IT3205: Fundamentals of Software Engineering (Compulsory)

BIT – 2nd Year

Semester 3

Learning Outcome

After successfully completing this course students should be able to;

- 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

Outline of Syllabus

1. Introduction
2. Software Development Process Models
3. Requirements Analysis & Specification
4. Design
5. Coding
6. Software Testing and Quality Assurance
7. Software Maintenance
8. Software Project Management

Main References

1. 

Software Engineering by Ian Sommerville, 7th edition, Addison-Wesley, 2006.

2. 

Software Engineering: A practitioner's approach by Roger S. Pressman, 6th edition, McGraw-Hill International edition, 2005.

3. http://www.softwareengineering-9.com

IT3205: Fundamentals of Software Engineering

Introduction

Duration: 4 hours

Learning Objectives

- Describe what software is, different types of software and software quality attributes
- Describe with the problems associated with software and software development
- Define what software engineering is and explain why it is important
- State some professional issues related to software development

Detailed Syllabus

1.1 Software

- 1.1.1 What is software?
- 1.1.2 Types of software
- 1.1.3 Characteristics of Software
- 1.1.4 Attributes of good software

1.2 Software Engineering

- 1.2.1 What is software engineering?
- 1.2.2 Software engineering costs
- 1.2.3 What are the key challenges facing software engineering?
- 1.2.4 Systems engineering & software Engineering
- 1.2.5 Professional Practice

1.1.1 WHAT IS SOFTWARE?

What is software?

- Instructions given to a computer (computer programs)
- Software is a general term for the various kinds of programs used to operate computers and related devices
- It can be;
  - System Software
  - Application Software

What is software?

- Software is the set of instructions that makes the computer work.

Example - When you type in words via the keyboard, the software is responsible for displaying the correct letters, in the correct place on the screen.

What is software?

- Software is held either on your computer's hard disk, CDROM, DVD or on a diskette (floppy disk) and is loaded (i.e. copied) into the computer's RAM (Random Access Memory), as and when required.

