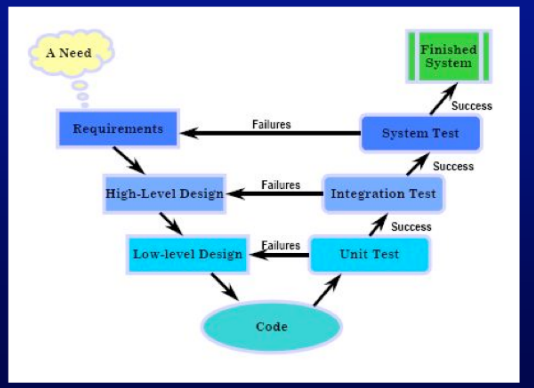
Introduction to Software Engineering

Prof. Lyle N. Long
lnl@psu.edu
http://www.personal.psu.edu/lnl

What is Software Engineering



Sources of Material

- Software Engineering, 7th Edition, by Ian Sommerville
- CSDP Exam Prep Course Notes, Richard Thayer, 2005
- For more info:
  - AERSP 440 course, Software Engineering, by Lyle Long

What is software?

- Computer programs and associated documentation such as requirements, design models and user manuals.
- Software products may be developed for a particular customer or may be developed for a general market.
- Software products may be
  - Generic - developed to be sold to a range of different customers e.g. PC software such as Excel or Word.
  - Custom - developed for a single customer according to their specification.
- New software can be created by developing new programs, configuring generic software systems or reusing existing software.

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.

Number of Jobs

Lockheed Martin (Feb. 2005)

708 Job Openings for recent graduates:

167 in Systems Engineering (23 %)
136 in Software Engineering (19 %)
59 in Mechanical Engineering (8 %)
56 in Information Technology (8 %)
45 in Electrical Engineering (6 %)
21 in Aerospace Engineering (3 %)

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.

What are the attributes of good software?

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

Critical Systems

- Safety-critical systems
  - Failure results in loss of life, injury or damage to the environment;
  - Chemical plant protection system; Aircraft avionics & guidance
- Mission-critical systems
  - Failure results in failure of some goal-directed activity;
  - Spacecraft navigation system;
- Business-critical systems
  - Failure results in high economic losses;
  - Customer accounting system in a bank;

System dependability

- For critical systems, it is usually the case that the most important system property is the dependability of the system.
- The dependability of a system reflects the user's degree of trust in that system. It reflects the extent of the user's confidence that it will operate as users expect and that it will not 'fail' in normal use.
- Usefulness and trustworthiness are not the same thing. A system does not have to be trusted to be useful.

Importance of dependability

- Systems that are not dependable and are unreliable, unsafe or insecure may be rejected by their users.
- The costs of system failure may be very high.
- Undependable systems may cause information loss with a high consequent recovery cost.

Development methods for critical systems

- The costs of critical system failure are so high that development methods may be used that are not cost-effective for other types of system.
- Examples of development methods
  - Formal methods of software development
  - Static analysis
  - External quality assurance

Maintainability

- Ease of repairing the system after a failure has been discovered or changing the system to include new features
- Very important for critical systems as faults are often introduced into a system because of maintenance problems

Survivability

- The ability of a system to continue to deliver its services to users in the face of deliberate or accidental attack
- This is an increasingly important attribute for distributed systems whose security can be compromised
- Survivability subsumes the notion of resilience - the ability of a system to continue in operation in spite of component failures

Dependability vs Performance

- Untrustworthy systems may be rejected by their users
- System failure costs may be very high
- It is very difficult to tune systems to make them more dependable
- It may be possible to compensate for poor performance
- Untrustworthy systems may cause loss of valuable information

Dependability costs

- Dependability costs tend to increase exponentially as increasing levels of dependability are required
- There are two reasons for this
  - The use of more expensive development techniques and hardware that are required to achieve the higher levels of dependability
  - The increased testing and system validation that is required to convince the system client that the required levels of dependability have been achieved

Costs of increasing dependability

Dependability economics

- Because of very high costs of dependability achievement, it may be more cost effective to accept untrustworthy systems and pay for failure costs
- However, this depends on social and political factors. A reputation for products that can't be trusted may lose future business
- Depends on system type - for business systems in particular, modest levels of dependability may be adequate

Elements of Software Engineering

- Professionalism, economics, ethics
- Software requirements
- Software design
- Software construction
- Software testing
- Software maintenance
- Software configuration management
- Software engineering management
- Software engineering processes
- Software engineering tools and methods
- Software quality

All these are covered on the IEEE CSDP Exam

Question: Where does programming fit in?

