Software Engineering I

Instructor: Dr. Jerry Gao
Introduction to Software Engineering

- Evolution of software
- About software
  - Software characteristics
  - Software components
  - Software applications
- Software crisis and myths
- Why software engineering?
- What is software engineering?
- Who does software engineering?
Evolution of Software

- The early years:
  - Batch orientation
  - Limited distribution
  - Custom software

- The second era:
  - Multiuser
  - Real-time
  - Database
  - Product software

- The third era:
  - Distributed systems
  - Embedded “intelligence”
  - Low cost hardware
  - Consumer impact

- The fourth era:
  - Powerful desk-top systems
  - Object-oriented technologies
  - Expert systems
  - Artificial neural networks
  - Parallel computing
  - Network computers

The early years

About Software - Characteristics

Software has a dual role. It is a product, but also a vehicle for delivering a product.

Software is a logical rather than a physical system element.

Software has characteristics that differ considerably from those of hardware.

- Software is developed or engineered, it is not manufactured in the classical sense.

- Software doesn’t “wear out”.

- Most software is custom-built, rather than being assembled from existing components.
The graph on the left shows the failure curve for hardware, which initially decreases rapidly, then stabilizes at a lower rate over time.

The graph on the right illustrates the failure curve for software (idealized), suggesting a constant rate of failure until obsolescence.

In the middle graph, the failure curve for software is depicted with an actual curve showing increased failure rates due to side effects, marked by peaks and troughs. The idealized curve is shown as a smooth line.

Overall, the graphs highlight the differences in failure patterns between hardware and software, with hardware showing a more predictable wear-out process, whereas software failure rates can be influenced by side effects and changes over time.
Software Applications

**System Software**- A collection of programs written to service other programs at system level. For example, compiler, operating systems.

**Real-time Software**- Programs that monitor/analyze/control real world events as they occur.

**Business Software**- Programs that access, analyze and process business information.

**Engineering and Scientific Software** - Software using “number crunching” algorithms for different science and applications. System simulation, computer-aided design.

**Embedded Software**- Embedded software resides in read-only memory and is used to control products and systems for the consumer and industrial markets. It has very limited and esoteric functions and control capability.

**Artificial Intelligence (AI) Software**- Programs make use of AI techniques and methods to solve complex problems. Active areas are expert systems, pattern recognition, games

**Internet Software** - Programs that support internet accesses and applications. For example, search engine, browser, e-commerce software, authoring tools.

**Software Tools and CASE environment** - Tools and programs that help the construction of application software and systems. For example, test tools, version control tools.
Software Crisis

In the software industry, we have had a “crisis” that has been with us for close to 30 years.

Meaning of the word “crisis”
--> a turning point in the course of anything; decisive or crucial time, stage or event.
--> the turning point in the course of a disease, when it becomes clear whether the patient will live or die.

What we actually have in software industry is a “chronic affliction”.
It means --> lasting a long time, recurring often, continuing indefinitely.

Software crisis or software affliction
--> a set of problems encountered in software production.
   --- Problems in developing software
   --- Problems in maintain a growing volume of existing software

Typical examples:
- Build a wrong product.
- Project schedule problems
- Cost estimation problems
Software Myths (I)

Many causes of a software affliction can be traced to a mythology that arose during the early history of software development.

-- Software myths propagated misinformation and confusions. They had a no. of attributes that made them insidious.

-- Software managers often under pressure to maintain budgets, keep schedules from slipping, and improve quality.

Myths --> misleading attitudes of people --> serious problems in software production

Management Myths:

“We already have a book that’s full of standards and procedures for building software. Won’t that provide my people with everything they need to know?”

“My people do have state-of-the-art software development tools. After all, we but them the newest computers.”

“If we get behind schedule, we can add more programmers and catch up.”
Software Myths (II)

Customers of a software may be:
- an outside company that has requested software under contract
- a person next to your desk
- an in-house group
- a marketing or sales group

Customer myths -- lead to false expectations (by customers) and ultimately, dissatisfaction with the developers.

