Software-Praktikum

WS97/98

Veranstaltungsnummer 8712

Prof. Willem-Paul de Roever
Lehrstuhl Softwaretechnologie
Christian-Albrechts-Universität zu Kiel
Institut für Informatik und Praktische Mathematik
Preußerstr. 1-9, D24105 Kiel
e-mail: wpr@informatik.uni-kiel.de
http://www.informatik.uni-kiel.de/inf/deRoever/SoftTech/

Scope of Software Engineering

Software errors and their consequences

- The $0.00 bill
- The Strategy Air Command alert november 1979 when the WWMCS (world-wide military monitoring command and control system) computer reported inability to distinguish between real and simulated missile attacks. Could have brought the world to an abrupt end
- The Therac-25 catastroph
- Patriot missile's timing error during the gulf war

JUST IN CASE YOU WANTED TO KNOW

In the case of the WWMCCS network, disaster was averted at the last minute. However, the consequences of other software faults have sometimes been tragic. For example, between 1985 and 1987 at least two patients died as a consequence of severe overdoses of radiation delivered by the Therac-25 medical linear accelerator [Leveson and Turner, 1993]. The cause was a fault in the control software.

During the 1991 Gulf War, a Scud missile penetrated the Patriot anti-missile shield and struck a barracks near Dhahran, Saudi Arabia. In all 28 Americans were killed, and 98 wounded. The software for the Patriot missile contained a cumulative timing fault. The Patriot was designed to operate for only a few hours at a time, after which the clock was reset. As a result, the fault never had a significant effect and, therefore, was not detected. In the Gulf, however, the Dhahran Patriot missile battery ran continuously for over 100 hours. This caused the accumulated time discrepancy to become large enough to render the system inaccurate.

During the Gulf War, the United States shipped Patriot missiles to Israel for protection against the Scuds. Israeli forces detected the timing problem after only 8 hours and immediately reported it to the manufacturer in the United States. They corrected the fault as quickly as they could, but tragically, the new software arrived the day after the direct hit by the Scud [Mellor, 1994].

- Software is being delivered full of faults, late and out of budget.

- Software Engineering is an attempt to solve these problems.

- Software Engineering (SE) is a discipline whose aim is the production of fault-free software, delivered on time and within budget, that verifies the user's needs.

- The scope of software engineering is broad. It involves some aspects of:

  - mathematics,
  - computer science,
  - economics,
  - management,
  - psychology.

Historical Aspects

- Software Engineering has been coined as a term in 1967 in the belief that software design, implementation and maintenance could be put on the same footing as traditional engineering disciplines.

- Is originated from the occurrence of the software crisis, namely, that the quality of software was generally unacceptably low and that deadlines and costs limits were not being met.

- The fact that the software crisis is still with us tells us that the software production process has is own unique properties and problems, which are different from these in traditional engineering disciplines.

Mayor Differences Between Software Engineering and Traditional Engineering Disciplines

Attitude to collapsing

- A collapsed bridge has to be redesigned and rebuilt.
- An operating system (OS) crash is usual (specially in case of Windows 95) and very rarely triggers an immediate investigation in its design: Just reboot !

Attitude to maintenance

- Maintaining a bridge leads to replacement of a very limited number of its parts (road repair, painting, repairing minor cracks, etc.)
- Maintaining an OS may lead to 50% replacement of its source code on a 5-year period, specially if ported to new hardware. This may even lead to a completely new design.

Their mathematical foundations: One may say the mathematical foundations of building a system are known when one can predict its behavior.

- Bridges are built to withstand any anticipated condition. Environmental influences to a bridge are well–studied and understood; Mathematical models exist.

- It is hard to predict the environment in which the computer program will be executed. One also has to consider absence of some desired behavior.

- This led to constructing the system fault-tolerant - capable of resisting certain classes of errors regardless of when and where they occur.

- Why after 50 years, the basic mathematical laws modeling OSs ARE NOT KNOWN ?

