SWE401: Software Engineering

Chapter 1: Introduction

Objectives

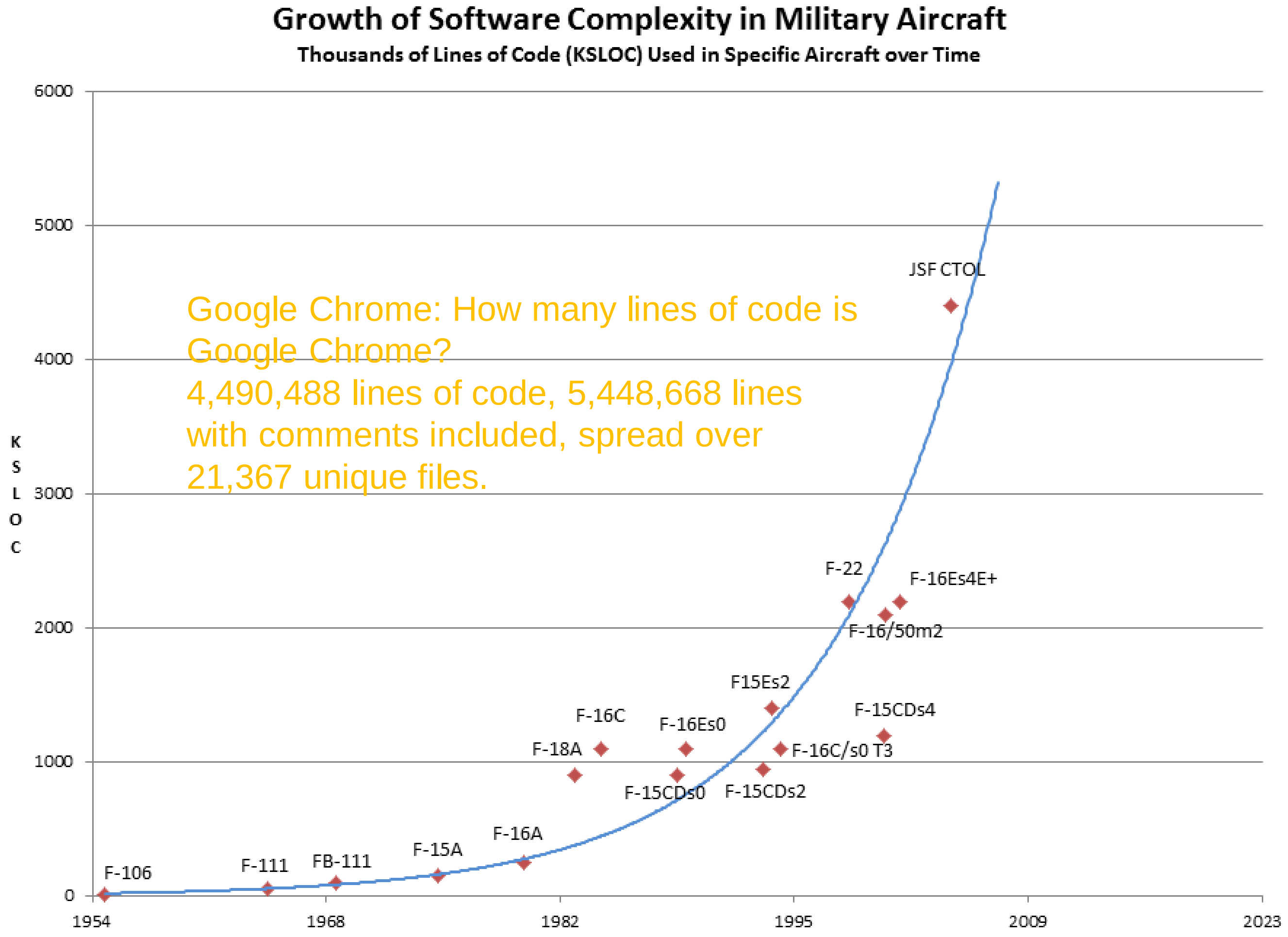
- ❑ understand what software engineering is and why it is important;
- ❑ Professional software development
  - What is meant by software engineering.
  - Understand that the development of different types of software system may require different software engineering techniques;
- ❑ Understand ethical and professional issues that are important for software engineers;
- ❑ Software engineering ethics
  - A brief introduction to ethical issues that affect software engineering.

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.

