Lecture 1

- **Introduction to Software Engineering**

Software Engineering:
A Practitioner's Approach, 8/e (McGraw-Hill 2015).

Table of Contents

- Defining Software
- Definition of Software Engineering
- Characteristics of a Software Engineers
- Software Engineering Domains
- Software Categories: WebApps, Mobile, Cloud, Product-Line Software.

1.Defining Sofware

What is Software?

Textbook description: Software is a set of items or objects that form a “configuration” that includes

(1) instructions (computer programs) that when executed provide desired features, function, and performance;

(2) data structures that enable the programs to adequately manipulate information

(3) documentation that describes the operation and use of the programs.

What is Software?

- *Software is developed or engineered, it is not manufactured in the classical sense.*
- *Software doesn't "wear out."*
- *Although the industry is moving toward component-based construction, most software continues to be custom-built.*
- Customized products
  - Software that is commissioned by a specific customer to meet their own needs.
  - E.g. – embedded control systems, air traffic control software, traffic monitoring systems.

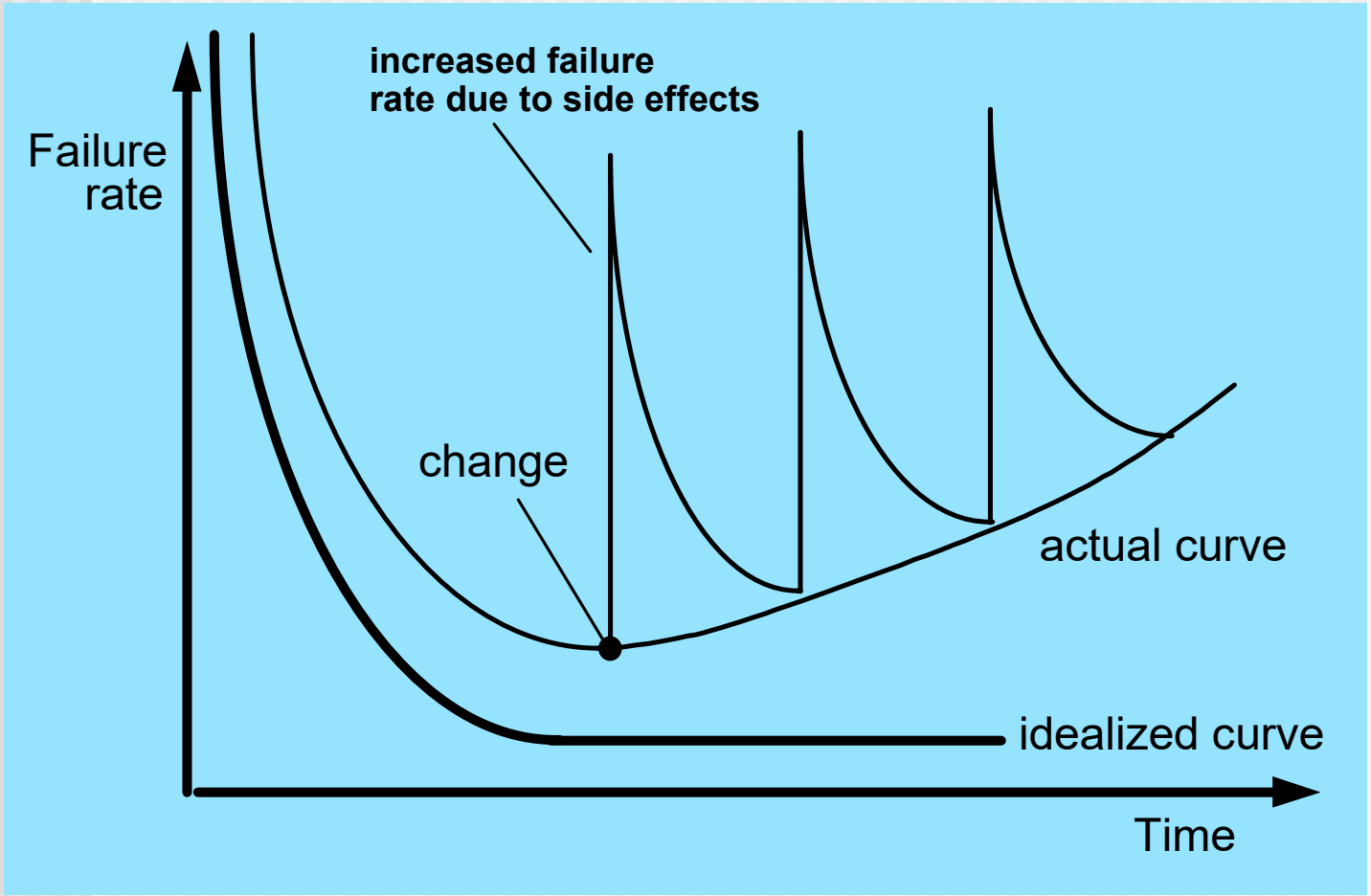
Examples of Software?

