

Overview and History of Software Engineering

Copyright © K.Goseva 2006 CS 230 Introduction to Software Engineering Slide 1

---

---

---

---

---

---

---

---

Outline

- Historical aspects - software crisis
- Software product
- Software process
- Software fault and failures
- Team aspects
- Structured versus object-oriented paradigm

Copyright © K.Goseva 2006 CS 230 Introduction to Software Engineering Slide 2

---

---

---

---

---

---

---

---

Historical Aspects

- A NATO study group coined the term software engineering in 1967
- 1968 NATO Conference in Garmisch, Germany
- Aim - to solve the *SOFTWARE CRISIS*
  - Software is delivered
    - o Late
    - o Over budget
    - o With residual faults

Copyright © K.Goseva 2006 CS 230 Introduction to Software Engineering Slide 3

---

---

---

---

---

---

---

---

### Examples of failures due to software

- Excessive radiotherapy doses (1985-1987)
- 9 hours outage of the long-distance in USA (1990)
- Scud missile missed by the Patriot (1991)
- Ariane 5 crash (1996)
- 8 hours delay in opening the London Stock Exchange (2000)

Copyright © K.Goseva 2006

CS 230 Introduction to Software Engineering

Slide 4

---

---

---

---

---

---

---

---

### Software Engineering

- Software engineering is a discipline whose aim is the production of fault-free software, delivered on time and within budget, that satisfies the user's needs
- To achieve these goals, a software engineer has to acquire a broad range of skills, both technical and managerial

Copyright © K.Goseva 2006

CS 230 Introduction to Software Engineering

Slide 5

---

---

---

---

---

---

---

---

### Software Characteristics

- Software is developed or engineered, it is not manufactured in classical sense
- Most software continues to be custom built, although the industry is moving toward component-based assembly
- Software is complex
- Software doesn't wear out

Copyright © K.Goseva 2006

CS 230 Introduction to Software Engineering

Slide 6

---

---

---

---

---

---

---

---

### Software Poses Challenges

- How do we ensure the quality of the software that we produce?
  - Why can't we find and fix all faults (bugs) before the software is released?
- How do we meet the budget and avoid disastrous time delays?
  - Why the development costs so high?
  - Why does it take so long to get software finished?
- How do we upgrade software?
- How do we successfully institute new software technologies?

Copyright © K.Goseva 2006      CS 230 Introduction to Software Engineering      Slide 7

---

---

---

---

---

---

---

---

### Economic Aspects

- Coding technique  $CT_{new}$  is 10% faster than currently used technique  $CT_{old}$ . Should it be used?
- Choose technique that reduce long - term cost
  - Consider costs in introducing  $CT_{new}$  into organization
  - Consider the effect of  $CT_{new}$  on maintenance

Copyright © K.Goseva 2006      CS 230 Introduction to Software Engineering      Slide 8

---

---

---

---

---

---

---

---

### Software Product

- Software is a set of items or objects that form a "configuration" that includes
  - Programs that when executed provide desired function and performance
  - Data structures that enable programs to adequately manipulate information
  - Documents that describe the operation and use of the programs

Copyright © K.Goseva 2006      CS 230 Introduction to Software Engineering      Slide 9

---

---

---

---

---

---

---

---

**Software Applications**

- System software – collection of programs written to service other programs
  - Heavy interaction with computer hardware, multiple users, concurrent operation, resource sharing, sophisticated process management, complex data structures, multiple external interfaces
  - Examples: operating system components, drivers, telecommunications processors, compilers, editors, file management utilities
- Application software – standalone programs that solve specific business or technical need
  - Examples: data processing applications, point-of-sale transaction processing, real-time manufacturing process control

Copyright © K.Goseva 2006      CS 230 Introduction to Software Engineering      Slide 10

---

---

---

---

---

---

---

---

**Software Applications (contd)**

- Business software – business information processing
  - Management information system (MIS) that accesses one or more databases containing business information (e.g., payroll, inventory)
- Engineering and scientific software (e.g., numerical estimations, simulation, etc.)
- PC software – (word processing, spreadsheets, computer graphics, multimedia, entertainment, personal and business financial applications, etc.)
- Web-based software
  - A set of linked hypertext files that present information using text and graphics, e-commerce, B2B applications

Copyright © K.Goseva 2006      CS 230 Introduction to Software Engineering      Slide 11

---

---

---

---

---

---

---

---

**Software Applications (contd)**

- Real-time software – monitors, analyzes, and controls real-world events as they occur in real-time
  - Response time typically ranges from 1 millisecond to 1 second
- Embedded software – control products and systems for consumer and industrial markets
- Artificial intelligence software – uses non-numerical algorithms to solve complex problems
  - Applications: robotics, expert systems, pattern recognition, adaptive control.
- Ubiquitous computing – growth of wireless networking, use of small mobile devices, laptops, etc.