  Because their basic building blocks, their hardware, grows COMPLEX FASTER THAN WE CAN MASTER.

The Object-oriented Paradigm

JUST IN CASE YOU WANTED TO KNOW

Suppose that you live in New Orleans, and you want to have a floral arrangement delivered to your aunt in Iowa City on her birthday [Budd, 1991]. One way would be to try to obtain a list of all the florists in Iowa City and then determine which one is located closest to your aunt's home. An easier way is to call 1-800-FLOWERS and leave the entire responsibility for delivering the floral arrangement to that organization. You do not need to know the identity of the Iowa City florist who will deliver the flowers.

In exactly the same way, when a message is sent to an object, not only is it entirely irrelevant how the request is carried out, but the unit that sends the message is not even allowed to know the internal structure of the object. The object itself is totally responsible for every detail of carrying out the message.

- Since 1967 a variety of techniques have been introduced to help solve the software crisis

- Mayor breakthroughs

  1. Between 1975-1985: The structured paradigm (SP), essentially based on stepwise refinement - a technique for decomposing large problems into smaller, more tractable problems
  2. 1985 - : The object-oriented paradigm (OOP)

Structured Analysis and Design

- Based on graphical representation and stepwise refinement

- Successful for products up to 50.000 lines of code. Modern systems however count $\geq 5 * 10^6$ lines of code

- Introduction of the SP within industry was a mayor reason for the world-scale adoption of SE practices

- Decomposition techniques in stepwise refinement:

  *** either relate to refining data or

  *** to refining actions

  but not to both simultaneously.

The Object-oriented Paradigm

Data and actions are simultaneously considered

Example

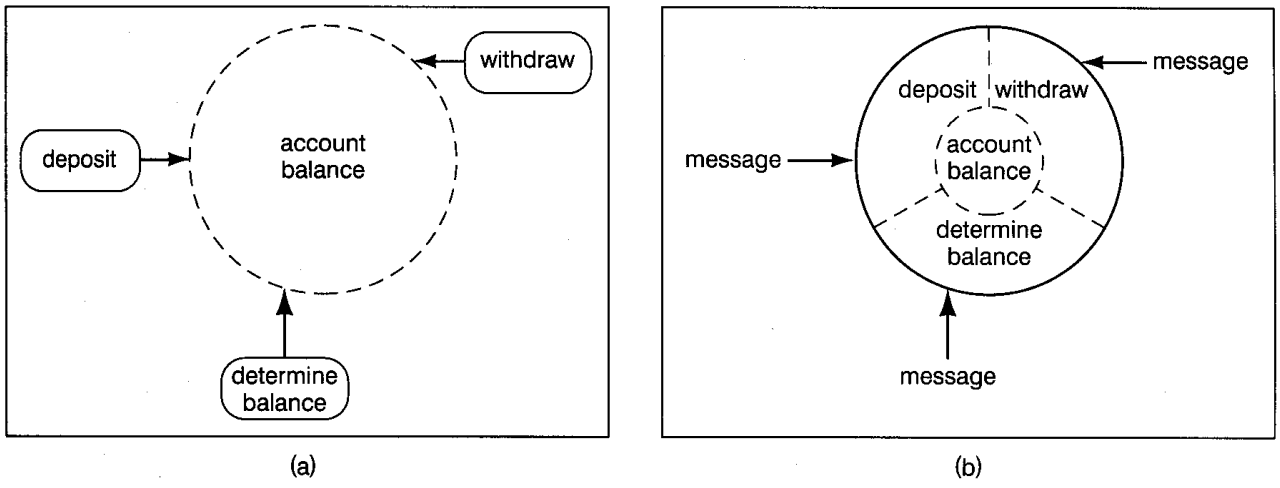

Figure 1.6 Comparison of implementations of a bank account using (a) structured paradigm and (b) object-oriented paradigm. Solid black line surrounding object denotes that details as to how account_balance is implemented are not known outside object.

