Course: Software Engineering

INTRODUCTION

Alexander Vazhenin

COURSE STRUCTURE AND ORGANIZATION

- The course lectures are covering the following topics:

  Part 1. The Software Process

  Part 2. Software Engineering Practice

  Part 3. Software Design Strategies

THE EXPERIMENTAL PART

- *Presentation Topic:* Students have to present advanced material based on reading of current software engineering journals and professional magazines (This course).

GRADING

Student evaluation method

1. Two tests (35 points each)
2. Presentation Topic (30 points)

USEFUL LINKS:

http://sealpv2.u-aizu.ac.jp/

Register on this site:
User ID → your student's ID
Password → your password
Enter to the Course
It contains handouts and submission sites.

Course: Software Engineering

Lecture 1
GENERIC VIEW

Alexander Vazhenin

TOPICS COVERED

- Professional software development
  - What is meant by software engineering.
- Historical overview
- Frequently Asked Questions
- Software engineering ethics
  - A brief introduction to ethical issues that affect software engineering.
- Case studies

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.

WHAT IS A SOFTWARE ENGINEERING?

- Part of the nature of software engineering is implied by its classification as an engineering discipline.
- There are some of its unique characteristics and problems. Uniqueness here does not imply that the characteristics and problems cannot be found in other engineering disciplines.
- It only means that they are not common to all engineering disciplines, or significantly more important in software engineering.

SOME UNIQUE FEATURES (1)

- Software engineering is characterized by its primary product, which is software - programs that direct a computer to perform some task.
- In software engineering, there is a well developed science, computer science, that covers, among the following things,
  - Concepts of programming languages,
  - Algorithms and data structures,
  - Important aspects of hardware systems and systems software.

SOME UNIQUE FEATURES (2)

- Many of the subject areas of computer science deal with software products.
- Because of this, the boundary between computer science and software engineering is difficult to define.
- In particular, the values of software engineering are an important element of computer science.

SOME UNIQUE FEATURES (3)

- In some engineering disciplines where production in large quantities is an objective, the quantities introduce a great deal of complexity into the production process.
- Reproducing software is relatively easy, once it has been designed. The engineering problems that do arise are also of a very different nature than the software design problems.
- They are problems that are handled by other engineering disciplines. Software engineering is concerned almost exclusively with the design of the product and not the production process.

SOFTWARE ENGINEERING PROCESS

- Although there is a clear-cut category that describes the product of software engineering, the functionality of that product is extremely broad.
- One indication of this is the fact that any engineering process could automated so that it becomes, in part, a software engineering project.
- More generally, software can be applied towards the automation of any human task. This has two important consequences.

SOFTWARE ENGINEERING THEORY

- There is no single domain of knowledge that covers the behavior of all software products.
- Software engineering does not have a single body of theory that tells us how to specify and measure the behavior of its products.
- Computer science deals with the components of software and some of the tools and techniques involved in its construction, but does not and cannot deal with all of the kinds of behavior exhibited by software.

SOFTWARE DESIGNERS AND CUSTOMERS

- The values of software engineering are driven by customers who have an unlimited range of needs. A classification of the wide range of values is a virtually impossible task.
- Even where there are common values, they vary considerably in importance.
- A value that is unimportant in one software product may be crucial in another. For these reasons, the ethical and value-oriented side of software engineering takes on a more prominent role.

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.

SOFTWARE FUNCTIONALITY (1)

- The extensive variety of software functionality and values can be dealt with in three ways.
  - First, software engineers can specialize. To some extent this has already happened. There are some software development firms, for example, that specialize in business and accounting software.
  - A more common mode of specialization is in-house software development. A firm that uses software extensively will often have a department dedicated to software development.
  - The software engineers that work in that department will automatically develop a specialization tailored to the needs of the firm.

SOFTWARE FUNCTIONALITY (2)

- The extensive variety of software functionality and values can be dealt with in three ways.
- A second option is that people that are trained in the application area develop programming skills.
  - ➢This is a workable option for relatively simple software.
  - ➢For more complex software, the loss in understanding of software construction may outweigh the advantage of familiarity with the application domain.

SOFTWARE FUNCTIONALITY (3)

- A more general solution to the problem is to modify the software development process. The basic idea is to allocate a significant amount of time early in the development on domain analysis.
  - *Domain analysis* involves communicating with experts in the application area, noting important concepts and their relationships.
  - During domain analysis, software developers build a domain model that can be readily converted into software.
  - Another important process change involves building the product in stages and getting feedback from the customer about the preliminary versions.