Copyright © K.Goseva 2006      CS 230 Introduction to Software Engineering      Slide 12

---

---

---

---

---

---

---

---

## Software Process

Software process is the way we produce software product (a nontrivial piece of software)

1. Requirements phase
2. Specification phase
3. Design phase
4. Implementation phase
5. Integration phase (in parallel with 4)
6. Maintenance phase
7. Retirement

Copyright © K.Goseva 2006

CS 230 Introduction to Software Engineering

Slide 13

---

---

---

---

---

---

---

---

## Software Process (contd)

- Testing is not a separate phase, but an activity that takes place through software production
- Testing occurs towards the end of each phase (verification) and before the product is handed over to the client (validation)
- *Although there are times when the testing predominates, there should never be the times when no testing is being performed*

Copyright © K.Goseva 2006

CS 230 Introduction to Software Engineering

Slide 14

---

---

---

---

---

---

---

---

## Software Process (contd)

- There is no separate documentation phase
- Documentation for each phase must be completed by the team responsible for that phase, before the next phase starts
- Documentation must be updated continually to reflect the current version of the product

Copyright © K.Goseva 2006

CS 230 Introduction to Software Engineering

Slide 15

---

---

---

---

---

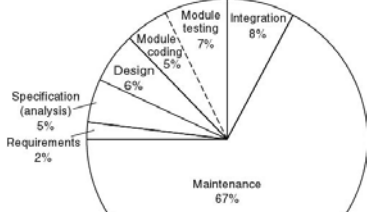
---

---

---

### Approximate Relative Cost of Each Phase

- 1976–1981 data
- Maintenance constitutes 67% of total cost



Copyright © K.Goseva 2006 CS 230 Introduction to Software Engineering Slide 16

---

---

---

---

---

---

---

---

### Comparative Relative Cost of Each Phase

	Various Projects between 1976 and 1981	132 More Recent Hewlett-Packard Projects
Requirements and specification (analysis) phases	21%	18%
Design phase	18	19
Implementation phase	36	34
Integration phase	24	29

Copyright © K.Goseva 2006 CS 230 Introduction to Software Engineering Slide 17

---

---

---

---

---

---

---

---

### Good and Bad Software

- Good software is maintained - Bad software is discarded
- Different types of maintenance
  - Corrective maintenance [about 20%]
  - Enhancement
    - Perfective maintenance [about 60%]
    - Adaptive maintenance [about 20%]

Copyright © K.Goseva 2006 CS 230 Introduction to Software Engineering Slide 18

---

---

---

---

---

---

---

---

## Software fault and failures

- Failure - departure from the specific system behavior
- Fault - defect in the software that when executed under particular conditions causes a failure
- 60 to 70 % of faults are specification and design faults
- Inspection of JPL software for the NASA unmanned interplanetary space program detected (Data of Kelly, Sherif, and Hops [1992])
  - 1.9 faults per page of specification
  - 0.9 faults per page of design
  - 0.3 faults per page of code

Copyright © K.Goseva 2006

CS 230 Introduction to Software Engineering

Slide 19

---

---

---

---

---

---

---

---

## Software fault and failures (contd)

- Faults at end of the design phase of the new version of the product (Data of Bhandari et al. [1994])
  - 13% of faults from previous version of product
  - 16% of faults in new specifications
  - 71% of faults in new design
- The earlier we correct a fault, the better