- Google?
- Gmail?
- Facebook?
- CAMSys?
- WhatsApp?
- Waze?
- Word?
- Notepad?

Software's Dual Role

- Software is a product
  - Delivers computing potential
  - Produces, manages, acquires, modifies, displays, or transmits information
- Software is a vehicle for delivering a product
  - Supports or directly provides system functionality
  - Controls other programs (e.g., an operating system)
  - Effects communications (e.g., networking software)
  - Helps build other software (e.g., software tools)

Wear vs. Deterioration



2. Definition of Software Engineering

Software Engineering

- The seminal definition:
  - *[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 definition:
  - *(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).*

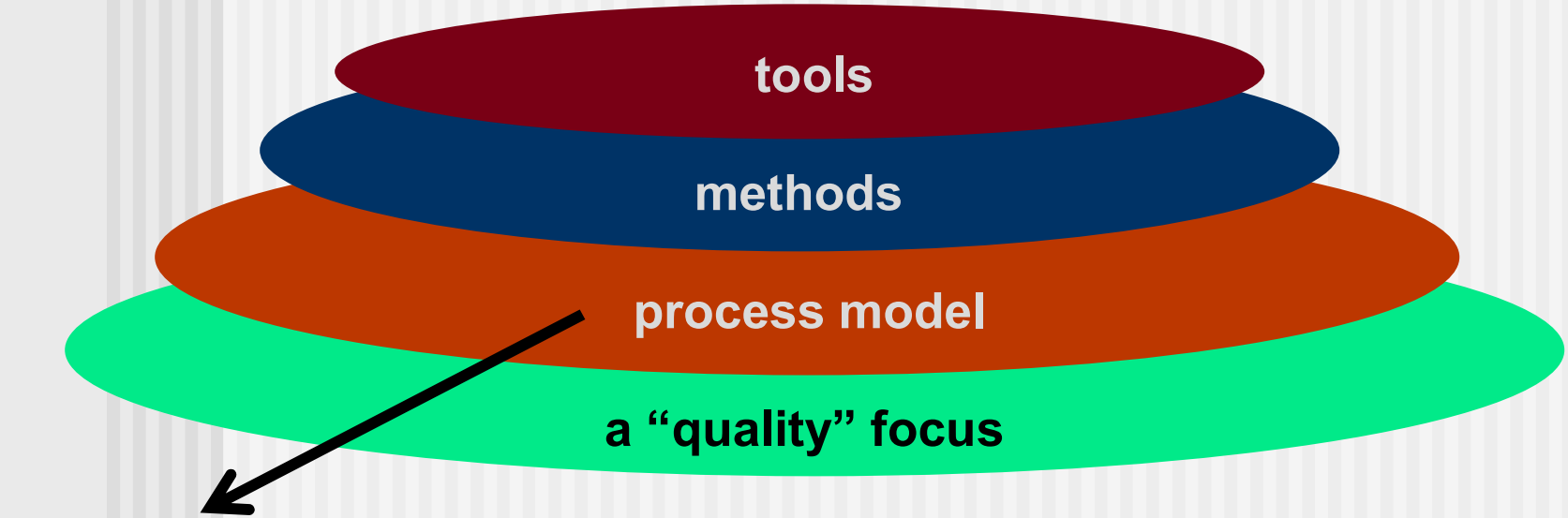
Importance of Software Engineering

- The number of people who have an interest in the features and functions provided by an application have grown dramatically - *a concerted effort should be made to understand the problem before a software solution is developed*
- The requirements have grown complex, the programs have to be developed by large teams of people and implementation involved many possible platforms and devices - *design becomes a pivotal activity*

Importance of Software Engineering

- Individuals, businesses, and governments rely on software for strategic and tactical decision making as well as day-to-day operations and control. There is great impact if the software fails - *software should exhibit high quality*
- As the perceived value of an application grows, it is likely that its user base and longevity grows, and demands for adaptation and enhancement will grow - *software should be maintainable*

Software Engineering Layers



management control
of software projects

Software Engineering Practice

- Polya suggests:
  1. *Understand the problem* (communication and analysis).
  2. *Plan a solution* (modeling and software design).
  3. *Carry out the plan* (code generation).
  4. *Examine the result for accuracy* (testing and quality assurance).