- Consider the object "account balance" with 3 operations: deposit, withdraw and determine balance
- In the SP these operations are modeled by procedures which operate on the internal representation of accounts, and this internal representation is KNOWN THROUGHOUT THE WHOLE PROGRAM.
- In the OOP this representation is NOT known outside the declaration of the object. To execute these operations corresponding messages are sent to the object.

Consequences of the OOP

- Regression faults (Making a change to one part of a product causes other part to fail) occur less frequently

- Changes of representation are much easier
  - only declarations of objects must be changed
  - while in SP the WHOLE PROGRAM MUST BE SCANNED FOR CHANGES

- Everything in the product relating to the portion of the real world which is modeled by the object can be found in the object itself: encapsulation

Summary

- A product designed using SP is essentially a single unit
- A product designed using OOP consists of many smaller INDEPENDENT units.
- Consequently, OOP reduces the level of complexity of a product and simplify both its development and its maintenance.

Introduction to Software Engineering as a Technique

- Discover that programming (coding) is merely one component of software development
- Learn that when software is produced, it passes through a series of phases called the software life cycle
- Realize that software production consists of software development followed by maintenance
- Appreciate the importance of maintenance, which consumes about two-thirds of the software budget
- Appreciate that testing must be carried out in parallel with all other software activities
- Understand the need for software quality assurance (SQA) to ensure that software will be as fault-free as possible
- Realize that software is usually developed by teams
- Discover that computer-aided software engineering (CASE) tools can greatly simplify software production

Introduction to the Software Process

- The software process (SP) is the way we produce software.

- It starts with concept exploration and ends when the product is finally decommissioned.

Software Life Cycle

- The series of steps that software undergoes, from concept exploration to final retirement, is called its life cycle.

- Until the end of the 70's most organizations produced software using the waterfall model; this consists of 8 phases:

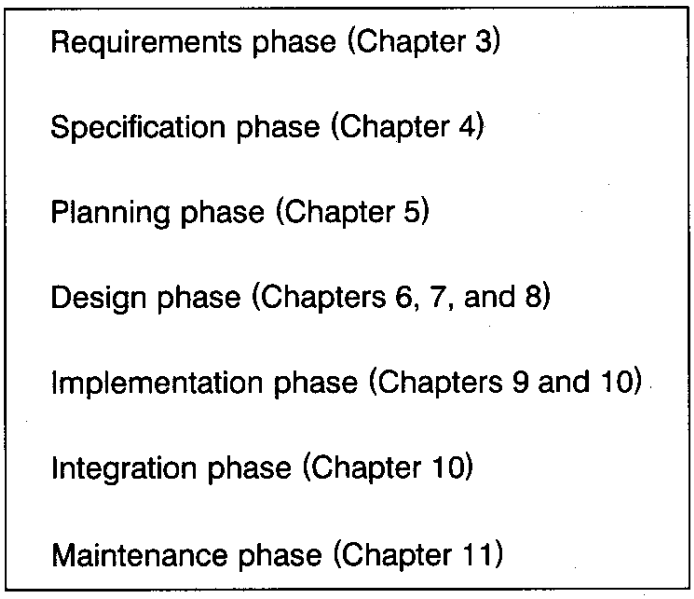

Figure 1.1 Life-cycle phases for software production and the chapter in this book in which each is presented.

- The SP also includes: (a) Tools and techniques used to develop and maintain software. (b) The software professionals involved.

- Newer software life cycle models which will be discussed include:

  1. Rapid Prototyping,
  2. The incremental model,
  3. The spiral model.

- Techniques for software production must be cost–effective and promote constructive interaction between the members of the software team.

Software Life Cycle

- Client generally does not have the necessary expertise for developing the software, so he/she calls in software developers.

- During the requirements phase, the developers work with the client to determine precisely what software the client needs (this is called system analysis.)

