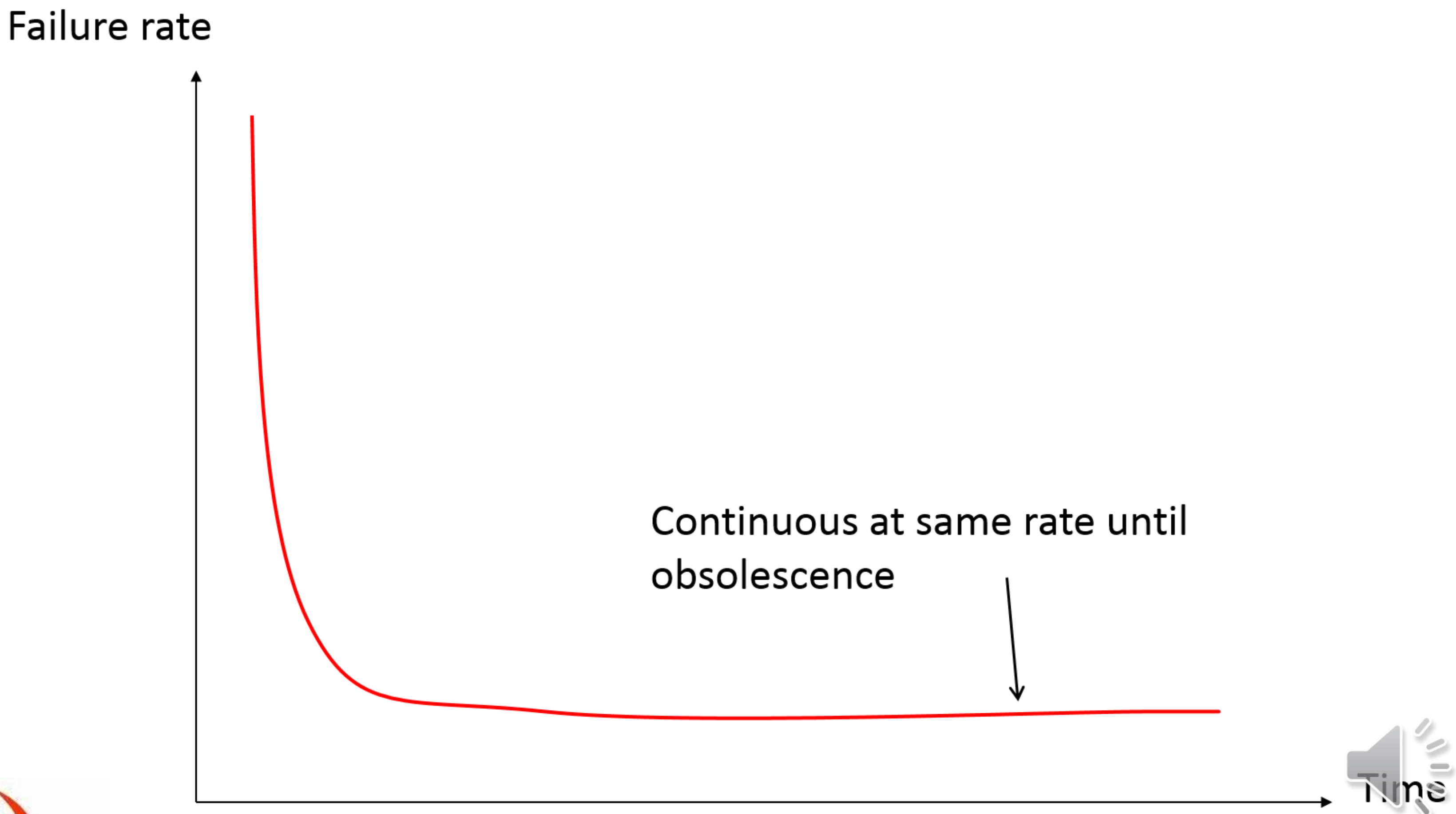
Failure Curves for Software & Hardware

- Software is engineered or developed, not manufactured in the traditional sense.
- Software does not wear out in the same sense as hardware.