Understand the Problem

- *Who has a stake in the solution to the problem?* That is, who are the stakeholders?
- *What are the unknowns?* What data, functions, and features are required to properly solve the problem?
- *Can the problem be compartmentalized?* Is it possible to represent smaller problems that may be easier to understand?
- *Can the problem be represented graphically?* Can an analysis model be created?

Plan the Solution

- *Have you seen similar problems before?* Are there patterns that are recognizable in a potential solution? Is there existing software that implements the data, functions, and features that are required?
- *Has a similar problem been solved?* If so, are elements of the solution reusable?
- *Can subproblems be defined?* If so, are solutions readily apparent for the subproblems?
- *Can you represent a solution in a manner that leads to effective implementation?* Can a design model be created?

Carry Out the Plan

- *Does the solution conform to the plan?* Is source code traceable to the design model?
- *Is each component part of the solution provably correct?* Has the design and code been reviewed, or better, have correctness proofs been applied to algorithm?

Examine the Result

- *Is it possible to test each component part of the solution?* Has a reasonable testing strategy been implemented?
- *Does the solution produce results that conform to the data, functions, and features that are required?* Has the software been validated against all stakeholder requirements?

Software Process

- A process is a collection of activities, actions, and tasks that are performed to create some work product
- Software process – an adaptable approach that enables the software team to pick and choose the appropriate set of work actions and tasks
- To deliver software in a timely manner and with sufficient quality to satisfy those who have sponsored its creation and those who will use it

Process Framework

The framework for software engineering process consists of:

- Framework activities – applicable to all software projects
- Umbrella activities – applicable across entire software process

For different projects the process may vary, for example the overall flow of activities, degree to which actions and tasks are defined, etc.