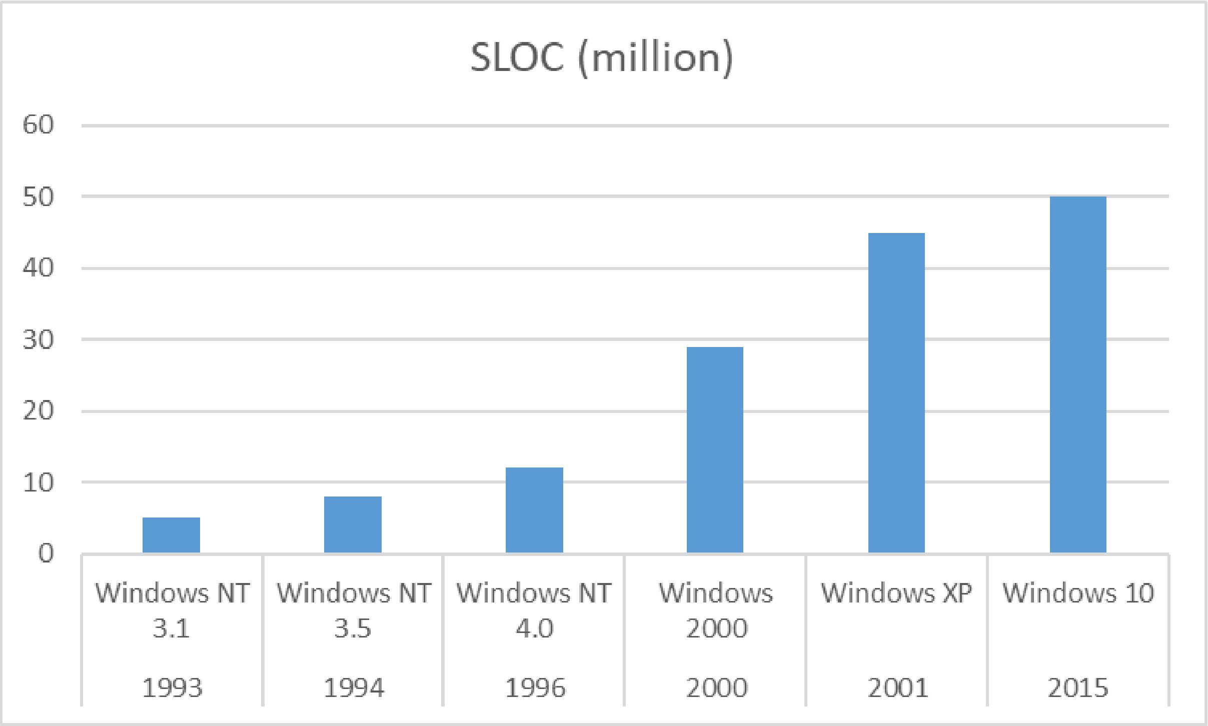
Software costs

- Software costs often dominate computer 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.



Google Chrome: How many lines of code is Google Chrome?
4,490,488 lines of code, 5,448,668 lines with comments included, spread over 21,367 unique files.

Size of Windows: Millions of LOC



Programming in the Small

- There are many different types of software system, ranging from simple embedded systems to complex, worldwide information systems.
- There are no universal notations, methods, or techniques for software engineering because different types of software require different approaches.
- Developing an organizational information system is completely different from developing a controller for a scientific instrument.
- For small programs:
  - Requirements simple, precise, do not change
  - Only used by one person
  - Only used a few times
- Developing an organizational complex information system is completely different from developing a simple program.
- All of these applications need software engineering; they do not all need the same software engineering methods and techniques.
- Let's analyze the difference between engineering methods with example of civil engineering for buildings.

Building a Small Building

Building a Skyscrapper

Some Key Differences

Shed

- No real planning
- No foundation dig
- Start Construction straight away
- No internal structure
- Simple, fixed instructions to follow
- No design decision to make
- Make adjustments on the fly
- Done by one person with basic skills
- Few tools needed
- Few number of users with simple requirements

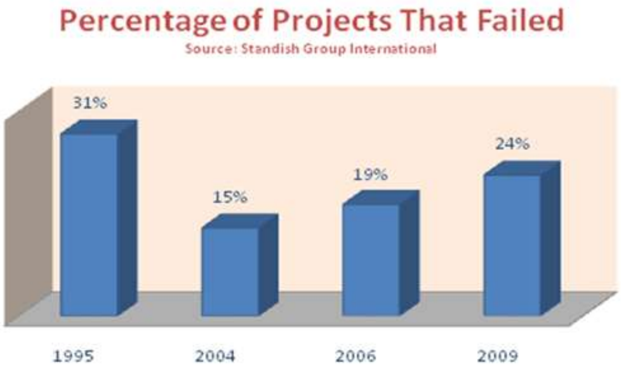
Skyscraper

- Major planning and project management
- Major work on foundations
- No construction until excavation laid
- Major internal structure
- Complex, ill-defined requirements that may change
- Major design tasks
- All adjustments must be planned
- Multiple people teams with multiple skills
- Many tools needed
- Multiple users with complex requirements

A Software Failure?

There are still many reports of software projects going wrong and of “software failures.” Software engineering is criticized as inadequate for modern software development.

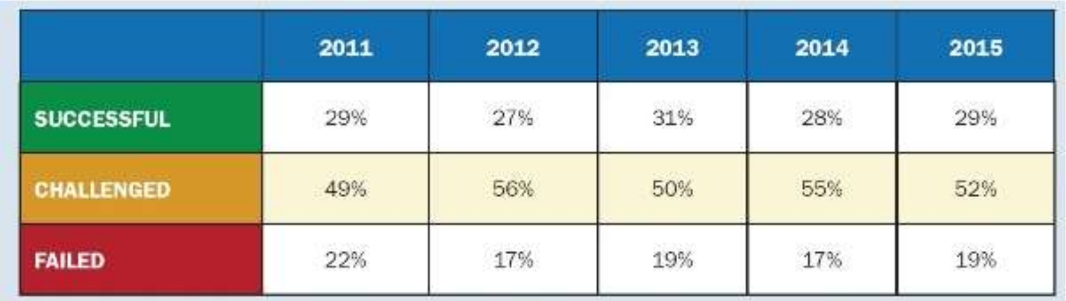
A Software Failure?