1.1.2 TYPES OF SOFTWARE

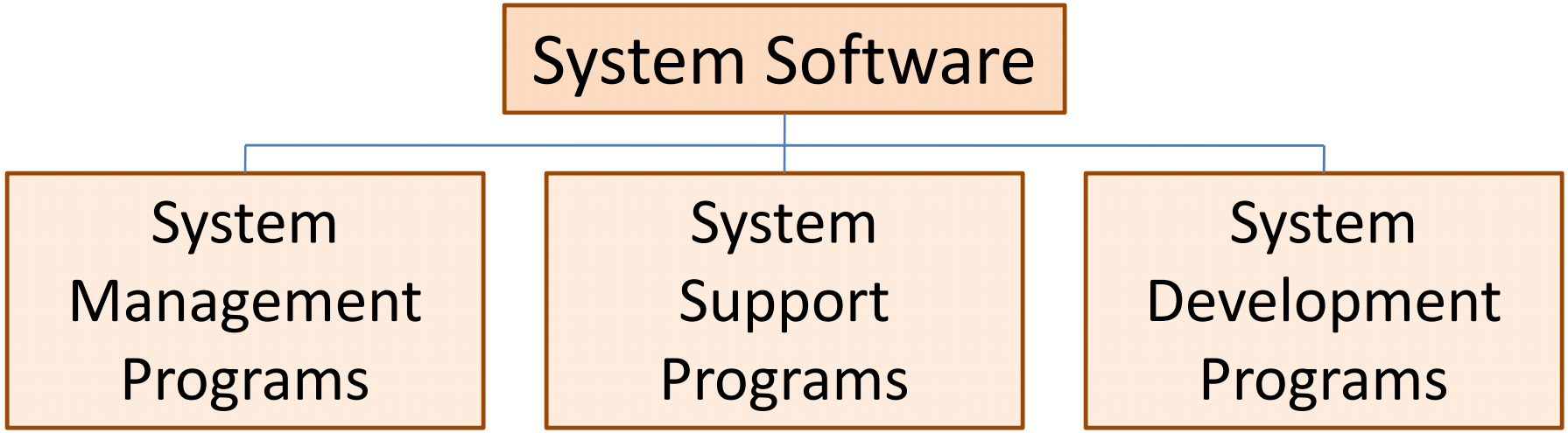
Main Types of Software

- 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

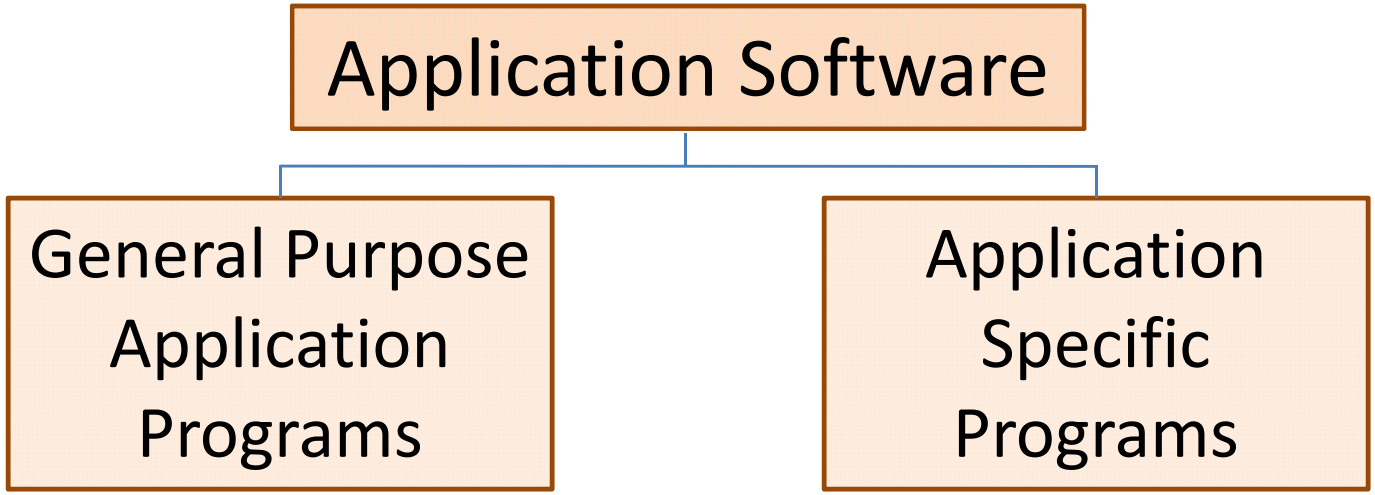


- Operating Systems
- Operating Environments
- Database Management Systems

- System Utilities
- Performance Monitors
- Security Monitors

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

Application Software



- 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

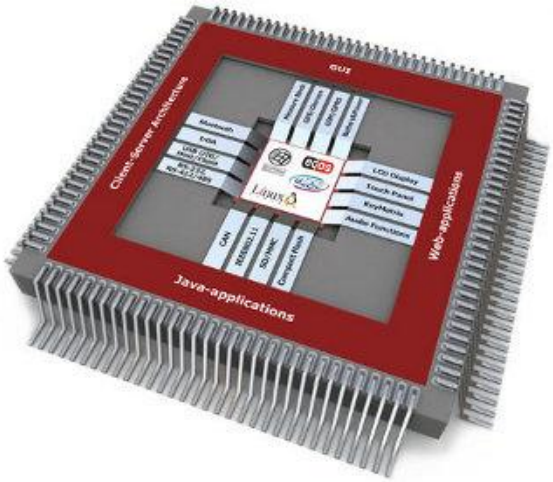
- **System Software:**
  - System software is a collection of programs is written to service the other programs.

Eg: Operating system component, drivers, telecommunication process

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.