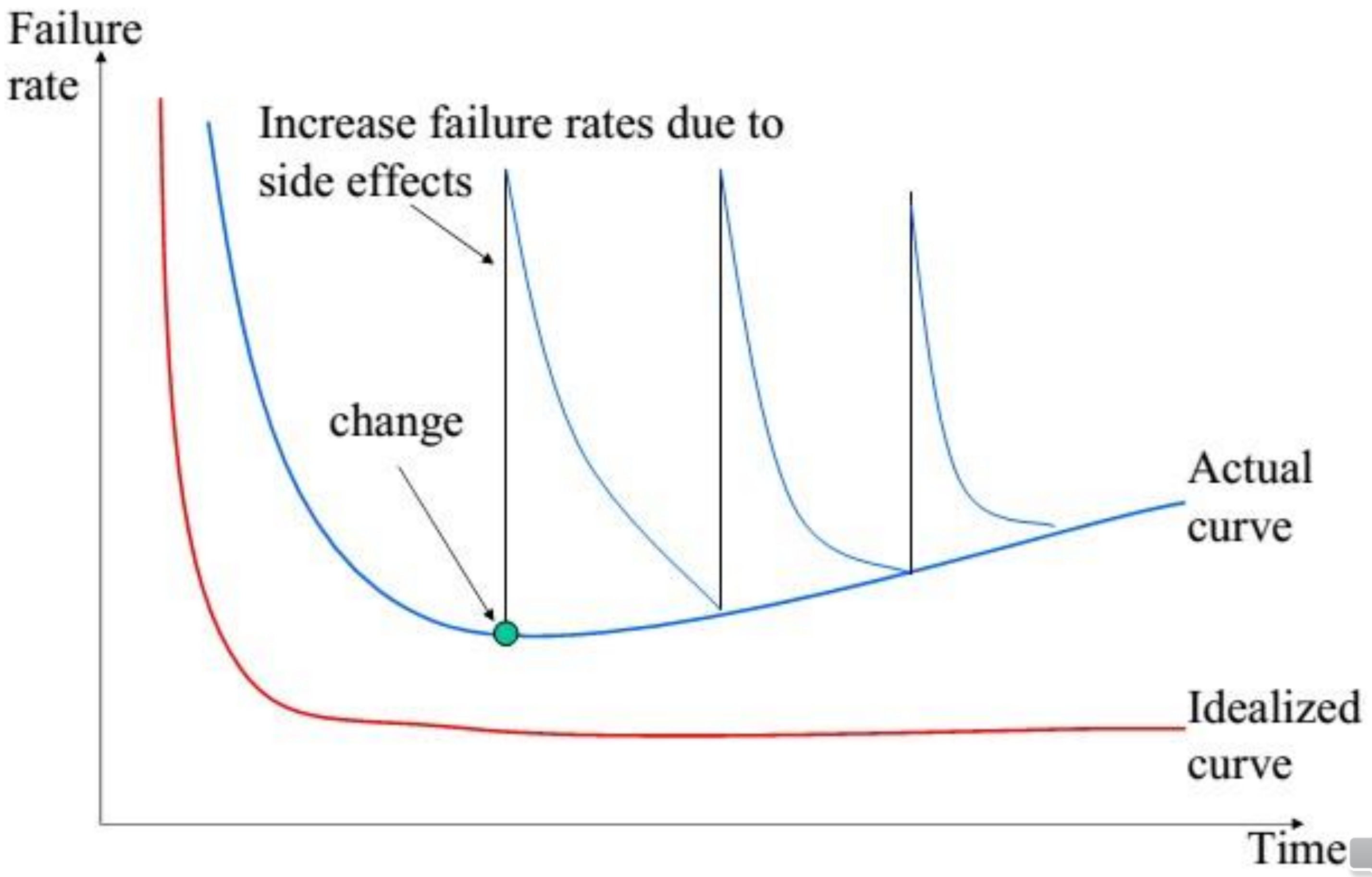
Failure "Bathtub" curve for hardware



Failure curve for software



Failure curve for software: in reality



Software project failure

Increasing system complexity

As new software engineering techniques help us to build larger, more complex systems, the demands change. Systems have to be built and delivered more quickly; larger, even more complex systems are required; systems have to have new capabilities that were previously thought to be impossible.

Failure to use software engineering methods

It is fairly easy to write computer programs without using software engineering methods and techniques. Many companies have drifted into software development as their products and services have evolved. They do not use software engineering methods in their everyday work. Consequently, their software is often more/less expensive but less reliable than it should be.

Software Development

- Software development is the computer programming, documenting, testing, and bug fixing resulting in a software product.
- **A software development methodology** (process, model, or life cycle) is a framework that is used to structure, plan, and control the process of developing software systems.
- Software development process is considered to be a life cycle of a software product.

UCSC

Software Development Life Cycle

- SDLC: Software Development Life Cycle has the following stages of software development:
    1. Requirements Gathering and Specification
    2. Design
    3. Implementation
    4. Testing
    5. Delivery
    6. Maintenance

UCSC

Members of a Software Project Team

1. Project manager
2. Systems analyst
3. Designer
4. Programmer
5. Tester (Quality Assurance)
6. Technical clerk

Project Success Criteria

- Deliver the software to the customer at the agreed time.
- Keep overall costs within budget.
- Deliver software that meets the customer’s expectations.
- Maintain a happy and well-functioning development team.

Project Failure Causes

- Unrealistic or unarticulated project goals
- Inaccurate estimates of needed resources
- Badly defined system requirements
- Poor reporting of the project's status
- Unmanaged risks
- Poor communication among customers, developers, and users
- Use of immature technology
- Inability to handle the project's complexity
- Sloppy development practices
- Poor project management
- Stakeholder politics
- Commercial pressures

UCSC

Software Quality Management

Why quality management is important in software ?