A Process Framework

Process framework

Framework activities

- work tasks
- work products
- milestones & deliverables
- QA checkpoints

Umbrella Activities

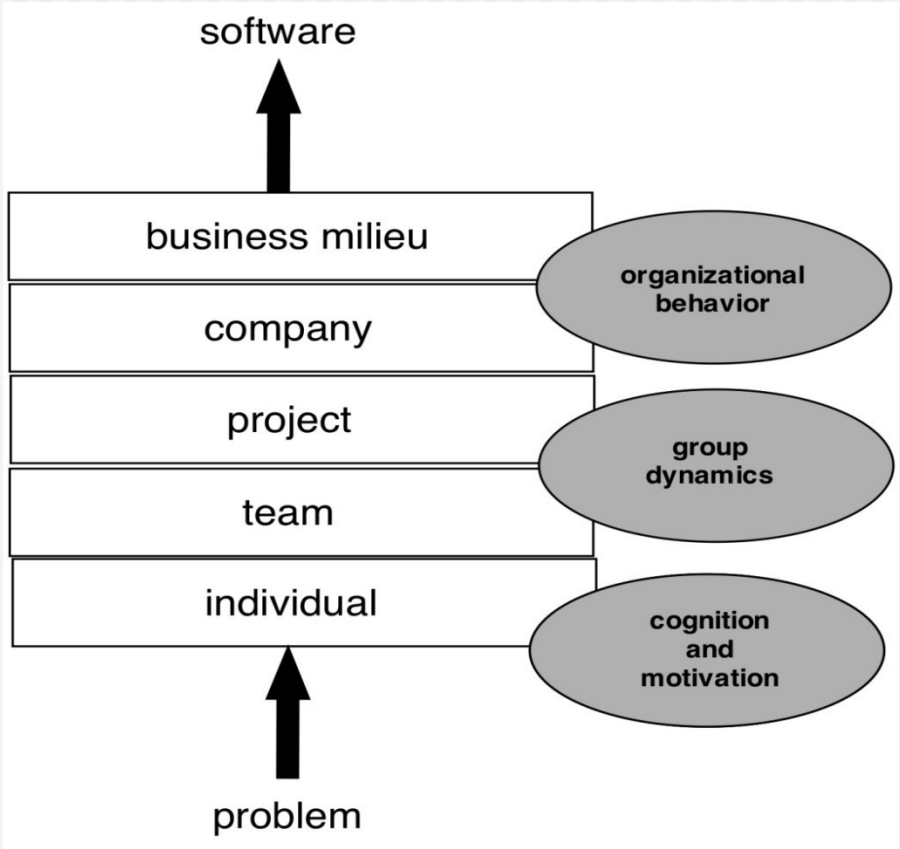
Framework Activities

- Communication
- Planning
- Modeling
  - To better understand requirements
  - Design to achieve the requirements
- Construction
  - Code generation
  - Testing
- Deployment

Umbrella Activities (Manage & Control)

- Software Project Tracking and Control
- Risk Management
- Software Quality Assurance
- Technical Reviews
- Measurement
- Software Configuration Management
- Reusability Management
- Work Product Preparation and Production

Behavioral Model for Software Engineering



3. Characteristics of a Software Engineer

Traits of Successful Software Engineers

- Sense of individual responsibility
- Acutely aware of the needs of team members and stakeholders
- Brutally honest about design flaws and offers constructive criticism
- Resilient under pressure
- Heightened sense of fairness
- Attention to detail
- Pragmatic

4. Software Engineering Domains

Software Application Domains

- System Software: a collection of programs written to service other programs (e.g.- compiler, editor, file management utilities)
- Application Software: stand-alone program that solve a specific business need (e.g.- POS, inventory control, cinema ticketing)
- Engineering/Scientific Software: ”number-crunching” programs that support engineering/scientific work (e.g. astronomy, volcanology, automotive stress analysis, orbital dynamics, CAD)
- Embedded Software: resides within a product/system and perform limited functions (e.g.- fuel control, braking system, dashboard display, keypad control for microwave oven)

Software Application Domains

- Product-Line Software: software designed to provide specific capability for use by many different customers (e.g. inventory control products)
- Web/Mobile Applications: browser-based apps and software that resides on mobile devices
- AI Software: makes use of nonnumerical algorithms to solve complex problems that are not amenable to computation or straightforward analysis (e.g. robotics, expert systems, pattern recognition, artificial neural networks, game playing)

Legacy Software

Why must it change?

- Software must be adapted to meet the needs of new computing environments or technology.
- Software must be enhanced to implement new business requirements.
- Software must be extended to make it interoperable with other more modern systems or databases.
- Software must be re-architected to make it viable within a network environment.

5. Software Categories: WebApps, Mobile, Cloud, Product-Line Software

Software Categories

These categories of software have dominated the industry at the moment:

- WebApps — Web-based systems and applications that not only provides standalone function to the end user, but integrated with corporate databases and business applications
- Mobile Applications — software specifically designed to reside on a mobile platform
- Cloud Computing — an infrastructure or “ecosystem” that enables any user, anywhere, to use a computing device to share computing resources on a broad scale
- Product Line Software — a set of software-intensive systems that share a common, managed set of features satisfying specific needs of a particular market segment or mission and are developed from a common set of core assets