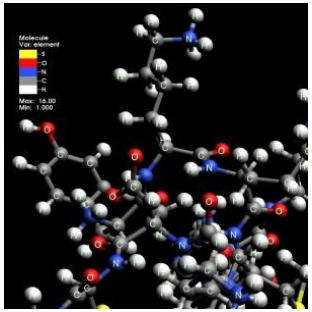
- **Artificial intelligence software:**

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

Common Software Types

- **Engineering and scientific software:**

  They have been characterized by number crunching algorithms



- **Personal computer software:**

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

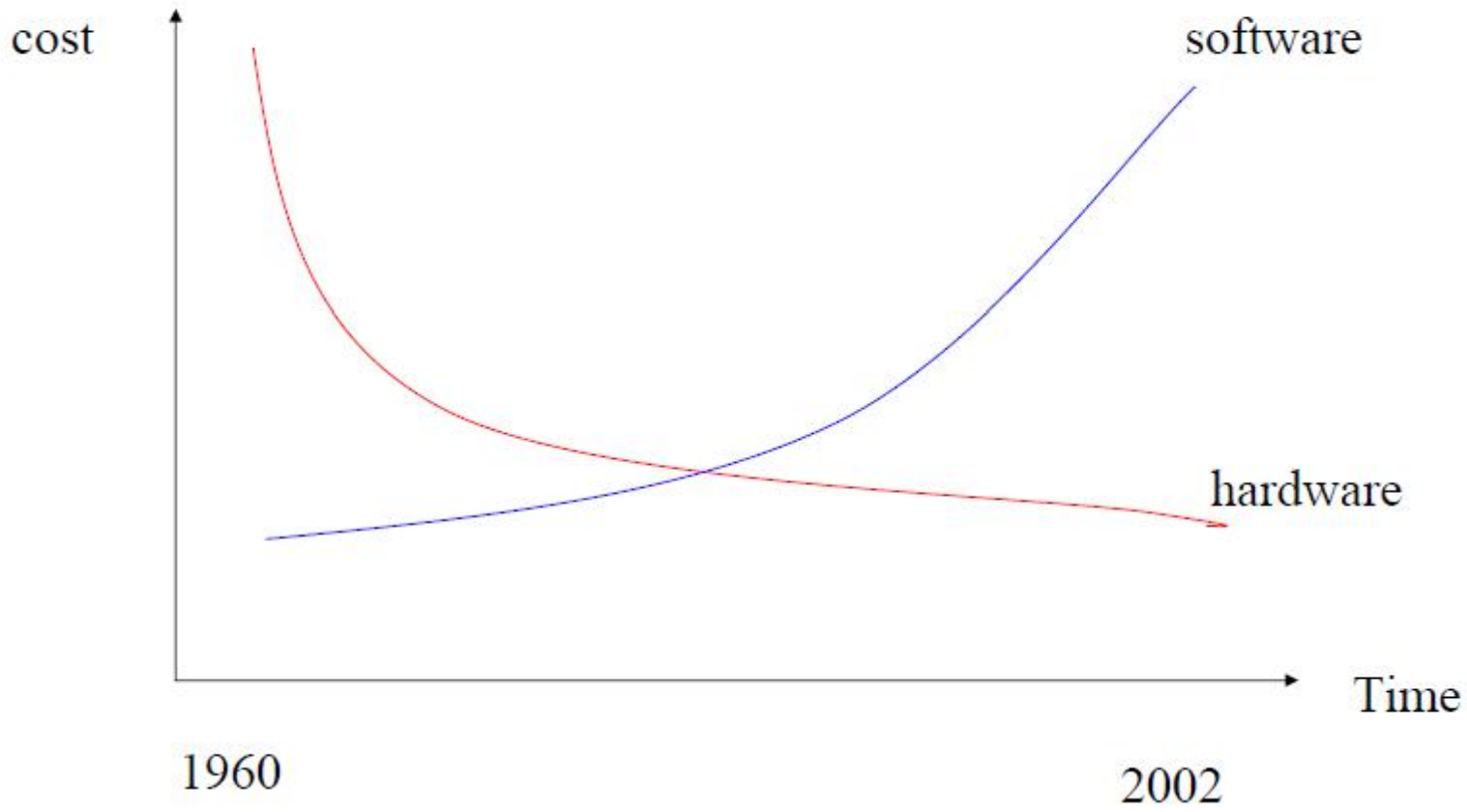
  Word processing, spreadsheets, computer graphic, multimedia and db management