Percentage of Projects That Failed

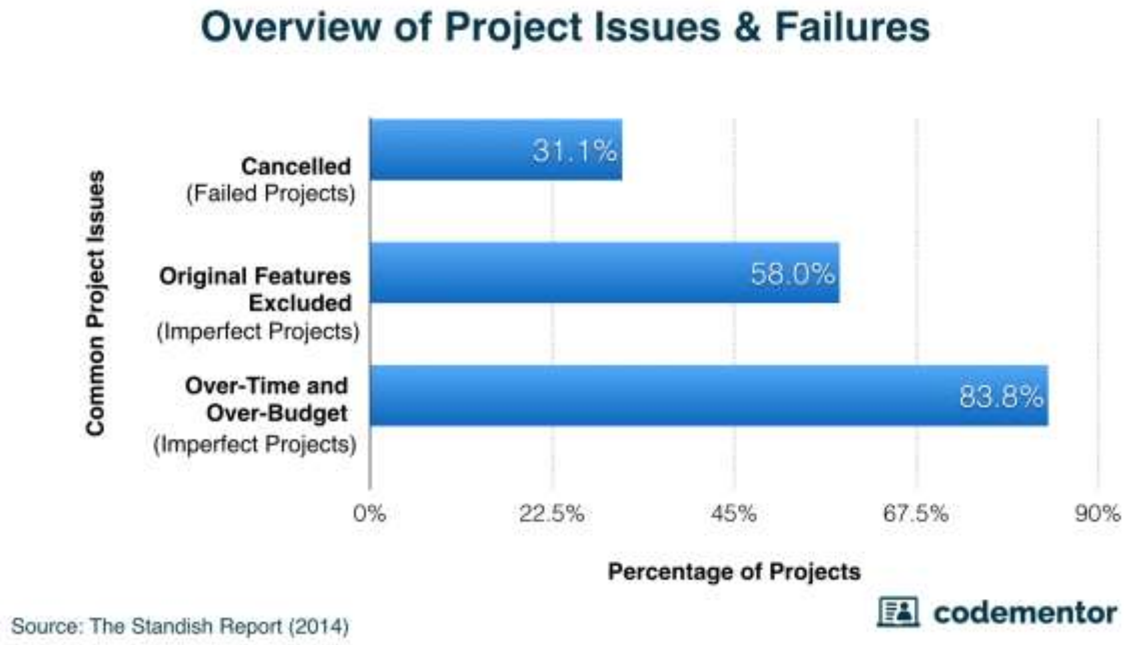
Source: Standish Group International

Figure 1 - The resurgence of failed software projects



	2011	2012	2013	2014	2015
SUCCESSFUL	29%	27%	31%	28%	29%
CHALLENGED	49%	56%	50%	55%	52%
FAILED	22%	17%	19%	17%	19%

A Software Failure?



Source: The Standish Report (2014)

Software project failure

- *Many of these so-called software failures are a consequence of two factors:*
- *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 expensive and less reliable than it should be.

Why Large Projects Fail

- Lack of end-user involvement
- Poor communication (internal and external)
- Poor configuration management
- Inadequate testing
- Poor management
- Technical complexity issues

Consequences

- Poor Quality
- Late delivery
- Excessive costs

What is Engineering?

- The use of scientific (including mathematical) principles to construct artifacts
- "Application of knowledge of the mathematical and natural sciences, gained by study, experience and practice, to the efficient use of the materials and forces of nature". (MS Encarta)
- "Creation and design of practical solutions to real physical problems"

What is Software Engineering?

- "The systematic approach to the development, operation, maintenance and retirement of software" IEEE Standard Glossary of Software Engineering Terminology, 1983
- The construction of quality software with a limited budget and a given deadline in the context of constant change

Professional software development

Frequently asked questions about software engineering

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.

Frequently asked questions about software engineering

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.

Software products

- Generic products
  - Stand-alone systems that are marketed and sold to any customer who wishes to buy them.
  - Examples – PC software such as graphics programs, project management tools; CAD software; software for specific markets such as appointments systems for dentists.
- Customized products
  - Software that is commissioned by a specific customer to meet their own needs.
  - Examples – embedded control systems, air traffic control software, traffic monitoring systems.