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.

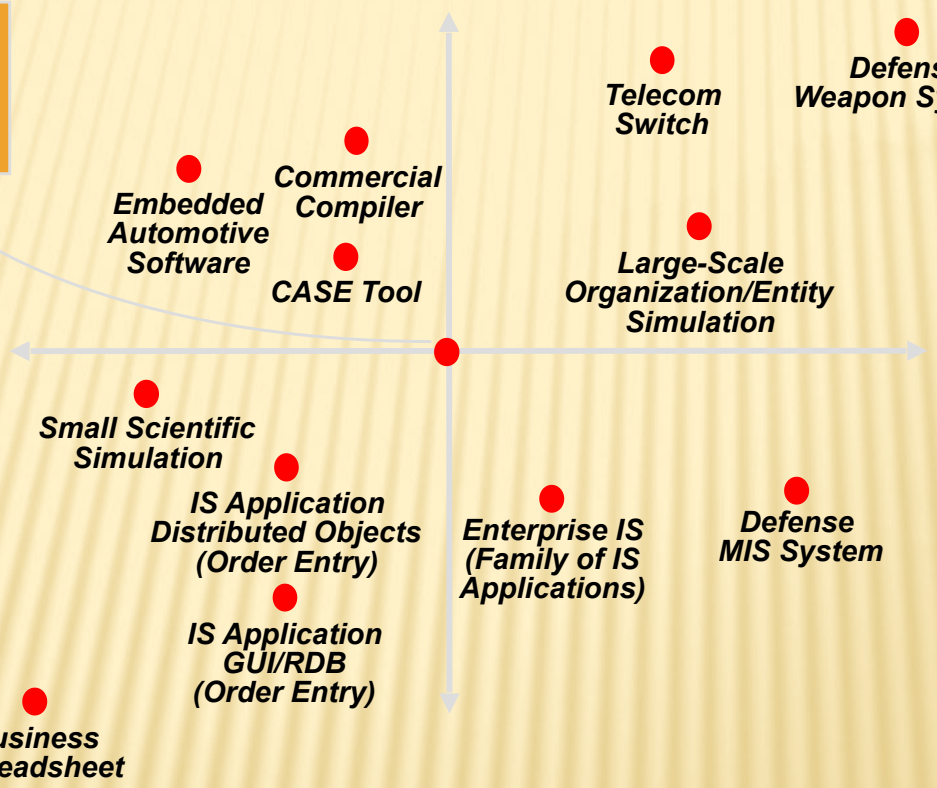
PRODUCT SPECIFICATION

- 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.

DIMENSIONS OF SOFTWARE COMPLEXITY

Higher technical complexity
- Embedded, real-time, distributed, fault-tolerant
- Custom, unprecedented, architecture reengineering
- High performance

An average software project
- 5-10 people
- 3-9 month duration
- 3-5 external interfaces
- Some unknowns & risks