1.1.3 CHARACTERISTICS OF SOFTWARE

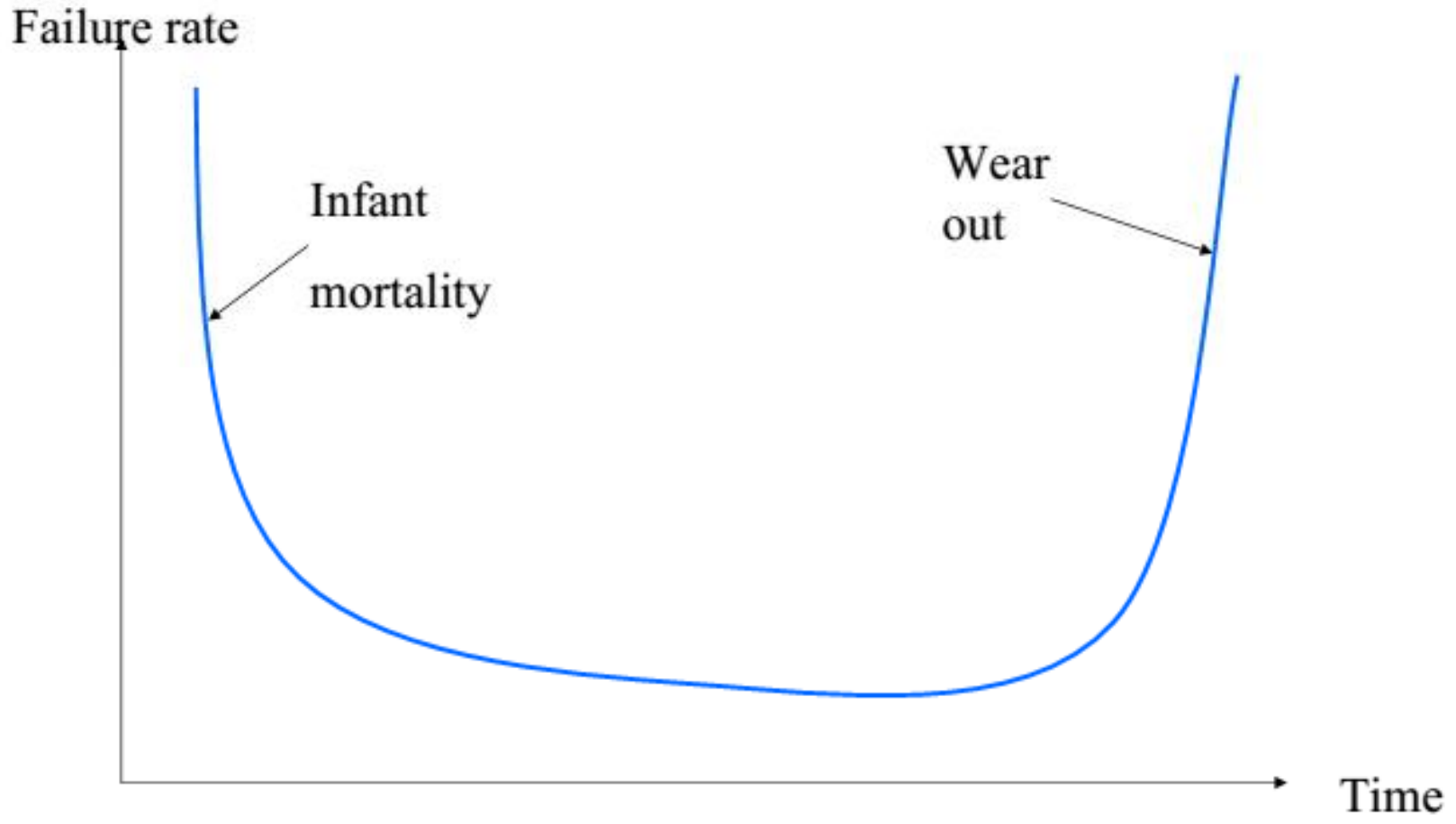
Characteristics of Software

- Intangibility
  - Cannot touch software
- Increase use will not introduce any defects
- Software is configurable
  - able to build software by combining a basic set of software components in different ways
  - One can change the product easily by re-implementing it without changing the design
- Custom built
  - Most software are made upon order