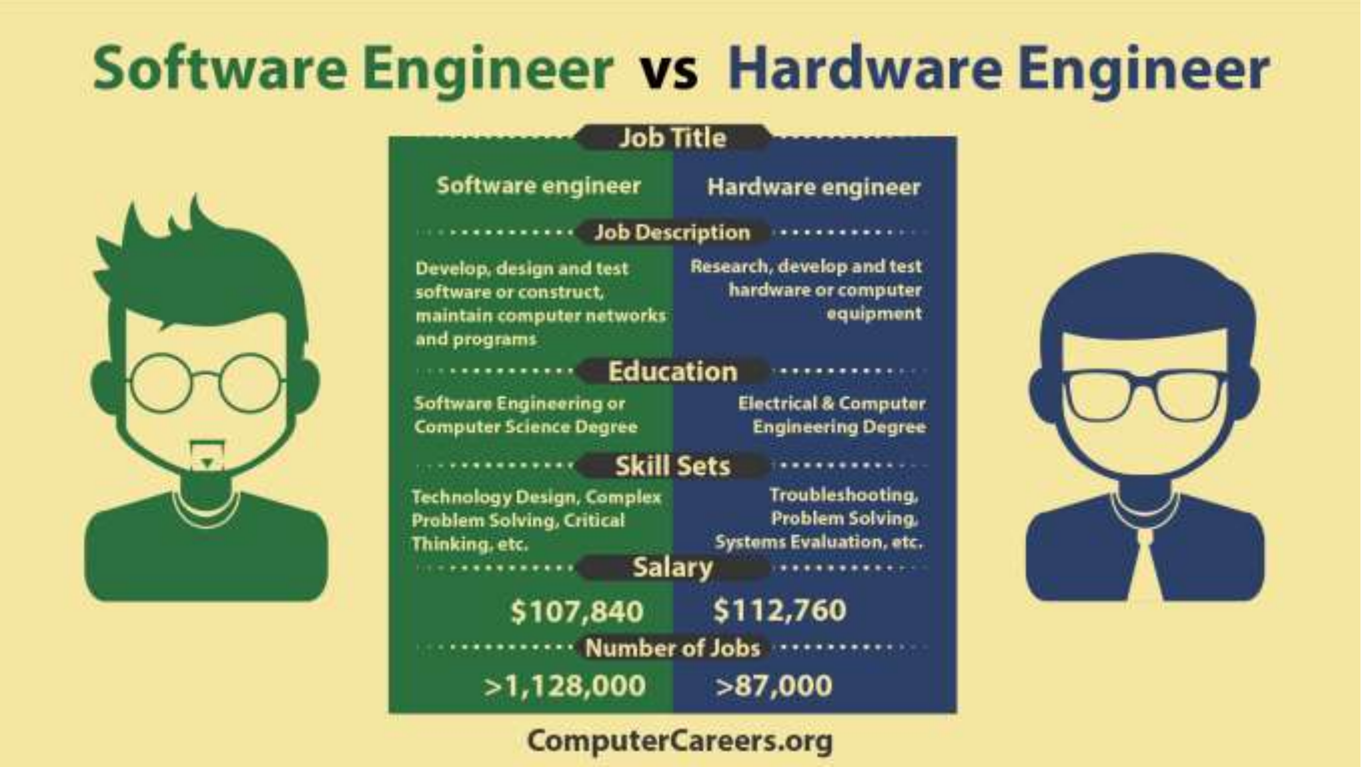
Product specification

- ❑ Generic products
  - The specification of what the software should do is owned by the software developer and decisions on software change are made by the developer.
- ❑ Customized products
  - The specification of what the software should do is owned by the customer for the software and they make decisions on software changes that are required.

Essential attributes of good software

Product characteristic	Description
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
Dependability and security	Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.
Acceptability	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.

What is Software Engineering?



Software Engineer vs Hardware Engineer

Software Engineer	Hardware Engineer
Job Title	
Software engineer	Hardware engineer
Job Description	
Develop, design and test software or construct, maintain computer networks and programs	Research, develop and test hardware or computer equipment
Education	
Software Engineering or Computer Science Degree	Electrical & Computer Engineering Degree
Skill Sets	
Technology Design, Complex Problem Solving, Critical Thinking, etc.	Troubleshooting, Problem Solving, Systems Evaluation, etc.
Salary	
$107,840	$112,760
Number of Jobs	
>1,128,000	>87,000

ComputerCareers.org

Software engineering

- Software engineering is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use.
- Engineering discipline
  - Using appropriate theories and methods to solve problems bearing in mind organizational and financial constraints.
- All aspects of software production
  - Not just technical process of development. Also project management and the development of tools, methods etc. to support software production.

Importance of software engineering

- More and more, individuals and society rely on advanced software systems. We need to be able to produce reliable and trustworthy systems economically and quickly.
- It is usually cheaper, in the long run, to use software engineering methods and techniques for software systems rather than just write the programs as if it was a personal programming project. For most types of system, the majority of costs are the costs of changing the software after it has gone into use.

Software process activities

- Software specification, where customers and engineers define the software that is to be produced and the constraints on its operation.
- Software development, where the software is designed and programmed.
- Software validation, where the software is checked to ensure that it is what the customer requires.
- Software evolution, where the software is modified to reflect changing customer and market requirements.

General issues that affect software

- ❑ Security and trust
  - As software is intertwined with all aspects of our lives, it is essential that we can trust that software.
- ❑ Scale
  - Software has to be developed across a very wide range of scales, from very small embedded systems in portable or wearable devices through to Internet-scale, cloud-based systems that serve a global community.

