QUEZON CITY UNIVERSITY
COLLEGE OF COMPUTER STUDIES

WEEK 2
SOFTWARE AND SOFTWARE ENGINEERING

SE 101 – SOFTWARE ENGINEERING

LEARNING OUTCOMES:

- Explain the nature of Software Engineering.
- Comprehend the software process.
- Determine how software engineering started.

TOPICS

Software and Software Engineering

- The Nature of Software
- Software Engineering
- The Software Process
- Software Engineering Practice
- Software Engineering Myths
- How It All Starts

The Nature of Software Engineering

Software is

(1) instructions (computer programs) that when executed provide desired features, function, and performance; (2) data structures that enable the programs to adequately manipulate information, and (3) descriptive information in both hard copy and virtual forms that describes the operation and use of the programs.

Software has characteristics that are considerably different than those of hardware:

1. Software is developed or engineered; it is not manufactured in the classical sense.
2. Software doesn't "wear out."
3. Although the industry is moving toward component-based construction, most software continues to be custom built.

Software Application Domains

- System Software
- Application software
- Engineering/Scientific software
- Embedded software
- Product-line software
- Web applications
- Artificial intelligence software

- **System Software**
  - a collection of programs written to service other programs. Some system software (e.g., compilers, editors, and file management utilities) process complex, but determinate, information structures. Other systems applications (e.g., operating system components, drivers, telecommunications processors) process largely indeterminate data.

- **Application Software**
  - stand-alone programs that solve a specific business need. Applications in this area process business or technical data in a way that facilitates business operations or management/technical decision making.

- **Engineering/scientific Software**
  - Engineering and scientific software have been characterized by "number crunching" algorithms. Applications range from astronomy to volcanology, from automotive stress analysis to space shuttle orbital dynamics, and from molecular biology to automated manufacturing.

- **Embedded software**
  - resides within a product or system and is used to implement and control features and functions for the end user and for the system itself.
- **Product-line software**
  - designed to provide a specific capability for use by many different customers.

- **Web applications**
  - called "WebApps," this network-centric software category spans a wide array of applications. In their simplest form, WebApps can be little more than a set of linked hypertext files that present information using text and limited graphics.
- **Artificial Intelligence (AI) software**
  - makes use of nonnumerical algorithms to solve complex problems that are not amenable to computation or straightforward analysis. Applications within this area include robotics, expert systems, pattern recognition (image and voice), artificial neural networks, theorem proving, and game playing.

The Evolving Role of Software

- **It is a product and, at the same time, the vehicle for delivering a product.**
  - **As a product,** it delivers the computing potential embodied by computer hardware or, more broadly, a network of computers that are accessible by local hardware.
  - **As the vehicle used to deliver the product,** software acts as the basis for the control of the computer (operating systems), the communication of information (networks), and the creation and control of other programs (software tools and environments).

SOFTWARE ENGINEERING

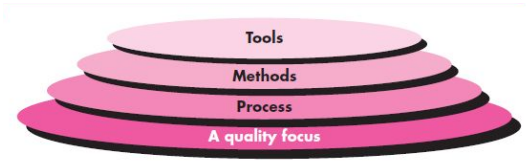
- Software has become deeply embedded in virtually every aspect of our lives.
- The information technology requirements demanded by individuals, businesses, and governments grow increasing complex with each passing year.
- Individuals, businesses, and governments increasingly rely on software for strategic and tactical decision making as well as day-to-day operations and control.
- As the perceived value of a specific application grows, the likelihood is that its user base and longevity will also grow.

Software Engineering is an engineering approach on a software development of systematic application. It is a process of analyzing user needs and designing, constructing, and testing end-user applications that will satisfy these needs through the use of software programming languages.

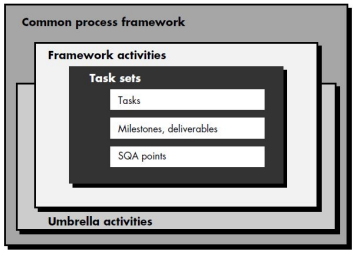
The IEEE has developed a more comprehensive definition when it states:

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).

A Layered Technology



THE SOFTWARE PROCESS



A generic process framework for software engineering encompasses five activities:

- Communication
- Planning
- Modeling
- Construction
- Deployment

Typical umbrella activities include:

- Software project tracking and controlPlanning
- Risk management
- Software quality assurance
- Technical reviews
- Measurement
- Software configuration management
- Reusability management
- Work product preparation and production

Software Engineering Practice

The essence of practice:

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).

Software Engineering Ethics

Ethical behavior is more than simply upholding the law but involves following a set of principles that are morally correct.

- 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.

Issues of Professional Responsibility

- Confidentiality
- Competence
- Intellectual property rights
- Computer misuse

Legacy Software

Systems were developed decades ago and have been continually modified to meet changes in business requirements and computing platforms. The proliferation of such systems is causing headaches for large organizations who find them costly to maintain and risky to evolve.

SOFTWARE MYTHS

- **Management myths**

Managers with software responsibility, like managers in most disciplines, are often under pressure to maintain budgets, keep schedules from slipping, and improve quality.

- **Customer myths**

A customer who requests computer software may be a person at the next desk, a technical group down the hall, the marketing/sales department, or an outside company that has requested software under contract.

- **Practitioner's myths**

Myths that are still believed by software practitioners have been fostered by 50 years of programming culture.
During the early days of software, programming was viewed as an art form.

How It All Starts

Every software project is precipitated by some **business need**—the need to correct a defect in an existing application; the need to adapt a “legacy system” to a changing business environment; the need to extend the functions and features of an existing application; or the need to create a new product, service, or system.

END OF PRESENTATION.
THANK YOU!