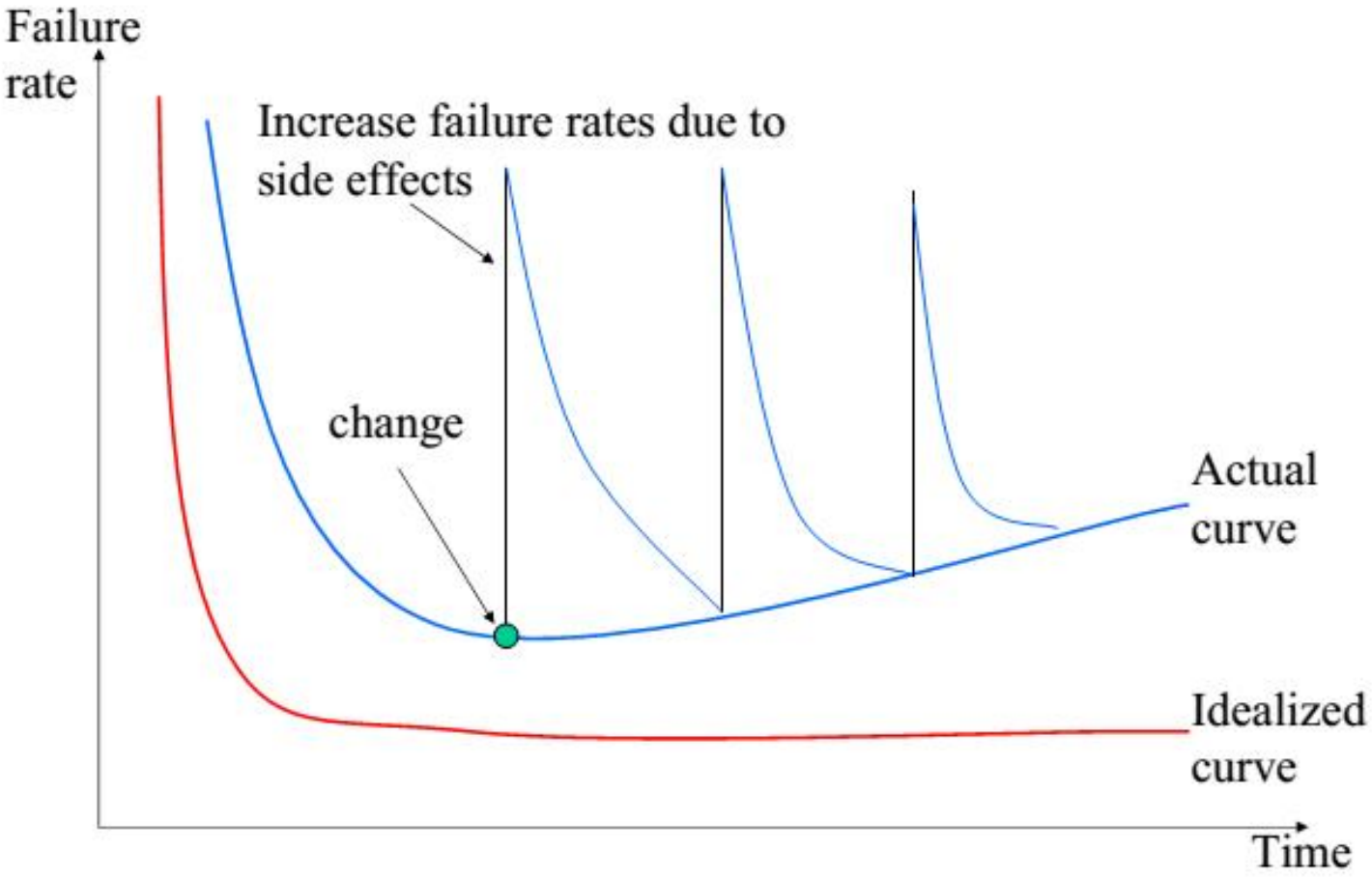
Cost of Hardware vs. Software



Failure curve for hardware (Pressman)



Failure curve for software (Pressman)



1.1.4 ATTRIBUTES OF GOOD SOFTWARE

Software Quality

- The degree in which software possesses a desired combination of quality attributes

- The software should deliver the required functionality and performance to the user and should be maintainable, dependable and acceptable

Software Quality

- Maintainability
  - Software must evolve to meet changing needs
- Dependability
  - Software must be trustworthy
- Efficiency
  - Software should not make wasteful use of system resources
- Acceptability
  - Software must accepted by the users for which it was designed. This means it must be understandable, usable and compatible with other systems

Bohem's Classification

- Current Usefulness
  - The qualities expected from a software system in user's point of view
- Potential Usefulness
  - The qualities expected from a software system

Current usefulness

- Efficiency
  - Software should not make wasteful use of system resources
- Reliability
- Usability
- Correctness
  - The degree with which software adheres to its specified requirements
- User friendliness
- Robustness

Potential usefulness

- Maintainability
  - Software must evolve to meet changing needs. The ease with which changes can be made to satisfy new requirements or to correct deficiencies
- Modularity
- Reusability
  - The ease with which software can be reused in developing other software
- Portability
  - The ease with which software can be used on computer configurations other than its current one

McCall's Classification

- Product operation
- Product revision
- Product transition

Product Operation

- Efficiency
  - The degree with which software fulfills its purpose without waste of resources
- Correctness
- User friendliness
- Usability
- Reliability
  - The frequency and criticality of software failure, where failure is an unacceptable effect or behavior occurring under permissible operating conditions
- Robustness