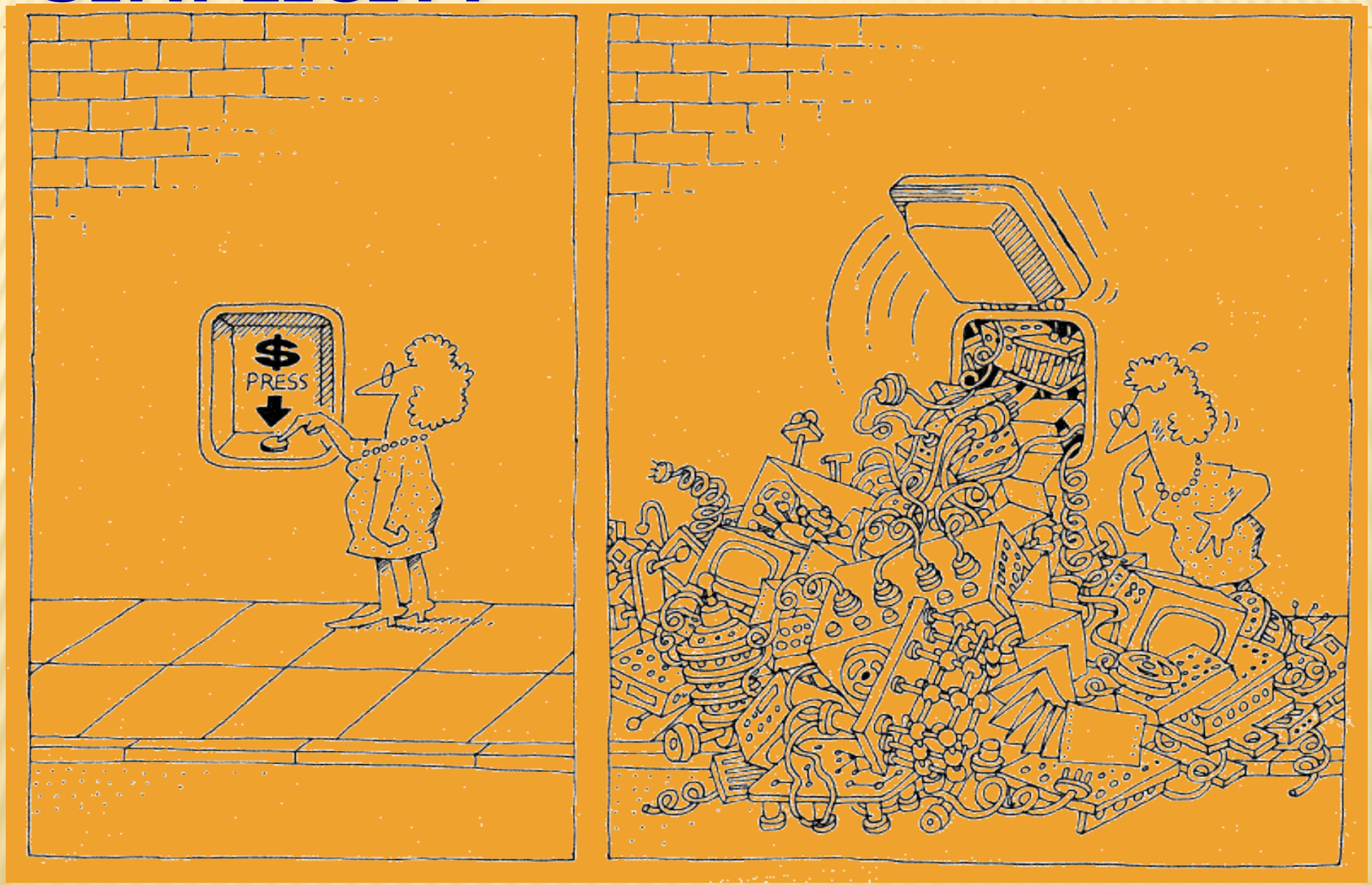


Lower management complexity
- Small scale
- Informal
- Single stakeholder
- "Products"