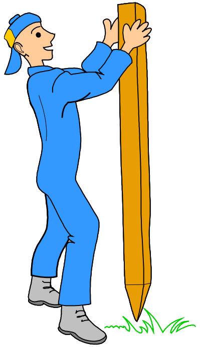
Software engineering diversity

- ❑ There are many different types of software system and there is no universal set of software techniques that is applicable to all of these.
- ❑ The software engineering methods and tools used depend on the type of application being developed, the requirements of the customer and the background of the development team.

Introduction: Software is Complex

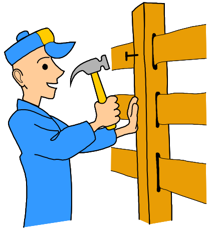
- ❑ Complex ≠ complicated
- ❑ Complex = composed of many simple parts related to one another
- ❑ Complicated = not well understood, or explained

Complexity Example:
Scheduling Fence Construction Tasks

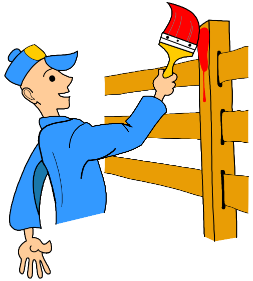


Setting posts
[3 time units]

Cutting wood
[2 time units]



Nailing
[2 time units for unpainted;
3 time units otherwise]



Painting
[5 time units for uncut wood;
4 time units otherwise]

Setting posts < Nailing, Painting

Cutting < Nailing

…shortest possible completion time = ?

[⇒ “simple” problem, but hard to solve without a pen and paper]

More Complexity

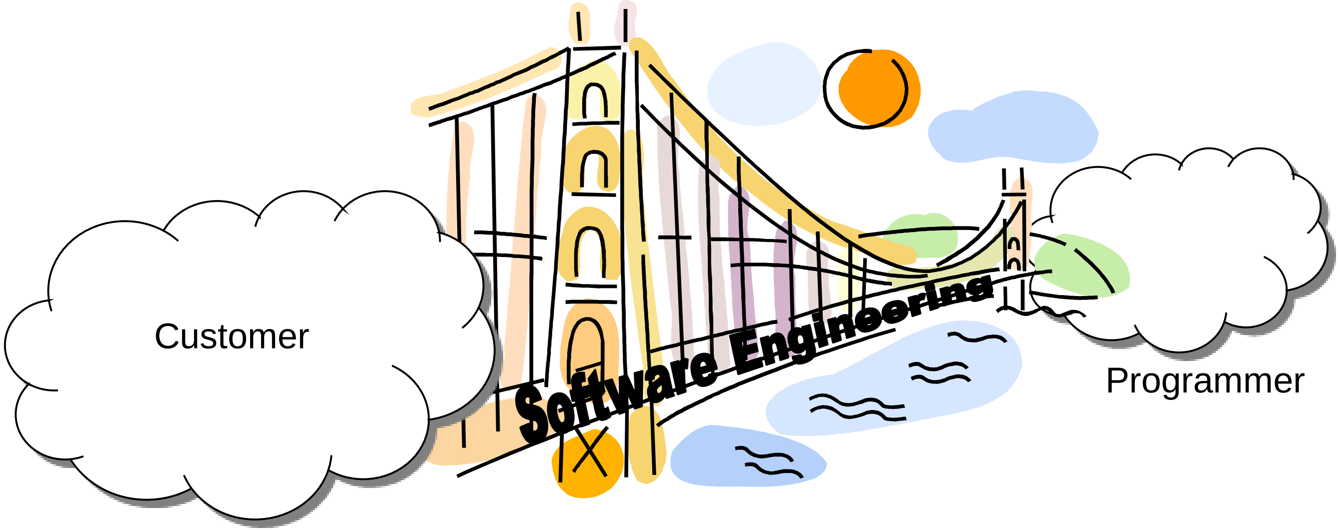
Suppose today is Sunday, November 29

What day will be on January 3?

[To answer, we need to bring the day names and the day numbers into coordination, and for that we may need again a pen and paper]

The Role of Software Engg. (1)