Copyright © K.Goseva 2006

CS 230 Introduction to Software Engineering

Slide 20

---

---

---

---

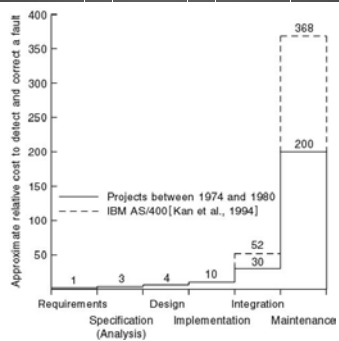
---

---

---

---

## Cost to Detect and Correct a Fault



Copyright © K.Goseva 2006

CS 230 Introduction to Software Engineering

Slide 21

---

---

---

---

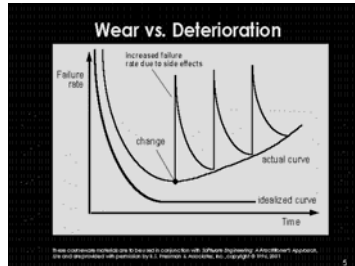
---

---

---

---

## Failure Curve for Software



Copyright © K.Goseva 2006

CS 230 Introduction to Software Engineering

Slide 22

---

---

---

---

---

---

---

---

## Team Aspects

- Products are too large to be written by one person in the available time
- Teams should be properly organized and managed
- Suppose that a single developer can complete a product in 1 year. How long it will take if it is assigned to a team of three developers?

Copyright © K.Goseva 2006

CS 230 Introduction to Software Engineering

Slide 23

---

---

---

---

---

---

---

---

## Structured versus Object-Oriented Paradigm

- Structured methods are action oriented or data oriented, but not both
- Object - software component that incorporates both data and the actions that are performed on that data
  - Example: Bank account
    - o Data: account balance
    - o Actions: deposit, withdraw, determine balance

Copyright © K.Goseva 2006

CS 230 Introduction to Software Engineering

Slide 24

---

---

---

---

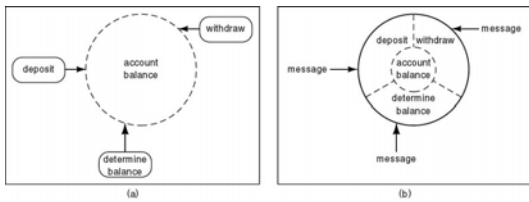
---

---

---

---

## Structured versus Object-Oriented Paradigm



- Data component of an object - Attribute (generic)
- Action component of an object - Method (generic)

Copyright © K.Goseva 2006

CS 230 Introduction to Software Engineering

Slide 25

---

---

---

---

---

---

---

---

## Structured versus Object-Oriented Paradigm

- Encapsulation – conceptual independence
  - Everything in the product that relates to the proportion of the real world modeled by that object can be found in the object itself
- Information hiding - physical independence
  - Implementation details are hidden from everything outside that object
- Responsibility-driven design (or Design by contract)
  - The way the action is carried out is entirely the responsibility of the object itself

Copyright © K.Goseva 2006

CS 230 Introduction to Software Engineering

Slide 26

---

---

---

---

---

---

---

---

## Example

Send flowers to your aunt in Iowa City

- List of all florists in Iowa City
- Which florists do the delivery
- Which one is located close to aunt's home

OR

- Call 1-800-FLOWERS and leave the entire responsibility for delivering flowers to that organization

Copyright © K.Goseva 2006

CS 230 Introduction to Software Engineering

Slide 27

---

---

---

---

---

---

---

---

## Structured versus Object-Oriented Paradigm

### Structured Paradigm

1. Requirements phase
2. Specification (analysis) phase
3. Design phase
4. Implementation phase
5. Integration phase
6. Maintenance phase
7. Retirement

### Object-Oriented Paradigm

1. Requirements phase
- 2'. Object-oriented analysis phase
- 3'. Object-oriented design phase
- 4'. Object-oriented programming phase
5. Integration phase
6. Maintenance phase
7. Retirement

Copyright © K.Goseva 2006

CS 230 Introduction to Software Engineering

Slide 28

---

---

---

---

---

---

---

---

---

---

## Warning

- Do not use the object-paradigm to enhance a product developed using the structured paradigm
  - Water and oil do not mix
- Exception: if the new part is totally disjoint
  - Example: adding a GUI (graphical user interface)

Copyright © K.Goseva 2006

CS 230 Introduction to Software Engineering

Slide 29

---

---

---

---

---

---

---

---

---

---