- Once the client's needs have been determined, the next phase is to draw up the specification document that will reflect exactly what the product is to do.

- Next, the developers plan the entire software development project answering how long it will take, how much it will cost, how many software professionals are involved, and what are the schedules and deadlines.

- Then comes the design phase. Whereas the specification phase describes what the product is to do, the design document describes how the product will do this. To that effect the product is broken down into smaller components, called modules, and then each module is designed on its own.

- After completion of the design phase, the design is implemented in the language determined in the specification document. This is called the implementation phase. Each module is coded and tested, then they are linked together to form the complete product, which is tested as a whole – this is called integration. This phase ends when the product passes its acceptance test, i.e. the client agrees that the product satisfies its specification.

- Any later change to the product is called maintenance.

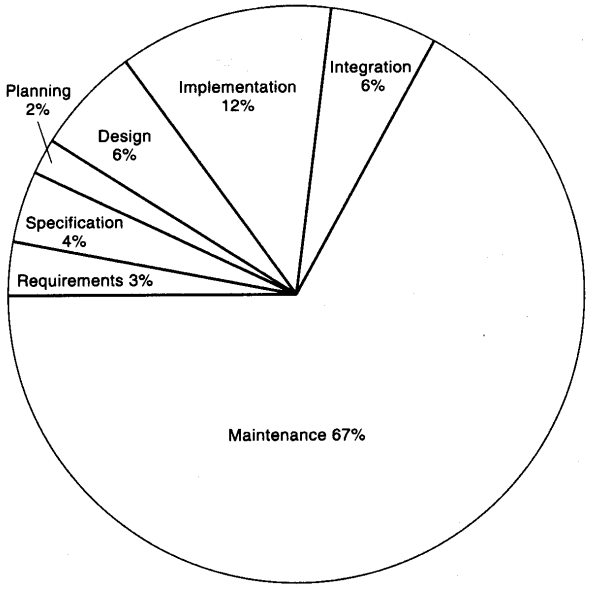

Figure 1.2 Approximate relative costs of phases of software production process.

- Maintenance takes 2/3 of the software production budget. Consequently, any technique for shorting and improving maintenance decreases that budget by a much higher factor than any improvement to the other phases.

- Corrective maintenance: removal of remaining faults ($\pm$ 20% of maintenance)

- Enhancement: adding or changing of features

- Bad products are thrown away. Good products are maintained for sometimes 20 years.

The Role of Testing

- Testing is a systematic process which aims at finding faults in the program.
- Fault: an incorrect step, process or data definition. Sometimes called a bug.
- There are 2 types of testing associated with every phase: during the phase by the software development team. and after the phase by the Software Quality Assurance Team (SQA) team.
- The SQA team performs the verification of each phase.
- Validation takes place after the whole product has been developed and before acceptance testing: It is the activity by the SQA team that determines whether the product as a whole satisfies its specification document.
- Testing is equally important during maintenance.
- Mayor problem: Making a change to one part of a product causes another part to fail — regression fault
- One has to ensure that new features function correctly but also that they do not affect the old one.