Product Revision

- Maintainability
- Flexibility
- Testability

Product Transition

- Interoperability
- Reusability
- Portability

1.2.1 INTRODUCTION TO SOFTWARE ENGINEERING

Need for Software Engineering

- The economies of ALL developed nations are dependent on software.
- More and more systems are software controlled
- Expenditure on software represents a significant fraction of GNP in all developed countries

Need for Software Engineering

- Software is found in products and situations where very high reliability is expected
  - E.g. Monitoring and controlling Nuclear power plants
- Contain millions of lines of code
- Comparably more complex

Thus, need a systematic process to produce high quality software product

The Solution – Software Engineering

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

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

Software Engineering - Definitions

Simple Definition: *Designing, building and maintaining large software systems*

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

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

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

Members of a software engineering team

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

1.2.2 SOFTWARE ENGINEERING COSTS

Software Engineering Costs

Distribution of costs across the different activities in the software process depends on the process used and the type of software that is being developed.

- Eg: Real-time software usually requires more extensive validation and testing than web-based systems.

Software Engineering Costs

- In the waterfall approach, the cost of specification, design, implementation and integration are measured separately.
- System integration and testing is the most expensive development activity.
- Normally this is about 40% of the total development costs

Development Failures

IBM Survey, 2002

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

Bureau of Labour Statistics (2004)

- 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 - Real Examples;

- Over Budget

Home Office IT project millions over budget

Home Office (UK) IT project runby 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 Reportis 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

Development Failures - Real Examples;

- Over Schedule

New air traffic system is already obsolete

National Air Traffic Services (Nats) is already looking at replacing the systems at its newcontrol 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

Development Failures - Real Examples;

- Safety

London Ambulance Dispatching System

The full introduction of the computer system effectively did away with the radio and telephone calls to stations, with the computer dispatching crews to answer calls. But within hours, during the morning rush, it became obvious to crews and control room staff that calls were going missing in the system; ambulances were arriving late or doubling up on calls. Distraught emergency callers were also held in a queuing system which failed to put them through for up to 30 minutes. Chris Humphreys, Nupe's divisional officer, said that it was hard to verify how many people might have died because of the delays but it could be as many as 20.

Causes: The managers who produced the software were naïve. They made a terrible mistake of trying to go on-line abruptly, without running the new and old systems together for a while

Development Failures - Real Examples;

- Programming/testing Error

Ariane 5 (June 1996)

It took the European Space Agency 10 years and $7 billion to produce Ariane 5, a giant rocket capable of hurling a pair of three ton satellites into orbit.

At 39 seconds after launch, as the rocket reached an altitude of two and a half miles, a self-destructmechanism finished off Ariane 5, along with its payload of two expensive and uninsured scientific satellites. The rocket was makingan abrupt course correction that was not needed, compensating for a wrong turn that had not taken place.