How quality is managed in the software?

What are product qualities and process qualities?

Real World Statistics

IBM Survey, 2000

- 55% of systems cost more than expected
- 68% overran the schedules
- 88% had to be substantially redesigned

Bureau of Labour Statistics (2001)

- for every 6 new systems put into operation, 2 cancelled
- probability of cancellation is about 50% for large systems
- average project overshoots schedule by 50%

Development Failures

Over Budget

Home Office IT project millions over budget

Home Office (UK) IT project run by Bull Information Systems is expected to blow its budget by millions of pounds and is hampered by a restrictive contract, according to a leaked report. The National Audit Office Report is expected to reveal damning evidence that the project to implement two systems – the National Probation Service Information System, and the Case Record and Management System will cost 118m pounds by the end of the year, 70% over its original budget.

www.computing.co.uk/News/111627

Over Schedule

New air traffic system is already obsolete

National Air Traffic Services (*Nats*) is already looking at replacing the systems at its new control center at Swanwick in Hampshire, even though the system doesn't become operational until next week. This project is six years late and 180m pounds over budget. Swanwick was originally meant to be operational by 1997, but problems with the development of software by Lockheed Martin caused delays, according to *Nats*.

www.vnunet.com/News/1128597

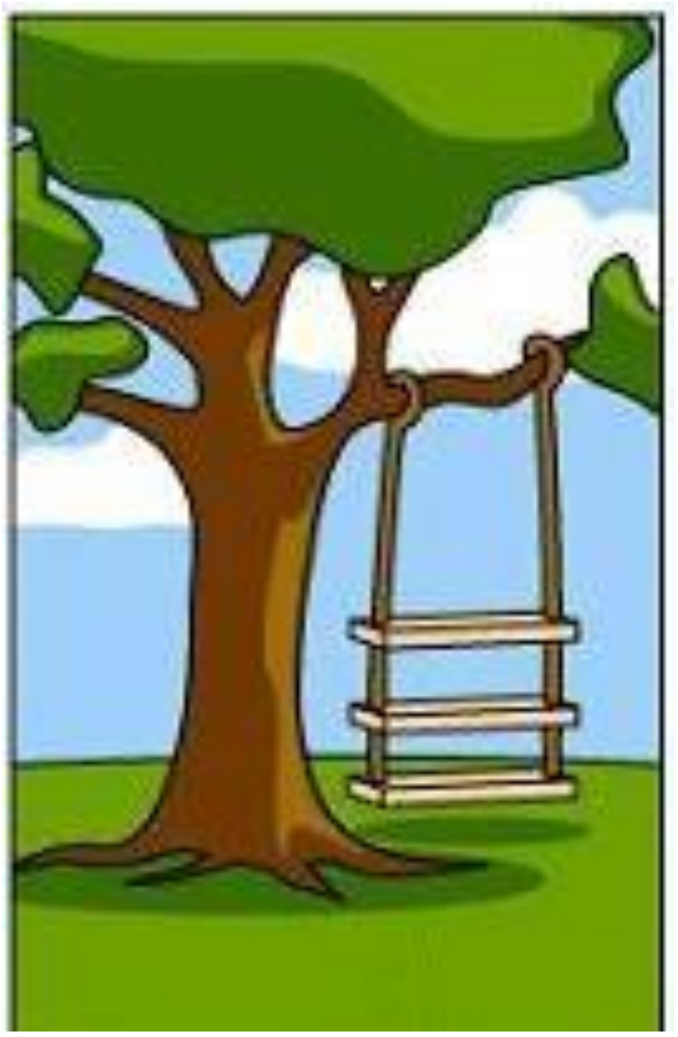
UCSC

Software Crisis

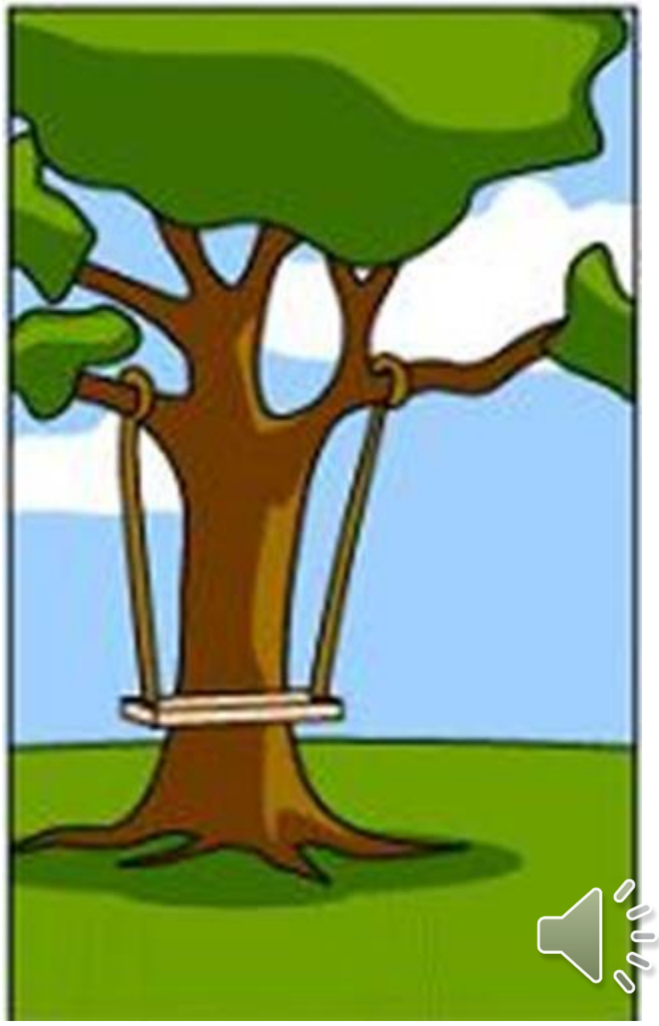
- The term was coined in early 70's
- The causes of the software crisis were linked to the overall complexity of hardware and the software development process.
- The crisis manifested itself in several ways:
  - Projects running over-budget.
  - Projects running over-time.
  - Software was very inefficient.
  - Software was of low quality.
  - Software often did not meet requirements.
  - Projects were unmanageable and code difficult to maintain.
  - Software was never delivered.

A typical software project:

