CMPE 331
Software and Software Engineering

Dr. İlknur Dönmez

(Source: Pressman, R. *Software Engineering: A Practitioner's Approach*. McGraw-Hill, 2005)

Resources

- Pressman, R. S. (2005). *Software engineering: a practitioner's approach*. Palgrave Macmillan.

- Sommerville, I. (2004). Software Engineering. International computer science series. *ed: Addison Wesley*.

- Leffingwell, D., & Widrig, D. (2000). *Managing software requirements: a unified approach*. Addison-Wesley Professional.

- Gamma, E. (1995). *Design patterns: elements of reusable object-oriented software*. Pearson Education India.

Grading

- The Percentages of Grades:

4 Assignment	15%
Term Project	25%
Term Project Presentation	5%
2 Quizes	15%
Midterm	20%
Final	20%

Topics Covered

- Professional software development
  - What is meant by software engineering.
- Software engineering ethics
  - A brief introduction to ethical issues that affect software engineering.
- Case studies
  - Some design examples

Professional software development

- Introduction to software engineering
- Software Life Cycle Model
- Requirements Analysis and specifications
- Software Design Issues
- Function-Oriented Software design
- Basic concepts in Object Orientation
- Object Modeling using UML
- Object Oriented Software development
- User Interface Design
- Coding and testing
- Software Project Planning
- Software Project Monitoring and control
- Software reliability and Quality Management
- Software Maintenance

Introduction to Software Engineering

- Dual Role of Software (product has software and the tools that we generate pruduct also has software)
- Software questions haven't changed
- A definition of software
- Differences between hardware and software
- Changing nature of software
- Dealing with legacy software
- Software myths

Dual Role of Software

- Both a product and a vehicle for delivering a product
  - Product
    - Delivers computing potential
    - Produces, manages, acquires, modifies, display, or transmits information
  - Vehicle
    - Supports or directly provides system functionality
    - Controls other programs (e.g., operating systems)
    - Effects communications (e.g., networking software)
    - Helps build other software (e.g., software tools)

Questions About Software Haven't Changed Over the Decades

- Why does it take so long to get software finished?
- Why are development costs so high?
- Why can't we find all errors before we give the software to our customers? (Sometimes it is hard to make tests at its place)
- Why do we spend so much time and effort maintaining existing programs? (Sometimes new expectations in business need new software designs)
- Why do we continue to have difficulty in measuring progress as software is being developed and maintained? (There are lots of ways to write a function or differnt ways to design the software)

A Software includes:

- **Instructions** (computer programs) that when executed provide desired features, function, and performance
- **Data structures** that enable the programs to adequately manipulate information
- **Documents** that describe the operation and use of the programs

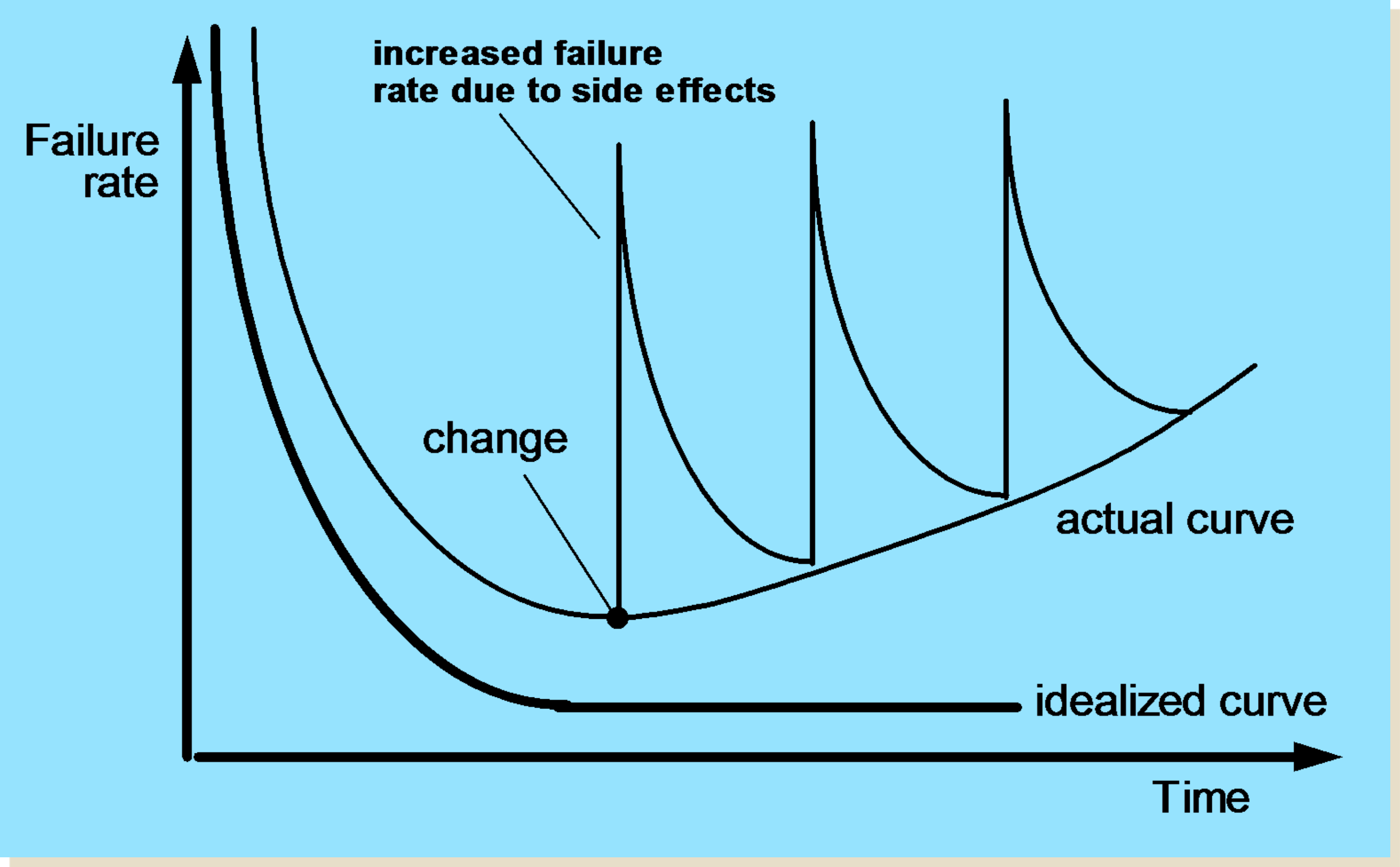
Differences between Software and Hardware