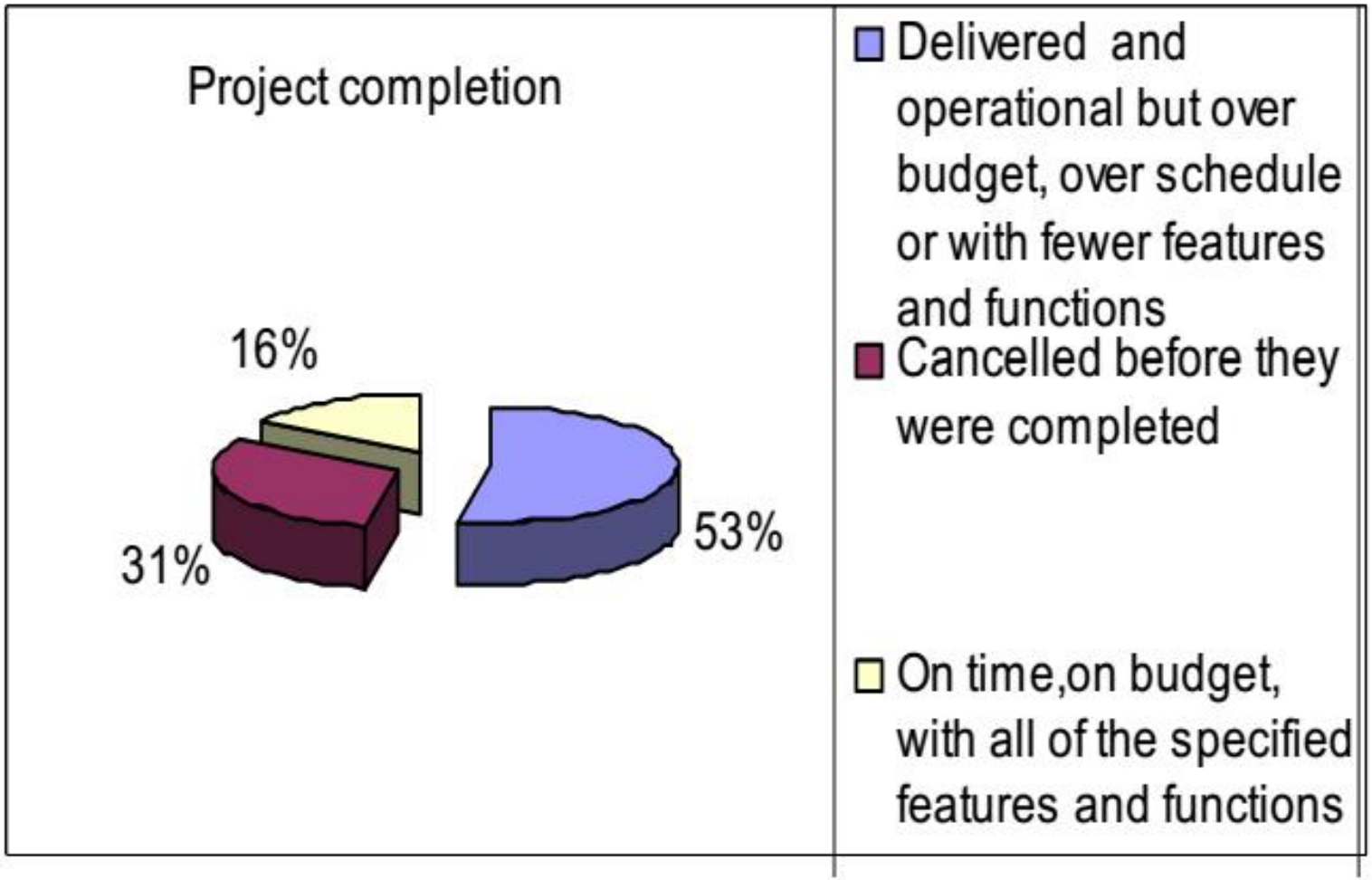
Development Failures - Real Examples;

- Programming/testing Error

Ariane 5 (June 1996)

The cause: Steering was controlled by the on-board computer, which mistakenly thought the rocket needed a course change because of the numbers, which in fact was an error, coming from the inertial guidance system. The guidance system had in fact shut down 36.7 seconds after launch,when the guidance system's own computer tried to convert one piece of data – the sideways velocity of the rocket – from a 64 bit format to a 16 bit format = overflow error.

Statistics


Project completion
16%
31%
53%
Delivered and operational but over budget, over schedule or with fewer features and functions
Cancelled before they were completed
On time,on budget, with all of the specified features and functions

1.2.3 KEY CHALLENGES FACING SOFTWARE ENGINEERING

Key challenges facing Software Engineering

- Heterogeneity
  - Developing techniques for building software that can cope with heterogeneous platforms and execution environments
- Delivery
  - Developing techniques that lead to faster delivery of software
- Trust
  - Developing techniques that demonstrate that software can be trusted by its users

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

Problems of software development

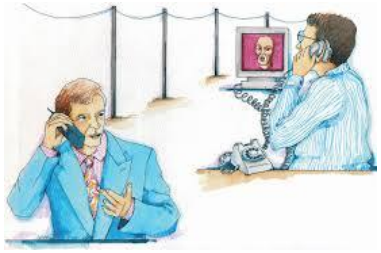
- Systems must be designed to last many years in a changing environment.
- The process of efficiently and effectively developing requirements.
- Tooling required to create the solutions, may change as quick as the clients mind.

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. For example, developers may not have the domain knowledge of clients and / or users. The larger the project, the more difficult the communications problems become.



Problems of software development

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

1.2.4 SYSTEMS ENGINEERING & SOFTWARE ENGINEERING

software engineering vs. 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 process concerned with developing the software infrastructure, control, applications and databases in the system.

- System engineers are involved in system specification, architectural design, integration and deployment.

1.2.5 PROFESSIONAL PRACTICE

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.

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).