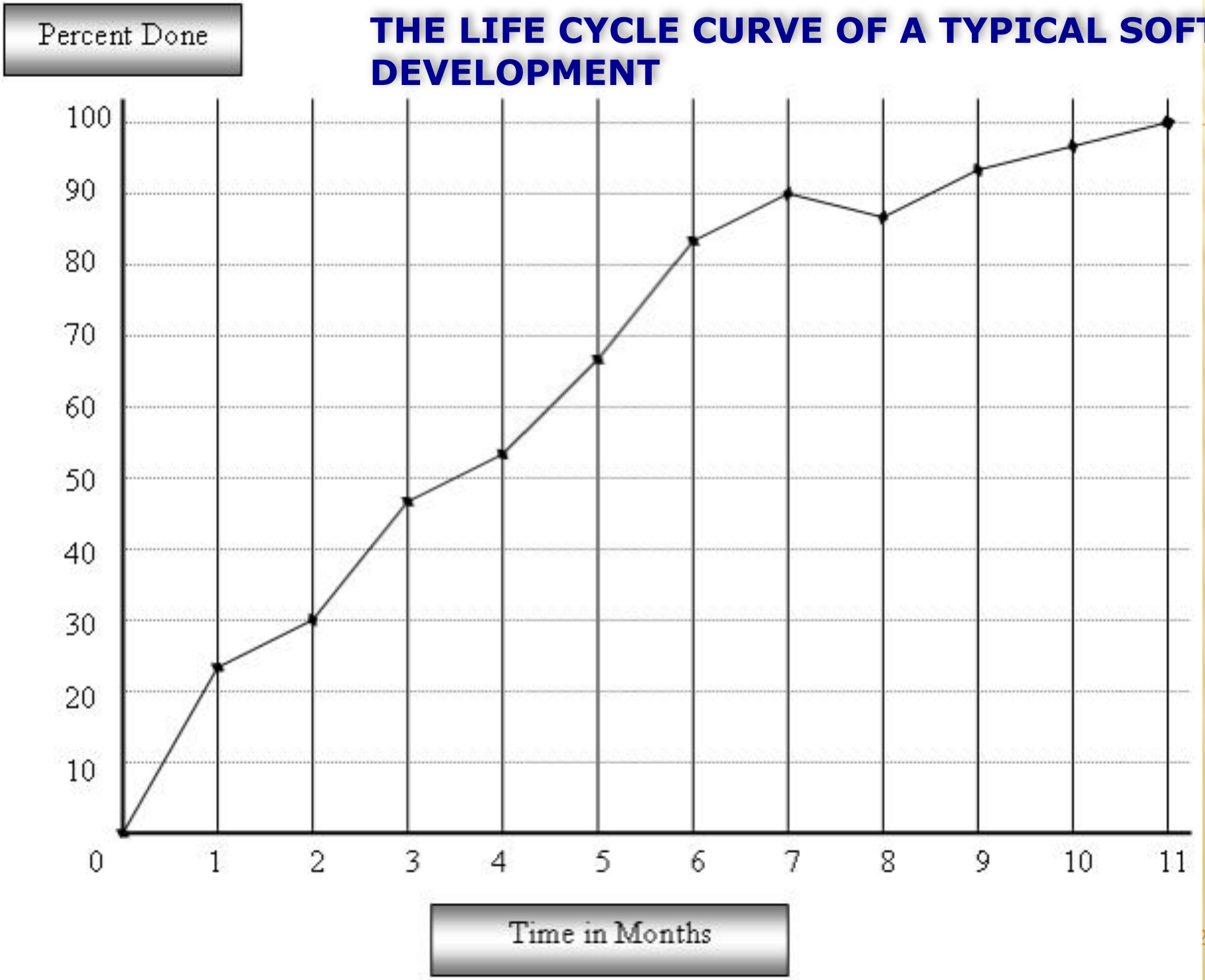
Higher management complexity
- Large scale
- Contractual
- Many stake holders
- "Projects"

Lower technical complexity
- Mostly 4GL, or component-based
- Application reengineering
- Interactive performance

CREATING THE ILLUSION OF SIMPLICITY



THE LIFE CYCLE CURVE OF A TYPICAL SOFTWA DEVELOPMENT



HISTORICAL OVERVIEW

SOFTWARE ENGINEERING STAGES

1. The Pioneering Era
2. 1945 to 1965: The origins
3. 1965 to 1985: The software crisis
4. 1985 to 1989: No silver bullet
5. 1990 to 1999: Prominence of the Internet
6. 2000 to Present: Lightweight Methodologies

WHEN A SOFTWARE ENGINEERING APPEARED?

- Software engineering tried to find a “best solution” for much of its early existence.
- The North Atlantic Treaty Organization (NATO) conference at Garmish, Germany, in 1968, at which the term *software engineering* became popularized was most remarkable in that the attendees came together to share solutions, and wound up sharing problems.

SOME HISTORICAL EXAMPLES (1)

Bruce H. Barnes (Deputy Division Director)
National Science Foundation mentioned:

- *When I joint the National Science Foundation in 1974, Software Engineering as an academic discipline hardly existed.*
- *Structured programming and top-down design had permeated the curriculum, but that was the extent of software engineering in the curriculum.*
- *"Curriculum 1978" did not use the term Software Engineering, but emphasized good software development practices.*

NO SILVER BULLET

- By 1986 most Computer Science programs had introduced senior project courses involving a significant portion of quality software engineering practices.
- Fred Brooks has published his "No Silver Bullet" essay in IEEE Computer magazine in 1987.
- In this article, he pointed out that there appeared to be more than one solution to any software-engineering problem. His reputation was so great that he put the skids on the Computer Aided Software Engineering (CASE) solutions then in vogue.

SOME HISTORICAL EXAMPLES (2)

- The 1989 ACM report on "Computing as a Discipline" included Software Methodology and Engineering as one of its elements.
- The "Curriculum 1991" report of the ACM and the IEEE-Computer Society recommends a significant amount of software engineering for every computer science graduate and includes a sample curriculum with a Software Engineering emphasis.
- Currently there are Software Engineering programs, especially in Europe.