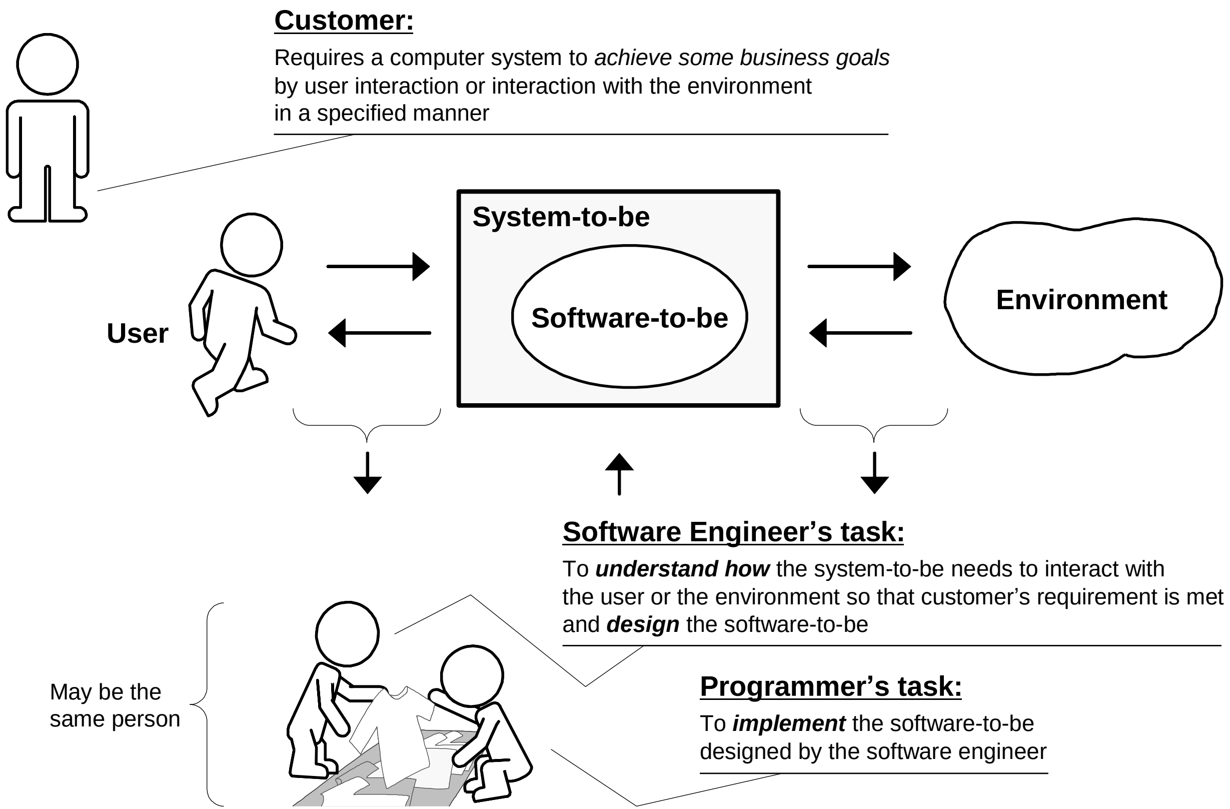
A bridge from customer needs to programming implementation



<u>First law of software engineering</u>

Software engineer is willing to learn the problem domain
(problem cannot be solved without understanding it first)

The Role of Software Engg. (2)



Customer:

Requires a computer system to *achieve some business goals*
by user interaction or interaction with the environment
in a specified manner

User

System-to-be

Software-to-be

Environment

Software Engineer's task:

To ***understand how*** the system-to-be needs to interact with
the user or the environment so that customer's requirement is met
and ***design*** the software-to-be

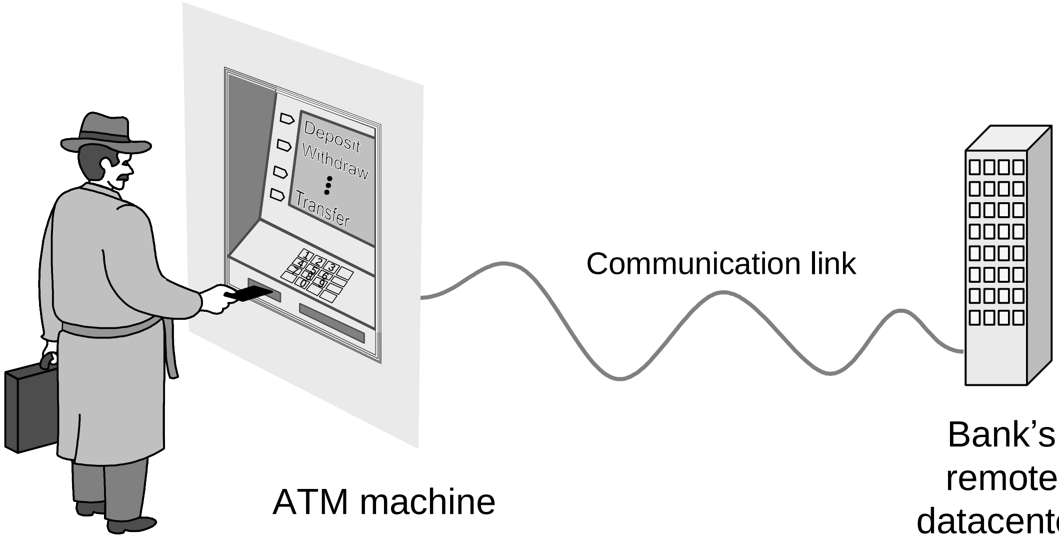
Programmer's task:

To ***implement*** the software-to-be
designed by the software engineer

May be the
same person

Example: ATM Machine

Understanding the money-machine problem:



Communication link

ATM machine

Bank customer

Bank’s remote datacenter

Importance of software engineering

- More and more, individuals and society rely on advanced software systems. We need to be able to produce reliable and trustworthy systems economically and quickly.
- It is usually cheaper, in the long run, to use software engineering methods and techniques for software systems rather than just write the programs as if it was a personal programming project. For most types of system, the majority of costs are the costs of changing the software after it has gone into use.