How the Customer Described it



How the Business Analyst Understood it

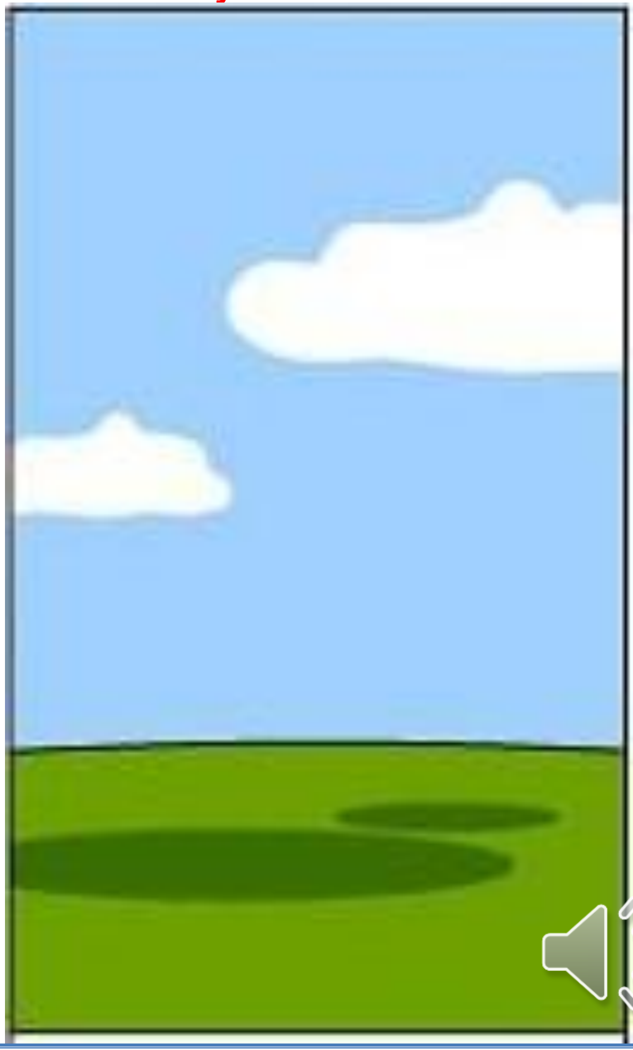


How the Architect Designed it

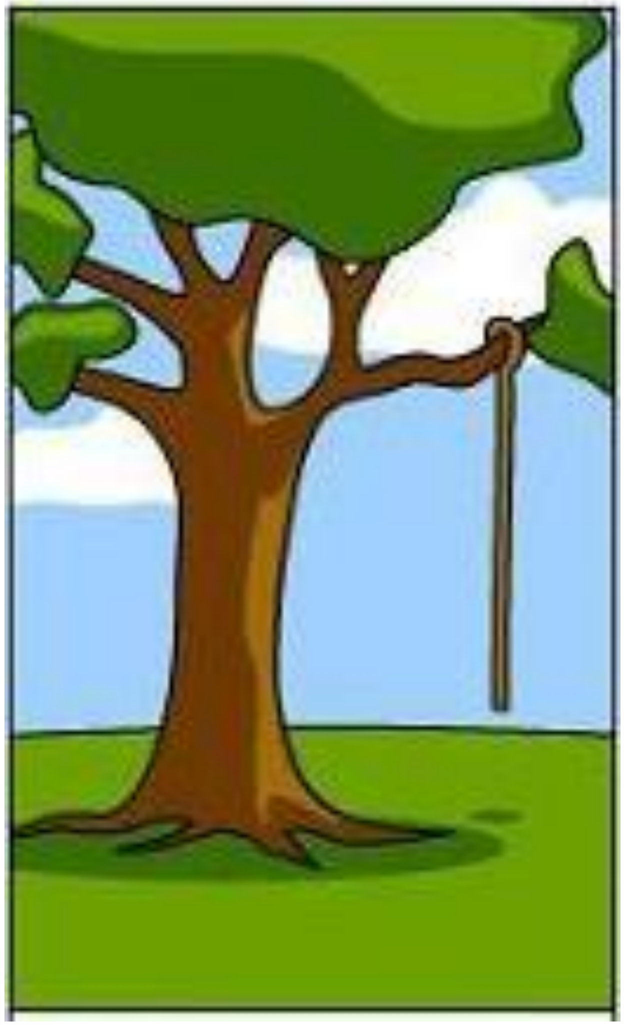
How the Developers coded it

How the Marketing team described it to the customer

How the Project was Documented

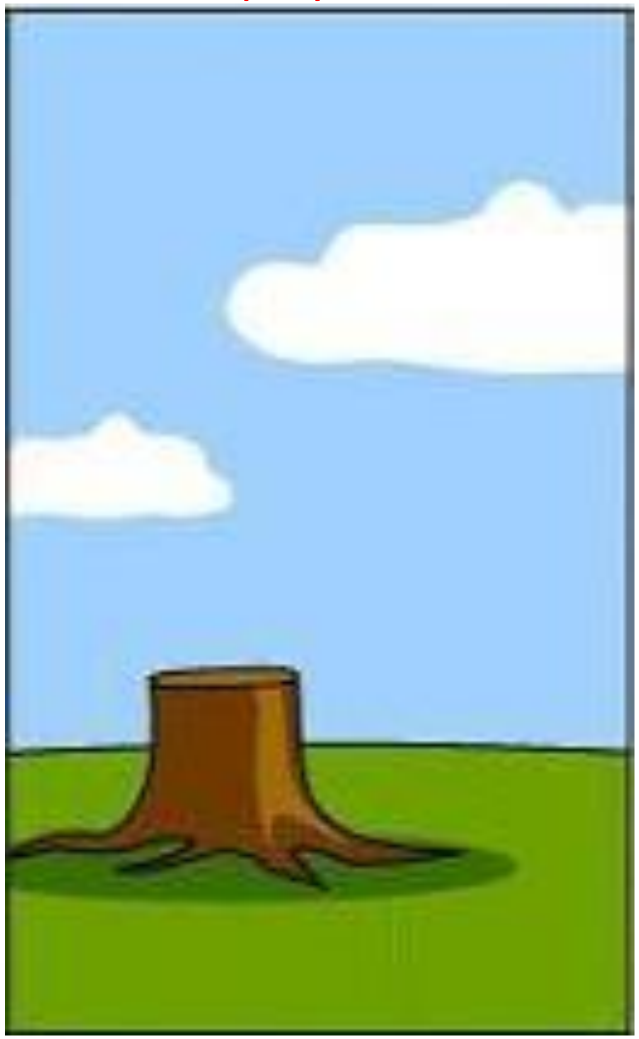


How it was Deployed at Customer Site



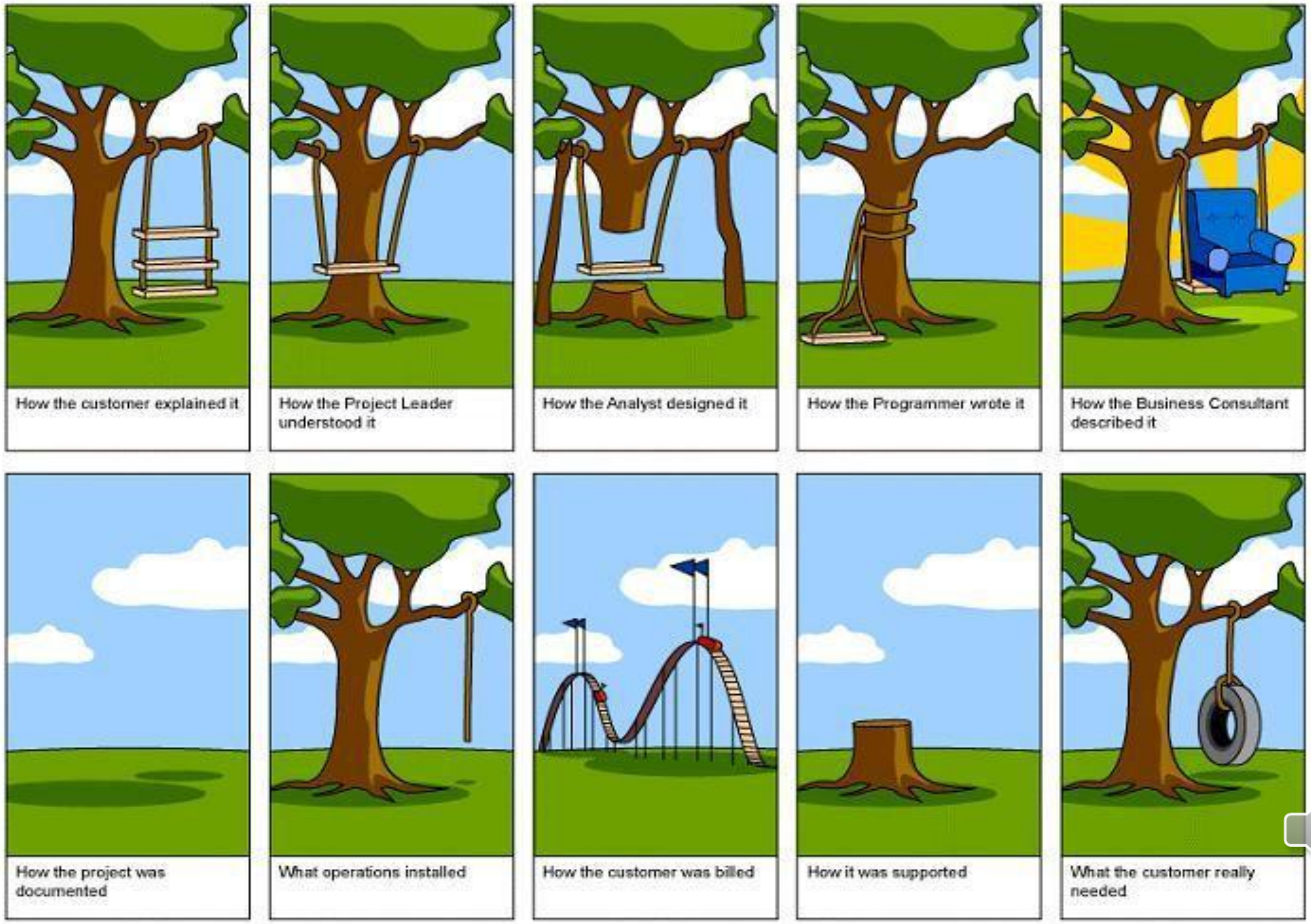
How the Customer was Billed

How it was Supported after Deployment



What Customers Really wanted!

How the Swing Project happened!!!


How the customer explained it
How the Project Leader understood it
How the 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

UCSC

The Solution: Software Engineering

- Software engineering is concerned with theories, methods and tools for professional software development
- Greater emphasis on systematic , scientific development (engineering) of Software products
- Usage in computer assistance in software development (CASE)

UCSC

The Solution: Software Engineering

- A concentration on finding out the user’s requirements
- Formal/Semi Formal specification of the requirements of a system
- Demonstration of early version of a system (prototyping)
- Greater emphasis on development of error free easy to understand code

What is software engineering?

- An engineering discipline that is concerned with all aspects of software production
- Software engineers should adopt a systematic and organized approach to their work and use appropriate tools and techniques depending on the problem to be solved, the development constraints and the resources available

UCSC

Software Engineering: Definitions

Simple Definition:

Designing, building and maintaining large software systems

A generic definition:

Use of systematic, engineering approach in all stages of software development and project management to develop high quality and economical software using appropriate software tools

UCSC

Software Engineering: Definitions

‘Software engineering is concerned with the theories, methods and tools for developing, managing and evolving software products’

– I Sommerville

‘The practical application of scientific knowledge in the design and construction of computer programs and the associated documentation required to develop, operate and maintain them’

– B.W.Boehm

Software Engineering: Definitions

'The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines’