Programming vs Software Engineering

- Programming ≠ Software Engineering
- Programming without Software Engineering is just hacking

IEEE Software Engineering Exam
(Certified Software Development Professional)

- http://www.computer.org/certification/
- Before taking exam, candidate needs:
  - At least a B.S. degree
  - A minimum of 9000 hours of software engineering experience
  - Must adhere to code of ethics
- Exam is 3.5 hours and is 180 multiple choice questions (closed book)
- Exam costs $450
- Industry group prepares exam questions
- I passed this exam in 2005, a very difficult exam.…

The Software Crisis

- A software failure is a software project that has one or more of the following:
  - Over budget
  - Late
  - Does not satisfy user needs or expectations
    - Does not meet functional or performance requirements
    - Does not meet quality requirements

Examples of Failures

- State of California:
  - $40M DMV project
  - $44M Prison software system
  - $100M State child support system
- $10B FAA modernization project
- UK tax filing system

THEY ALL FAILED !

The Software Crisis

"The construction of new software, which is pleasing to both user and buyer and does not contain errors, is an unexpectedly hard problem. It is perhaps the most difficult problem in engineering today. Referred to as the "software crisis," it has become the longest continuing crisis in the engineering world, and it continues unabated."

W. W. Royce
1929 - 1995

What is causing software crisis?

- Software requirements do not adequately describe user needs or customer expectations
- Project planning is frequently unrealistic, incomplete, or ignored
- Project cost and schedule estimates are underestimated or established by management edict
- Software quality is difficult to specify, design, and build-to
- Software development progress is difficult to see, progress is often unknown
- Changes in requirements are not accompanied by changes in software plans
- Design is changed without changing requirements
- Standards are not used or documented

- How do we solve software engineering crisis?

→ Software Engineering

Software Crisis

"The tragedy of software engineering is not that we don't know how to plan and conduct software projects, but that we know how and just don't do it…."

Richard E. Fairley

Software Crisis

"We can successfully build large reliable systems. Software engineering has worked!"

Ian Sommerville
2000

What is Software Engineering?

- Software Engineering could be more accurately called Software System Engineering, it builds upon System Engineering

What is software engineering?

- Software engineering is an engineering discipline that is concerned with all aspects of software production.
- Software engineers should adopt a systematic and organised approach to their work and use appropriate tools and techniques depending on the problem to be solved, the development constraints and the resources available.

> Developing software without using software engineering is like building a car by just grabbing some tools and metal and building it.

What is the difference between software engineering and computer science?

- Computer science is concerned with theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.
- Software engineering is closer to good business practices than science
- Computer science theories are still insufficient to act as a complete underpinning for software engineering (unlike e.g. physics and electrical engineering).

What is the difference between software engineering and system engineering?

- System engineering is concerned with all aspects of 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.

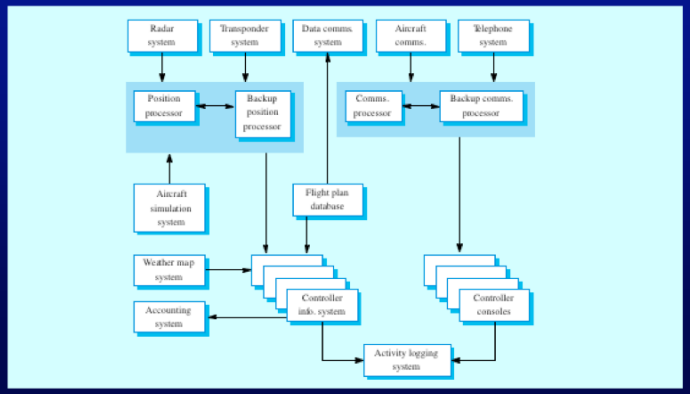
What is Software Engineering?