- Software is developed or engineered; it is not manufactured in the classical sense
  - It is harder to manage software projects

- Although the industry is moving toward component-based construction, most software continues to be custom built (it is still complex to build)

- Software doesn't wear out,
  - Hardware bathtub curve compared to the software ascending spiked curve

Software Failure Curve



One correction in a field may cause an error at an other field.

Changing Nature of Software

- System software
- Application software
- Engineering/scientific software
- Embedded software
- Product-line software (e.g., inventory control, word processing, multimedia)
- Artificial intelligence software
- Ubiquitous computing (small, wireless devices, related with internet of things)
- Net sourcing (net-wide computing)
- Open source (operating systems, databases, development environments)
- Web applications (game and other cites that earn money from advertisements, exchange web apps)
- The ".com" marketing applications

Legacy Software - Characteristics

It is some times hard to cope with old big softwares

What should we do:

- Support core business functions
- Have longevity and business criticality (no short time solutions)
- Do not allow poor quality
  - Convoluted code, (no order, no rule, it is mess)
  - poor documentation,
  - poor testing,
  - poor change management (when I change this what can be effected, does it have documentation for change)

Reasons for Evolving the Legacy Software

When we make a change in software what should be consider?

- (Adaptive) Must be adapted to meet the needs of new computing environments or more modern systems, databases, or networks
- (Perfective) Must be enhanced to implement new business requirements
- (Corrective) Must be changed because of errors found in the specification, design, or implementation

(Note: These are also the three major reasons for any software maintenance)

Software Myths - Management

- "We already have a book that is full of standards and procedures for building software. Won't that provide my people with everything they need to know?"
  - Not used, not up to date, not complete, not focused on quality, time, and money
- "If we get behind, we can add more programmers and catch up"
  - Adding people to a late software project makes it later
  - Training time, increased communication lines
- "If I decide to outsource the software project to a third party, I can just relax and let that firm build it"
  - No body knows your core business details.
  - Software projects need to be controlled and managed

Software Myths - Customer

- "A general statement of objectives is sufficient to begin writing programs – we can fill in the details later"
    - Not clear statement of objectives spells disaster
    - Some small details may change the time and cost of project
- "Project requirements continually change, but change can be easily accommodated because software is flexible"
    - Impact of change depends on where and when it occurs in the software life cycle (requirements analysis, design, code, test)
    - If the change is seen in the test stage, all the stage should be controlled and repeated.

Software Myths - Practitioner

- "Once we write the program and get it to work, our job is done"
  - 60% to 80% of all effort expended on software occurs after it is delivered
- "Until I get the program running, I have no way of assessing its quality"
  - Formal technical reviews of requirements analysis documents, design documents, and source code (more effective than actual testing)
- "The only deliverable work product for a successful project is the working program"
  - Software, documentation, test drivers, test results
- "Software engineering will make us create voluminous and unnecessary documentation and will invariably slow us down"
  - Creates quality, not documents; quality reduces rework and provides software on time and within the budget

A new developer in a project always complain about the lack of documentation. But when he or she become an expert software developer, does not want to write software documents.