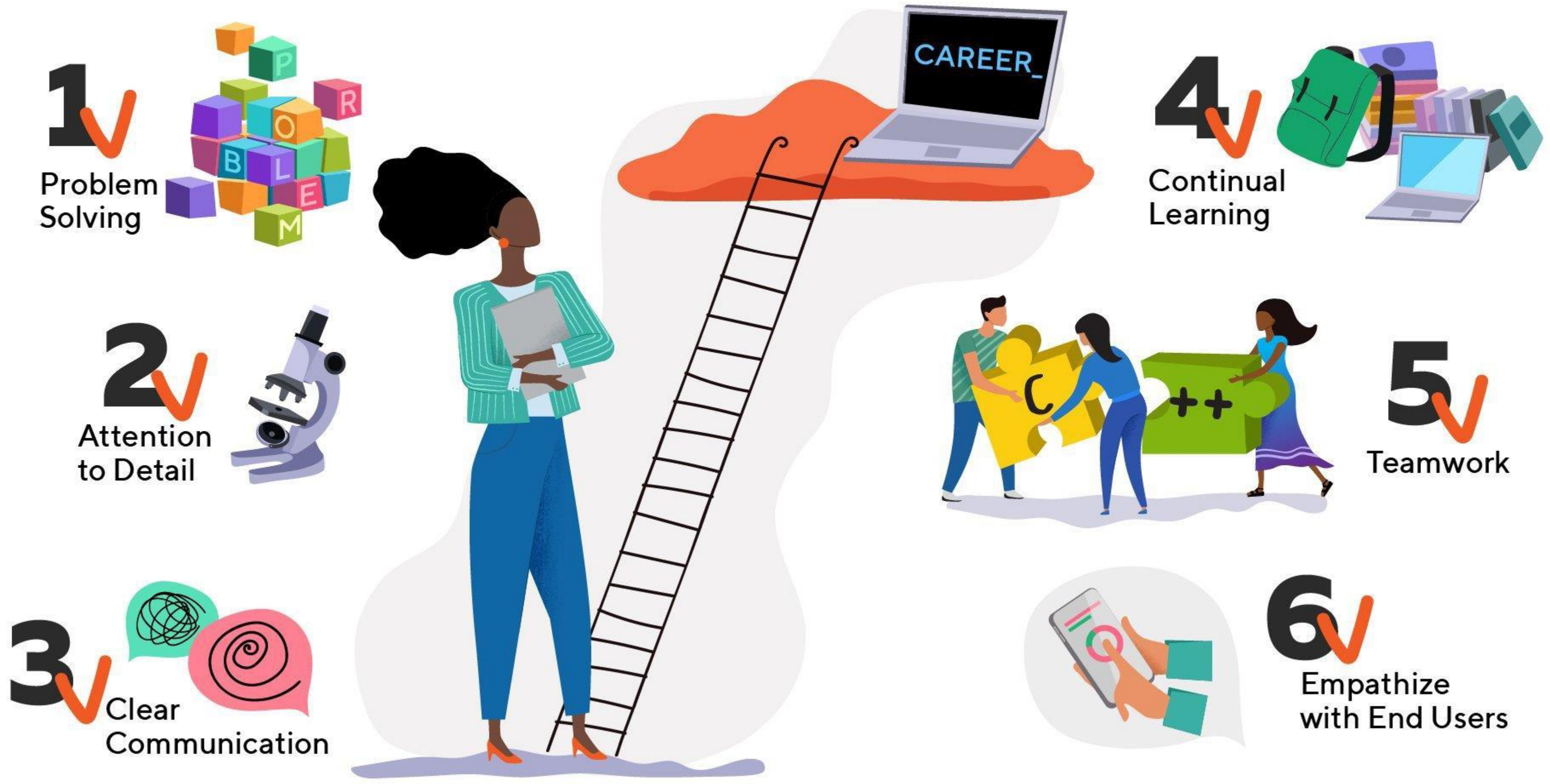
– F.L. Bauer

‘The application of systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software’

– IEEE Standard 610.12

Software Engineer vs Developer

Top Skills for Software Engineers



What makes software special?

- The main difference in software engineering compared to other engineering disciplines can be listed as below;
  1. It is difficult for a customer to specify requirements completely
  2. It is difficult for the developer to understand fully the customer needs
  3. Software requirements change regularly
  4. Software is primarily intangible; much of the process of creating software is also intangible, involving experience, thought and imagination
  5. It is difficult to test software exhaustively

UCSC

Distribution of software cost over life cycle

$$\left.\begin{array}{l}\text{1. Requirements capture}\\ \text{2. Requirement specification}\\ \text{3. Design}\\ \text{4. Implementation}\end{array}\right\}\ \mathbf{40\%}$$

$$\left.\begin{array}{l}\text{5. Testing}\\ \text{6. Maintenance}\end{array}\right\}\ \mathbf{60\%}$$

Software Problems

1. Time Schedules and cost estimates of many software projects are grossly inaccurate
2. Software is costly
3. The quality of software is not satisfactory
4. Software is difficult to maintain
5. The productivity of software people is not satisfactory to meet the demand

Problems of software development

- Large software is usually designed to solve 'wicked' problems
- Software engineering requires a great deal of coordination across disciplines
  - Almost infinite possibilities for design trade-offs across components
  - Mutual distrust and lack of understanding across engineering disciplines
- Systems must be designed to last many years in a changing environment.
- The process of efficiently and effectively developing requirements.
- Tooling required to create the solutions, may change as quick as the clients mind.

Problems of software development

- User expectations:
  - User expectations increase as the technology becomes more and more sophisticated
- The mythical man-month factor:
  - Adding personnel to a project may not increase productivity
  - Adding personnel to a late project will just make it later

Problems of software development

- Communications:
  - Communications among the various constituencies is a difficult problem. Sometimes different constituencies speak completely different languages.