THE SOFTWARE CRISIS

- Software engineering was spurred by the so-called *software crisis* of the 1960s, 1970s, and 1980s, which identified many of the problems of software development.
- Many software projects ran over budget and schedule. Some projects caused property damage.
- A few projects caused loss of life. The software crisis was originally defined in terms of productivity, but evolved to emphasize quality.
- Some used the term *software crisis* to refer to their inability to hire enough qualified programmers.

THE SOFTWARE CRISIS

- Cost and Budget Overruns: This decade-long project from the 1960s eventually produced one of the most complex software systems at the time. OS/360 was one of the first large (1000 programmers) software projects. Fred Brooks claims in *The Mythical Man Month* that he made a multi-million dollar mistake of not developing a coherent architecture before starting development.
- Property Damage: Software defects can cause property damage. Poor software security allows hackers to steal identities, costing time, money, and reputations.
- Life and Death: Software defects can kill. Some embedded systems used in radiotherapy machines failed so catastrophically that they administered lethal doses of radiation to patients. The most famous of these failures is the *Therac 25* incident.

FREQUENTLY ASKED QUESTIONS

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.

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.

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 MOST SOFTWARE

- Heterogeneity
  - Increasingly, systems are required to operate as distributed systems across networks that include different types of computer and mobile devices.
- Business and social change
  - Business and society are changing incredibly quickly as emerging economies develop and new technologies become available. They need to be able to change their existing software and to rapidly develop new software.
- Security and trust
  - As software is intertwined with all aspects of our lives, it is essential that we can trust that software.

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.

APPLICATION TYPES

- Stand-alone applications
  - These are application systems that run on a local computer, such as a PC. They include all necessary functionality and do not need to be connected to a network.
- Interactive transaction-based applications
  - Applications that execute on a remote computer and are accessed by users from their own PCs or terminals. These include web applications such as e-commerce applications.
- Embedded control systems
  - These are software control systems that control and manage hardware devices. Numerically, there are probably more embedded systems than any other type of system.

APPLICATION TYPES

- Batch processing systems
  - These are business systems that are designed to process data in large batches. They process large numbers of individual inputs to create corresponding outputs.
- Entertainment systems
  - These are systems that are primarily for personal use and which are intended to entertain the user.
- Systems for modeling and simulation
  - These are systems that are developed by scientists and engineers to model physical processes or situations, which include many, separate, interacting objects.

APPLICATION TYPES

- Data collection systems
  - These are systems that collect data from their environment using a set of sensors and send that data to other systems for processing.
- Systems of systems
  - These are systems that are composed of a number of other software systems.

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. Of course, 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.

SOFTWARE ENGINEERING AND THE WEB

- The Web is now a platform for running application and organizations are increasingly developing web-based systems rather than local systems.
- Web services (discussed in Chapter 19) 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 buy pay according to use.

WEB SOFTWARE ENGINEERING

- 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.
- 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.
- User interfaces are constrained by the capabilities of web browsers.
  - Technologies such as AJAX allow rich interfaces to be created within a web browser but are still difficult to use. Web forms with local scripting are more commonly used.

WEB-BASED SOFTWARE ENGINEERING

- Web-based systems are complex distributed systems but the fundamental principles of software engineering discussed previously are as applicable to them as they are to any other types of system.
- The fundamental ideas of software engineering, discussed in the previous section, apply to web-based software in the same way that they apply to other types of software system.

KEY POINTS

- Software engineering is an engineering discipline that is concerned with all aspects of software production.
- Essential software product attributes are maintainability, dependability and security, efficiency and acceptability.
- The high-level activities of specification, development, validation and evolution are part of all software processes.
- The fundamental notions of software engineering are universally applicable to all types of system development.

KEY POINTS

- There are many different types of system and each requires appropriate software engineering tools and techniques for their development.
- The fundamental ideas of software engineering are applicable to all types of software system.

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.

ETHICAL DILEMMAS

- Disagreement in principle with the policies of senior management.
- Your employer acts in an unethical way and releases a safety-critical system without finishing the testing of the system.
- Participation in the development of military weapons systems or nuclear systems.

KEY POINTS

- Software engineers have responsibilities to the engineering profession and society.
- They should not simply be concerned with technical issues.
- Professional societies publish codes of conduct which set out the standards of behaviour expected of their members.