- Practical application of computer science, management techniques, and other skills to: design, construct, and maintain software and its documentation
- Systematic application of methods, tools, and techniques to achieve a stated requirement or objective for software system
- Application of systems engineering to software development
- Uses engineering discipline to reduce problems of late delivery, cost overruns, and failure to meet requirements
- A means for communicating amongst stakeholders

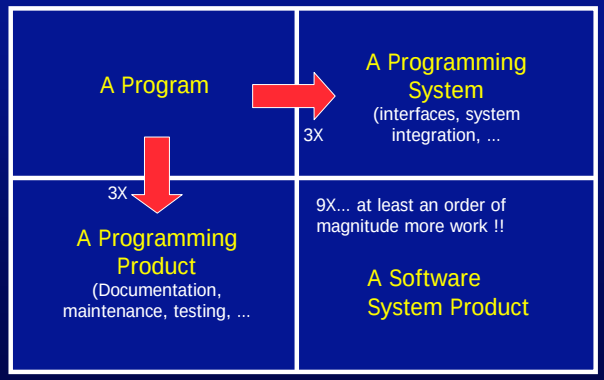
System Engineering

- Problem definition (requirements analysis)
- Solution analysis (software design)
- Process planning
- Process control
- Product evaluation (verification, validation, and testing)

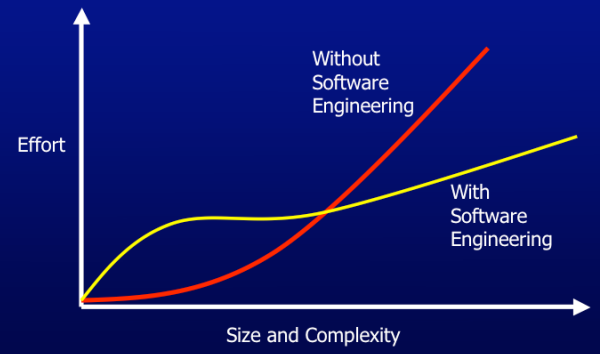
ATC system architecture



Computer Program vs Software Product

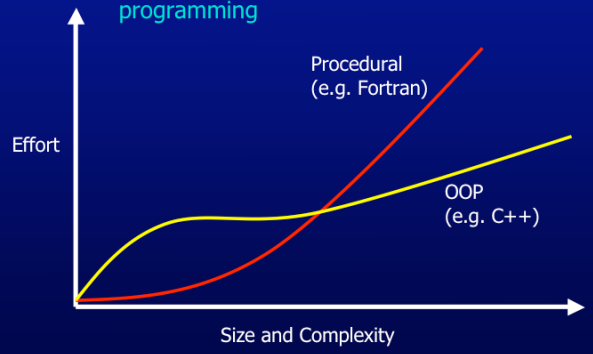


Effort, Software Size, & Complexity



Effort, Software Size, & Complexity

This same figure applies to OOP vs procedural programming



IEEE Software Engineering Standards

- IEEE-Std 1074-1997
- IEEE-Std 1012-1998
- IEEE-Std 829-1998
- IEEE-Std 830-1998
- IEEE-Std 12207.0-1996
- IEEE-Std 12207.1-1997
- IEEE-Std 12207.2-1998

What is a software process?

- A set of activities whose goal is the development or evolution of software.
- Generic activities in all software processes are:
  - Specification - what the system should do and its development constraints
  - Development - production of the software system
  - Verification – checking that the software meets the requirements
  - Validation - checking that the software is what the customer wants or meets intended use
  - Evolution - changing the software in response to changing demands.