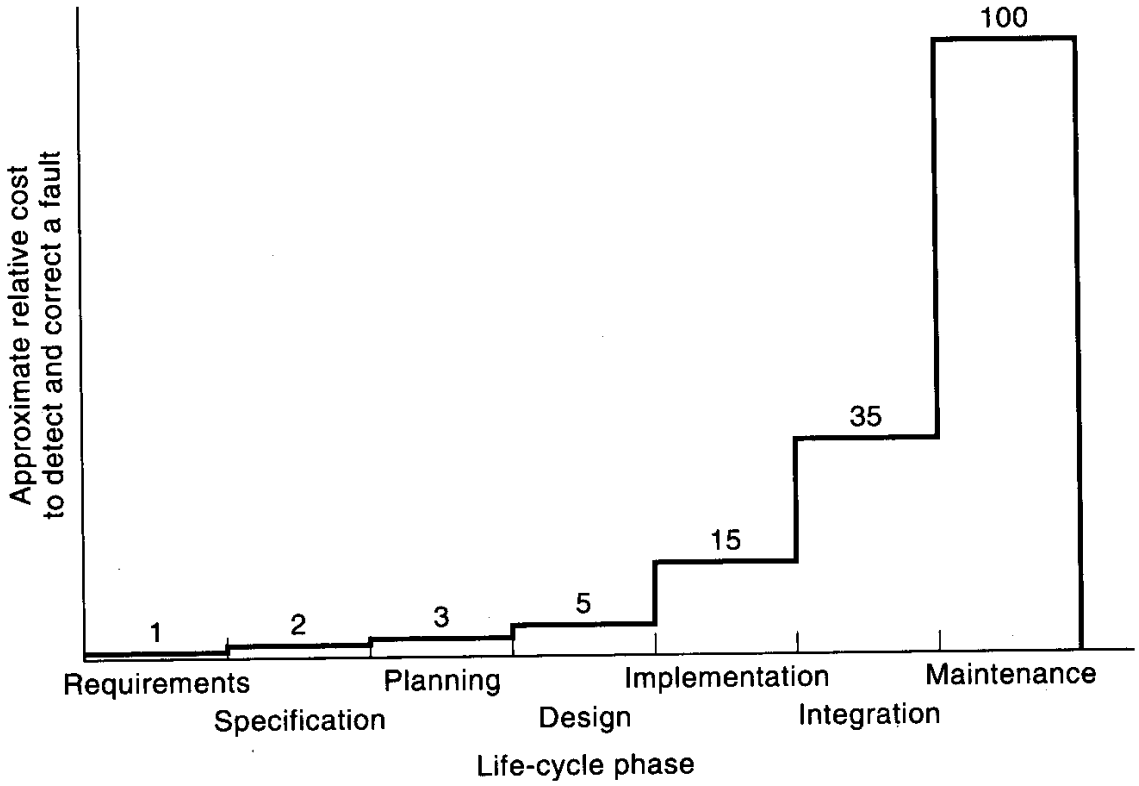

Figure 1.4 Approximate relative cost of detecting and correcting a fault during each life-cycle phase.

Reasons why later changes during life cycle cost more:

- easier to change an item in a specification document
- corrections during a later phase enforces change of documentation in previous phases
- easier to find fault during phase in which it was introduced that after that phase
- without documentation, after faults occur whole process of specification and design must be reconstructed in order to obtain a corrected implementation

65% of all faults are due to an incorrect or misinterpreted specification or requirements document $\Longrightarrow$ These should be found early !

Software Quality Assurance

Recall, the software quality assurance team tests each phase independently of the development team. Why?

- Easier to find our own faults when we show our results to others
- Potential misunderstanding of requirements and specification documents, in general, documentation of the previous phase, are better discovered if somebody else looks through the work

- There should be a managerial independence between development and quality assurance team.

  That is, development under one manager, testing under another manager and neither manager should be able to override the other. Why ?

- Separating V & V from development is called independent verification and testing.

- Software Quality Assurance (SQA) goes one step further that it also continuously ensures that the product is of high quality.

CASE Tools

- CASE: Computer Aided Software Engineering
- CASE is concerned with the use of the computer to assist the software engineer:
  - storing documentation
  - assist in reducing number of of regression faults during maintenance
  - checking consistency, e.g. that the design document includes all elements of the specification document
  - power of computer is limited: it "has" no intelligence on itself, all its intelligence is programmed into it
- CASE technology today improves productivity by an order of magnitude (i.e. a factor of 10)

Terminology

- Software: includes
  - specification documents,
  - desing documents,
  - legal and accounting documents,
  - software project management phases,
  - manuals,
  - implementation code

- Software production: consists of 2 phases:
  - software development followed by
  - its maintenance

- Product: a nontrivial piece of code, the result (product) of the software design process