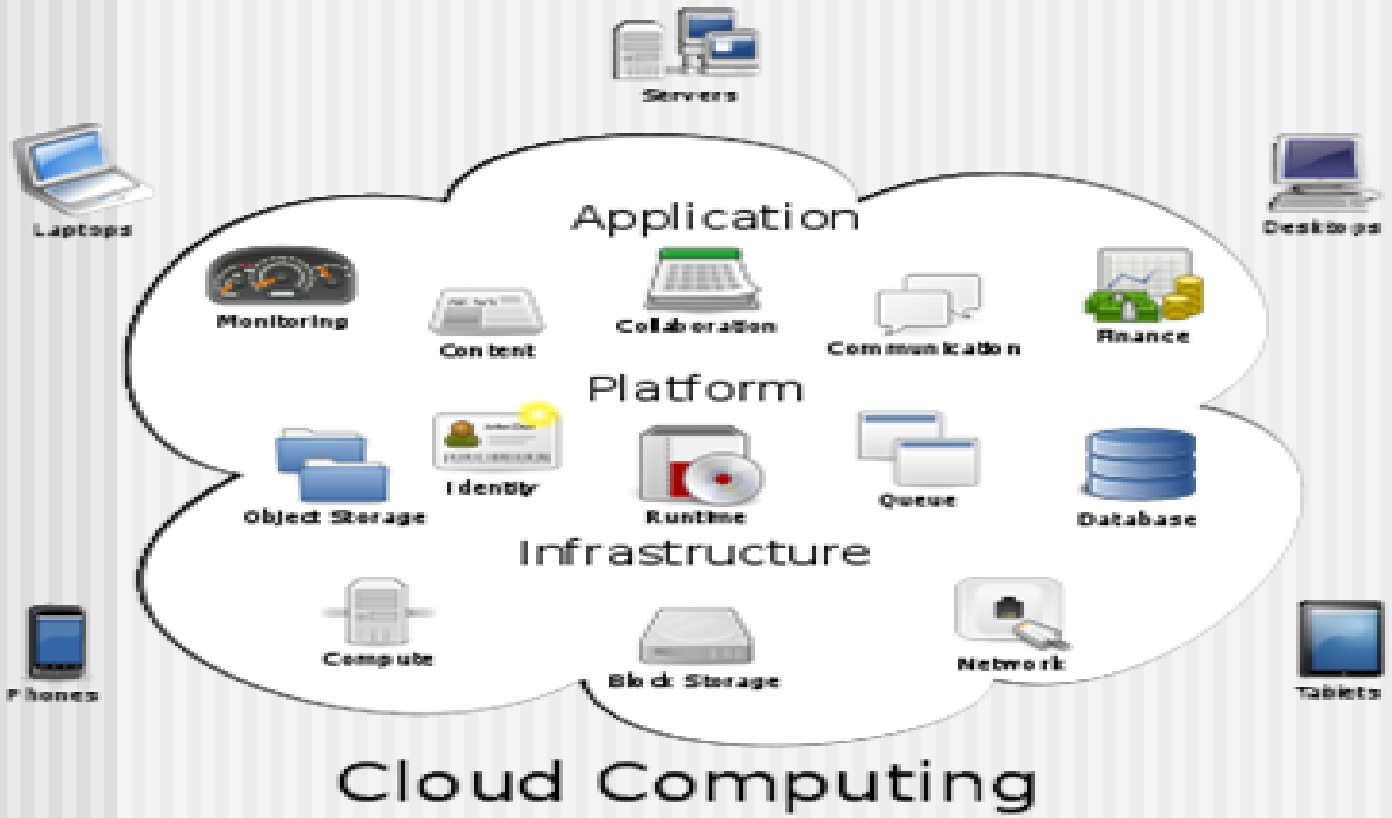
WebApps

- Modern WebApps are much more than hypertext files with a few pictures
- WebApps are augmented with tools like XML and Java to allow Web engineers including interactive computing capability
- WebApps may standalone capability to end users or may be integrated with corporate databases and business applications
- Semantic web technologies (Web 3.0) have evolved into sophisticated corporate and consumer applications that encompass semantic databases that require web linking, flexible data representation, and application programmer interfaces (API's) for access
- The aesthetic nature of the content remains an important determinant of the quality of a WebApp.

Mobile Apps

- Reside on mobile platforms such as cell phones or tablets
- Contain user interfaces that take both device characteristics and location attributes
- Often provide access to a combination of web-based resources and local device processing and storage capabilities
- Provide persistent storage capabilities within the platform
- A *mobile web application* allows a mobile device to access to web-based content using a browser designed to accommodate the strengths and weaknesses of the mobile platform
- A *mobile app* can gain direct access to the hardware found on the device to provide local processing and storage capabilities
- As time passes these differences will become blurred

Cloud Computing



Cloud Computing

- *Cloud computing* provides distributed data storage and processing resources to networked computing devices
- Computing resources reside outside the cloud and have access to a variety of resources inside the cloud
- Cloud computing requires developing an architecture containing both frontend and backend services
- Frontend services include the client devices and application software to allow access
- Backend services include servers, data storage, and server-resident applications
- Cloud architectures can be segmented to restrict access to private data

Product Line Software

- *Product line software* is a set of software-intensive systems that share a common set of features and satisfy the needs of a particular market
- These software products are developed using the same application and data architectures using a common core of reusable software components
- A software product line shares a set of assets that include *requirements, architecture, design patterns, reusable components, test cases,* and other work products
- A software product line allow in the development of many products that are engineered by capitalizing on the commonality among all products with in the product lin