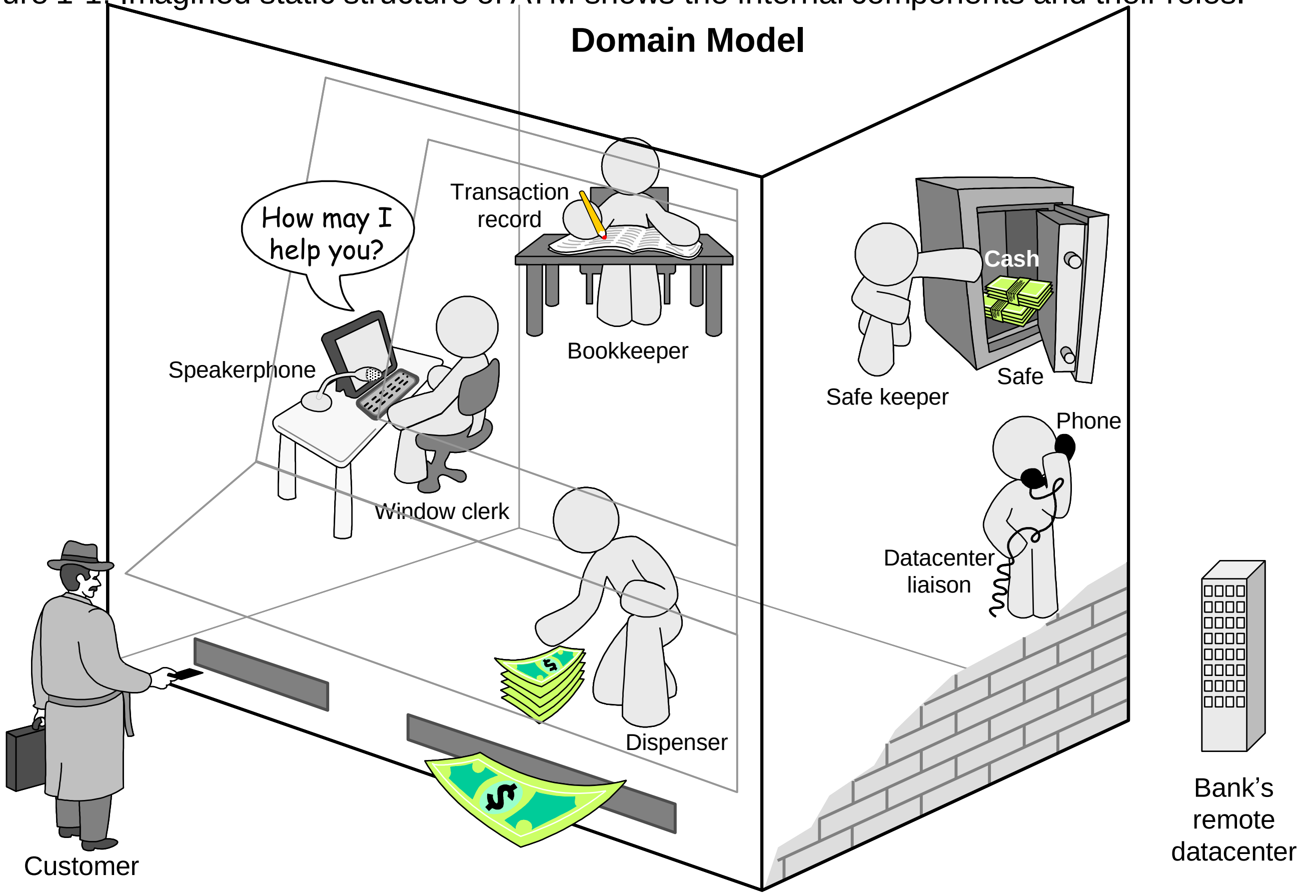
Problem-solving Strategy

Divide-and-conquer:

- Identify logical parts of the system that each solves a part of the problem
- Easiest done with the help of a domain expert who already knows the steps in the process (“how it is currently done”)
- Result:
  A Model of the Problem Domain
  (or “domain model”)