V&V = Verification and Validation

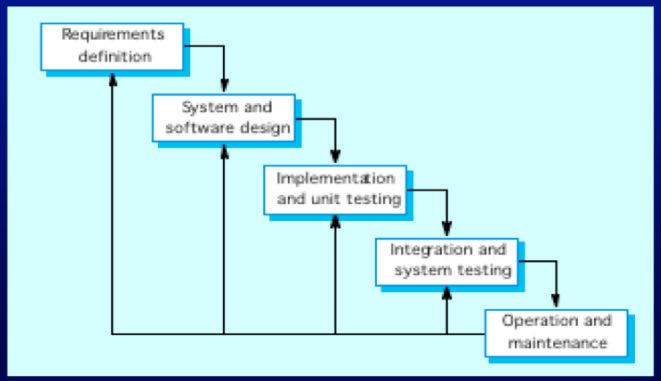
What is a software process model?

- A simplified representation of a software process, presented from a specific perspective.
- Generic process models
  - Waterfall;
  - Iterative development;
  - Component-based software engineering.

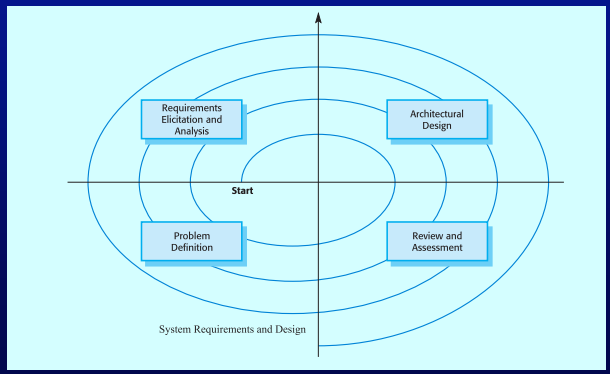
Life Cycle Models

Life Cycle Model	Model Emphasis
Waterfall (conventional)	Project Phases
Incremental	Evolving Product
Evolutionary	Exploratory Development
Spiral	Risk Management
Hacking (code and fix)	Coding

Waterfall model



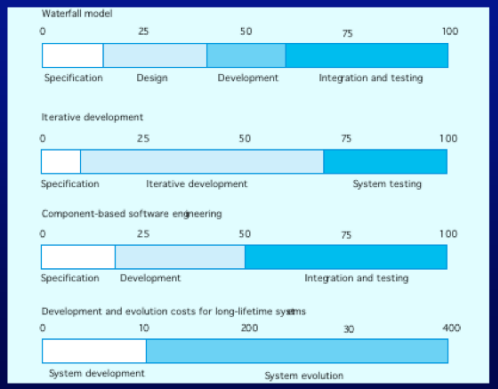
Spiral model of requirements/design



What are the costs of software engineering?

- Roughly 60% of costs are development costs, 40% are testing costs. For custom software, evolution costs often exceed development costs.
- Costs vary depending on the type of system being developed and the requirements of system attributes such as performance and system reliability.
- Distribution of costs depends on the development model that is used.

Activity cost distribution



What are the key challenges facing software engineering?

- Heterogeneity, delivery and trust.
- 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.

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 behaviour is more than simply upholding the law.

Ethics vs Law

- Ethics are personal code of behavior
- Law is minimum standard imposed by society
- Law represents will of majority (violation punishable by government)
- Ethics are suggested, not mandated (violation can result in malpractice suit, loss of job, …)

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 outside their competence level.

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).

IEEE Computer Society 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.

Code of ethics - preamble

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:

Code of ethics - 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.

Code of ethics - principles

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.

Code of ethics - principles

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 engineering is an engineering discipline that is concerned with all aspects of software production.
- Software products consist of developed programs and associated documentation. Essential product attributes are maintainability, dependability, efficiency and usability.
- The software process consists of activities that are involved in developing software products. Basic activities are software specification, development, validation and evolution.
- Methods are organised ways of producing software. They include suggestions for the process to be followed, the notations to be used, rules governing the system descriptions which are produced and design guidelines.

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.

Summary

- Software engineering developed out of necessity to handle large software projects that cannot be handled by few individuals using ad hoc methods
- Goal is to develop software systems that:
  - Satisfy technical requirements, user needs, and acquirer expectations
  - Are developed on time and on budget
  - Are easy to modify and maintain
  - Are developed with pride and personal satisfaction
  - Fulfills all ethical and legal considerations
- Good project management and software engineering processes are required