Customer Myths:

“A general statement of objectives is sufficient to begin writing programs - we can fill in the details later.”

“Project requirements continually change, but change can be easily accommodated because software is flexible.”
Software Myths (III)

Practitioners:
Planing group - System analysts, system architects
Development group - Software engineers
Verification group - Test engineers, quality assurance group
Support group - Customer supporters, technical supports
Marketing/sales - Marketing people and product sales

Practitioner’s Myths:

“Once we write the program and get it to work, our job is done.”

“Until I get the program ‘running,’ I really have no way of assessing its quality.”

“The only deliverable for a successful project is the working program.”

“The major task of a software engineer is to write a program.”

“Schedule and requirements are the only important things we should concern when we write programs.”
What is Software Engineering?

Although hundreds of authors have developed personal definitions of software engineering, a definition proposed by Fritz Bauer[NAU69] provides a basis:

“[Software engineering is] the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.”

The IEEE [IEE93] has developed a more comprehensive definition when it states:

“Software Engineering: (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1).”
What is Software Engineering?

Pressman’s view:
“Software engineering is a layered technology (Figure 2.1)”

<table>
<thead>
<tr>
<th>Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
</tr>
<tr>
<td>Process</td>
</tr>
<tr>
<td>A quality focus</td>
</tr>
</tbody>
</table>

**Software methods:**
Software engineering methods provide the technical “how to’s” for building software. Methods --> how to encompass a broad array of tasks:
- requirements analysis, design, coding, testing, and maintenance
Software engineering methods rely on a set of basic principles.
What is Software Engineering?

**Software process:**
Software engineering process is the glue that holds:
- technology together
- enables rational and timely development of computer software

Software engineering process is a framework of a set of key process areas. It forms a basis for:
- project management, budget and schedule control
- applications of technical methods
- product quality control

**Software tools:**
- programs provide automated or semi-automated support for the process and methods.
- programs support engineers to perform their tasks in a systematic and/or automatic manner.
What is Software Engineering?
- A Generic View

Engineering --> analysis, design, construction, verification, and management of technical (or social) entities.

Three general phases in software production:
- the definition phase
- the development phase
- the maintenance phase

Definition phase: (during the definition and planning of a software system)
The definition phase focus on what in a software system.
- What information should be processed in the system?
- What functions should be provided in the system?
- What system performance and criteria are required?
- What are the required system behaviors?
- In what way these requirements will be presented or specified?

Three major tasks in the definition phase:
- System or information engineering
- Software project planning
- Requirement analysis
What is Software Engineering?
- A Generic View

Development phase: (during the development of a software system)

The development phase focus on how.
- How the system will be structured?
- How data are to be structured?
- How information will be processed?
- How function is to be implemented?
- How interface are to be characterized?
- How the design will be specified?
- How the testing will be performed?

There are three types of tasks in the development phase:
- software design
- code generation, and
- software testing
What is Software Engineering?
- A Generic View

**Maintenance phase:** -> a software product evolution after system testing. The maintenance phase re-applies the steps of the definition and development phases, but does so in the context of existing software.

The development phase focuses on change of a software system.

- **Error correction** - changes the software to correct defects.

- **Adaptation** - modify the software to accommodate changes to its external environment.

- **Enhancement** - extend the software by adding new functions according to the new requirements from customers/users.

- **Prevention** - preventive maintenance, called software reengineering. It makes the changes of a software so that it can be easily corrected, adapted, and enhanced.
Why Software Engineering?

Objectives:

- Identify new problems and solutions in software production.
- Study new systematic methods, principles, approaches for system analysis, design, implementation, testing and maintenance.
- Provide new ways to control, manage, and monitor software process.
- Build new software tools and environment to support software engineering.

Major Goals:

- To increase software productivity and quality.
- To effectively control software schedule and planning.
- To reduce the cost of software development.
- To meet the customers’ needs and requirements.
- To enhance the conduction of software engineering process.
- To improve the current software engineering practice.
- To support the engineers’ activities in a systematic and efficient manner.