- System: combined hardware and software

- Method: a way of dealing with a complete life cycle phase
- Technique: a way of dealing with part of a phase (e.g., code inspection is a testing TECHNIQUE)
- Bug: where is originates from:

JUST IN CASE YOU WANTED TO KNOW

The first use of the word *bug* to denote a fault is attributed to the late Rear Admiral Grace Murray Hopper, one of the designers of COBOL. On September 9, 1945, a moth flew into the Mark II computer that Hopper and her colleagues used at Harvard, and lodged between the contact plates of a relay. Thus there was literally a bug in the system. Hopper taped the bug to the log book and wrote, "First actual case of bug being found." The log book, with moth still attached, is in the Naval Museum at the Naval Surface Weapons Center, in Dahlgren, Virginia.

While this may have been the first use of "bug" within a computer context, the word was used in engineering slang in the nineteenth century [Shapiro, 1994]. For example, Thomas Alva Edison wrote on November 18, 1878, "This thing gives out and then that—'Bugs'—as such little faults and difficulties are called . . . " [Josephson, 1992]. One of the definitions of a bug in the 1934 Edition of *Webster's New English Dictionary* is "A defect in apparatus or its operation." It is clear from Hopper's remark that she, too, was familiar with the use of the word in that context, otherwise she would have explained what she meant.

Team Programming Aspects

DEF. performance–price factor =
time to perform 10^6 additions *
cost of CPU and main memory

- Performance–price factor has gone down with each succeeding generation by a factor of 10. This has happened 6 times.

- Consequently, organization can afford hardware that runs large products — to large to be written by one person within the allowed time constraints

- Consequently, these products are built by teams

- Now, it is not the case that if one person can perform a task in 12 months, 3 people can perform it in 4 months. This is due to the overheads imposed. (A manager is required for 4 people.)

- In general 2 kinds of problems must be overcome:
  - Team programming leads to interface problems among code components and communication problems among team members
  - Unless the team is properly organized an inordinate amount of time is wasted on conferences between team members.

- Consequently, techniques must be developed ensuring that teams are properly organized and managed.

- The correct functioning of the team is the duty of its manager.

Case Study

This book is constructed around a case study. In 1987, Samuel Chesterton made a mayor discovery, namely that while chocolate–coated ice cream and chocolate–coated almonds are both delicious, neither can be compared to chocolate–coated chocolate. He immediately opened a factory in Mossbook, New York, to manufacture chocolates that he proceeded to coat with layer upon layer of different types of chocolate. The venture succeeded beyond all expectation. Within five years, the Chocolate-Coated Chocolate (C–CC) Corporation had expanded to 27 shops selling Chesterton's chocolate–coated chocolates. Unfortunately, for the past two years C–CC has been losing money. Mr. Chesterton calls in Arabella Swinson, an internationally recognized management consultant who suggests that C–CC should implement *management–by–objectives*. That is to say, each shop is assigned a sales target for each month. The manager of the shop then encourages every employee to reach that target. Despite the simplicity of the

technique, it has worked time after time in a variety of situations, and Arabella assures Mr. Chesterton it will do the trick at C–CC.

In order to keep track of how well each shop is performing relative to its sales target that month Mr. Chesterton decides to engage Essares Software Engineering in order to computerize the management–by–objectives scheme. What happens next is described in detail in the succeeding chapters.

Term Project

Read the specification document for the ChocAn software project in Appendix A.

1. There are a number of faults in the specification document. Can you find them ? (Hint: What happens when a provider no longer wishes to supply services to ChocAn members? Also consider the implications of deleting a member record.)

2. A computer auditor would be unhappy if the product were built as specified, because there are absolutely no financial controls. Suggest two ways to improve the specification document in this regard. (Hint: Think about the total of the fees to be paid for all services and the total of all checks. Also, what about copies of the various reports?)

3. The specification document is incomplete in that a key item is missing from every report. What is it?