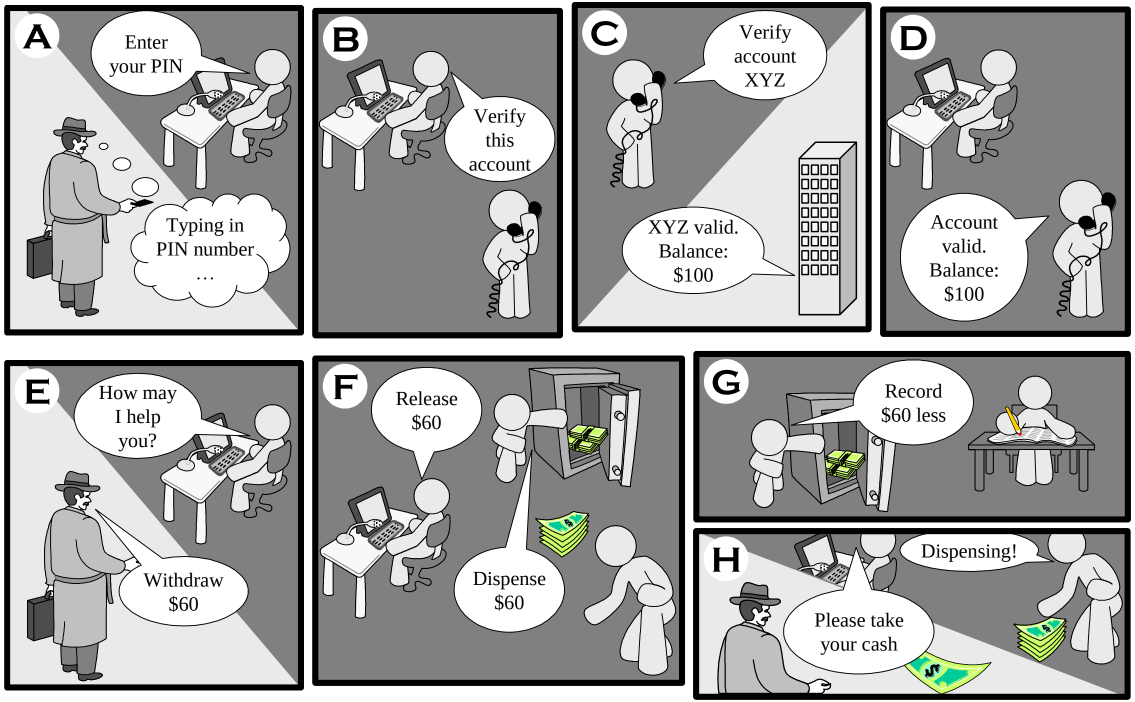
How ATM Machine Might Work

Figure 1-1: Imagined static structure of ATM shows the internal components and their roles.



Cartoon Strip: How ATM Machine Works

Figure 1-2: Dynamic interactions of the imagined components during task accomplishment.



Software Engineering Blueprints

- Specifying software problems and solutions is like cartoon strip writing

- Unfortunately, most of us are not artists, so we will use something less exciting:
  UML symbols

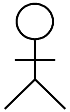
- However, in real-life the level of knowledge and experience to develop a software is high.

Second Law of Software Engineering

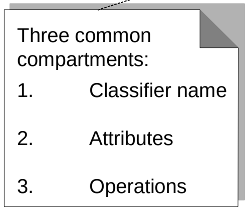
- Software should be written for people first
  - Software developer must keep in mind that software is written for people, not for computers.
  - Computers just run software—a minor point. *It is people who understand, maintain, improve, and use software to solve real-world problems.*
  - The particular technologies evolve or become obsolete, but the underlying principles and concepts will likely resurface in new technologies.

UML – Language of Symbols

UML = Unified Modeling Language



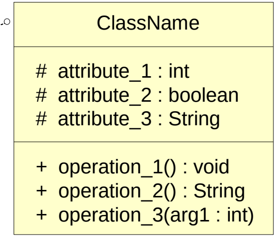
Actor



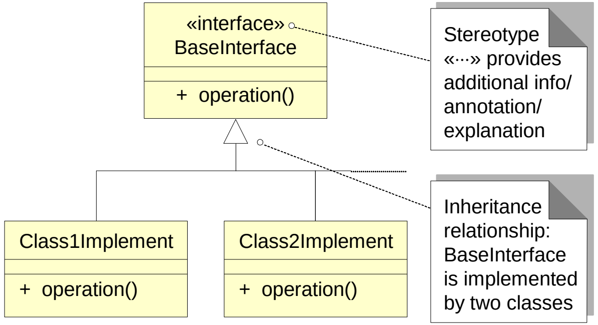
Three common compartments:

1. Classifier name
2. Attributes
3. Operations

Comment



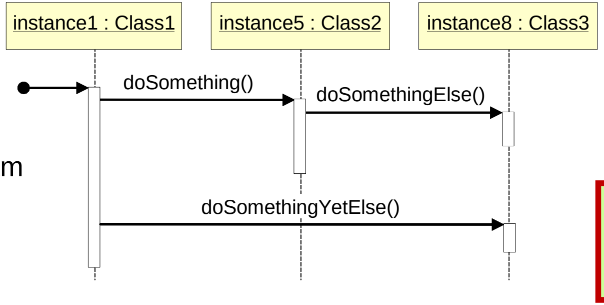
Software Class



Stereotype «···» provides additional info/ annotation/ explanation

Inheritance relationship: BaseInterface is implemented by two classes

Software Interface Implementation



Interaction Diagram

Online information:
http://www.uml.org

Software engineering ethics

Software engineering ethics

- 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 behaviour is more than simply upholding the law but involves following a set of principles that are morally correct.

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 outwith 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).

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.

Rationale for the code of ethics

- *Computers have a central and growing role in commerce, industry, government, medicine, education, entertainment and society at large. Software engineers are those who contribute by direct participation or by teaching, to the analysis, specification, design, development, certification, maintenance and testing of software systems.*
- *Because of their roles in developing software systems, software engineers have significant opportunities to do good or cause harm, to enable others to do good or cause harm, or to influence others to do good or cause harm. To ensure, as much as possible, that their efforts will be used for good, software engineers must commit themselves to making software engineering a beneficial and respected 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.