UCSC

Problems of software development

- Project characteristics:
  - size / complexity
  - novelty of the application
  - response-time characteristics
  - security requirements
  - user interface requirements
  - reliability / criticality requirements

Professional and ethical responsibility

- Software engineering involves wider responsibilities than simply the application of technical skills.
- Software engineers must behave in an honest and ethically responsible way if they are to be respected as professionals.
- Ethical behavior is more than simply upholding the law.

Issues of professional responsibility

- Confidentiality
  - Engineers should normally respect the confidentiality of their employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.
- Competence
  - Engineers should not misrepresent their level of competence. They should not knowingly accept work which is out with their competence.

UCSC

Issues of professional responsibility

- Intellectual property rights
  - Engineers should be aware of local laws governing the use of intellectual property such as patents, copyright, etc. They should be careful to ensure that the intellectual property of employers and clients is protected.
- Computer misuse
  - Software engineers should not use their technical skills to misuse other people's computers. Computer misuse ranges from relatively trivial (game playing on an employer's machine, say) to extremely serious (dissemination of viruses).

ACM/IEEE Code of Ethics

- The professional societies in the US have cooperated to produce a code of ethical practice.
- Members of these organisations sign up to the code of practice when they join.
- The Code contains eight Principles related to the behaviour of and decisions made by professional software engineers, including practitioners, educators, managers, supervisors and policy makers, as well as trainees and students of the profession.

The ACM/IEEE Code of Ethics

Software Engineering Code of Ethics and Professional Practice

ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices

PREAMBLE

The short version of the code summarizes aspirations at a high level of the abstraction; the clauses that are included in the full version give examples and details of how these aspirations change the way we act as software engineering professionals. Without the aspirations, the details can become legalistic and tedious; without the details, the aspirations can become high sounding but empty; together, the aspirations and the details form a cohesive code.

Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession. In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to the following Eight Principles:

Ethical principles

1. **PUBLIC** - Software engineers shall act consistently with the public interest.

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.

3**. PRODUCT** - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.

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

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

6. **PROFESSION** - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.

7. **COLLEAGUES** - Software engineers shall be fair to and supportive of their colleagues.

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.

Software Engineering Fundamentals

Some fundamental principles apply to all types of software system, irrespective of the development techniques used:

- Systems should be developed using a managed and understood development process. Different processes are used for different types of software.
- Dependability and performance are important for all types of system.
- Understanding and managing the software specification and requirements (what the software should do) are important.
- Where appropriate, you should reuse software that has already been developed rather than write new software.

Internet/Web-based Software Engineering

- The Web is now a platform for running application and organizations are increasingly developing web-based systems rather than local systems.
- Web services allow application functionality to be accessed over the web.
- Cloud computing is an approach to the provision of computer services where applications run remotely on the ‘cloud’.
- Users do not buy software but pay according to use.

Web-based software engineering

Web-based systems are complex distributed systems but the fundamental principles of software engineering are as applicable to them as they are to any other types of system.

The fundamental ideas of software engineering apply to web-based software in the same way that they apply to other types of software system.

Web-based software engineering

Software reuse

Software reuse is the dominant approach for constructing web-based systems. When building these systems, you think about how you can assemble them from pre-existing software components and systems.

Incremental and agile development

Web-based systems should be developed and delivered incrementally. It is now generally recognized that it is impractical to specify all the requirements for such systems in advance.

Web software engineering

Service-oriented systems

Software may be implemented using service-oriented software engineering, where the software components are stand-alone web services.

Rich interfaces

Interface development technologies such as AJAX and HTML5 have emerged that support the creation of rich interfaces within a web browser.

FAQs

Question	Answer
What is software?	Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.
What are the attributes of good software?	Good software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable.
What is software engineering?	Software engineering is an engineering discipline that is concerned with all aspects of software production.
What are the fundamental software engineering activities?	Software specification, software development, software validation and software evolution.
What is the difference between software engineering and computer science?	Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.
What is the difference between software engineering and system engineering?	System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. Software engineering is part of this more general process.

UCSC

FAQs

Question	Answer
What are the key challenges facing software engineering?	Coping with increasing diversity, demands for reduced delivery times and developing trustworthy software.
What are the costs of software engineering?	Roughly 60% of software costs are development costs, 40% are testing costs. For custom software, evolution costs often exceed development costs.
What are the best software engineering techniques and methods?	While all software projects have to be professionally managed and developed, different techniques are appropriate for different types of system. For example, games should always be developed using a series of prototypes whereas safety critical control systems require a complete and analyzable specification to be developed. You can't, therefore, say that one method is better than another.
What differences has the web made to software engineering?	The web has led to the availability of software services and the possibility of developing highly distributed service- based systems. Web-based systems development has led to important advances in programming languages and software reuse.

UCSC