INTRODUCTION TO SOFTWARE ENGINEERING

Dr.Lê Hùng Tiến

VLU

Introduction To Software Engineering

Carnegie Mellon University
The Practical Software Engineering Series

Software Engineering: A Perspective

Course Objective

- Upon completion of this course, students will have the ability to:
  - Understand the discipline and principles of Software Engineering.
  - Understand the evolution of software in industry and global competitive trends.
  - Understand software process, product and services.
  - Understand software modeling & techniques.
  - Demonstrate an appreciation for the breadth of software engineering.

Lecture Learning Objectives

- Upon completion of this lecture, students will be able to:
  - Understand the meaning of Software Engineering.
  - Appreciate the discipline of Software Engineering.

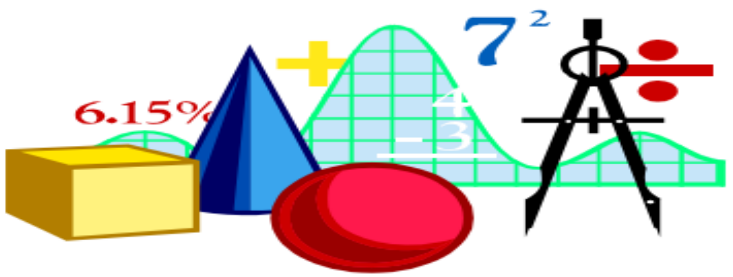
- Outcomes:
  - Take a broad view of Software Engineering as compared with other fields.
  - Demonstrate an understanding of Software Engineering and how it is needed by the software industry.

Software Engineering Definitions

- The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software.
- An engineering discipline that is concerned with all aspects of software development.
- The establishment and use of good engineering principles in order to economically obtain software that is reliable and works efficiently.
- The use of techniques, methods, and methodologies to develop quality software products under constraints of time, budget and resources.

Software Engineering

- Software engineering involves creating cost-effective solutions to practical problems, and preferentially applying scientific knowledge to the building of software systems.
- Software engineering is often confused with programming. Programming is only a subset of software engineering because the responsibilities of a software engineer are aimed at the purposeful creation of software that satisfies a wide range of technical, business, and regulatory requirements, not the ability to create code.



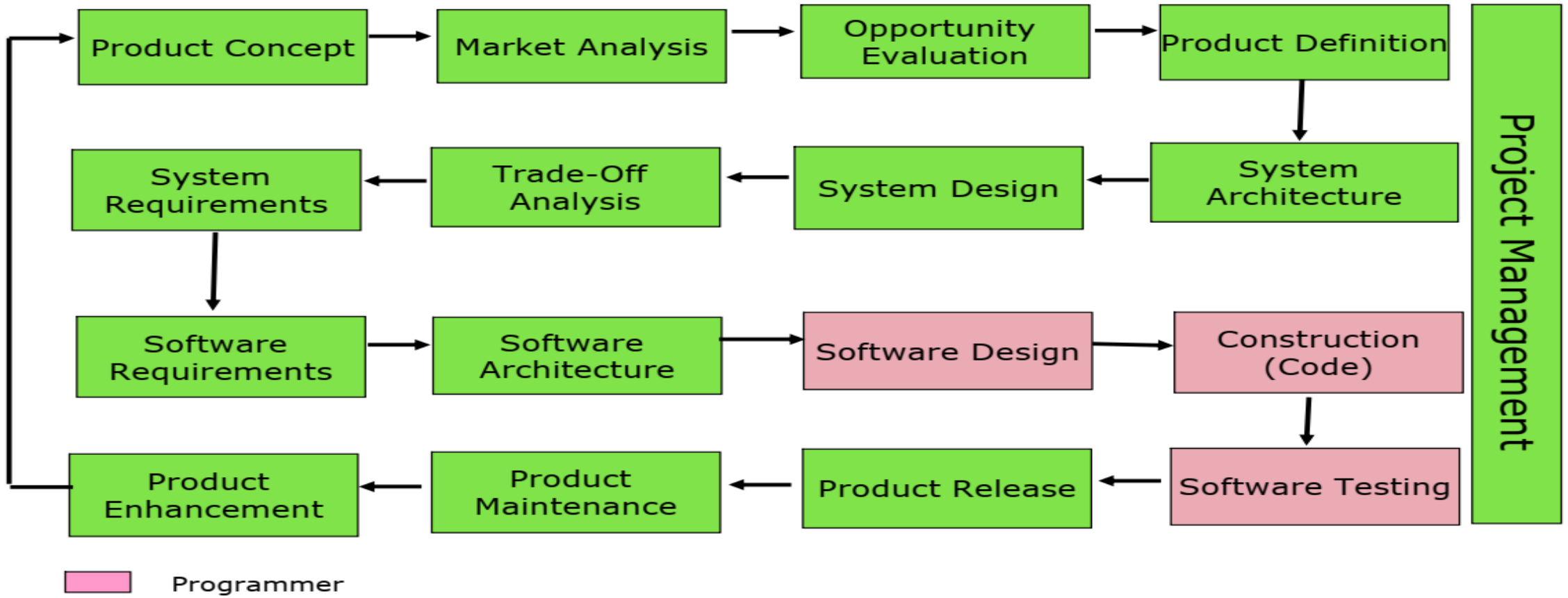
Software Engineer vs. Programmer

- Both software engineers and programmers work on software projects.
- Programmers work on the task of writing code to produce software applications.
- Software engineers focus on the entire software development process by understanding how to support a business context with relationships between marketing, sales, production, installation, training, and operations; using methods to integrate components into a large application or final product that satisfies business needs.

Some Comparisons

Issue	Software Engineer	Programmer
Scope	The entire project from requirements to final products	Coding, some design and tests
Context	Teamwork, collaborate with all stakeholders	Individual work, coordinate with team members
Focus	Software product & application for business solution	Project focus
Skills	Domain knowledge	Mathematics and programming

Example: Product Development Process



Computer Science vs. Software Engineering

- Computer Science (CS) programs focus on the principles and applications of computing with emphasis on data, data structures, algorithms, computing theory, some problem solving techniques and design methodologies.
- Most Computer Science students are trained in theories, computer languages, structure, and translation but not on the total concept of software development from product concept to release and maintenance services and how to apply an "Engineering discipline" to ensure product quality to meet business needs.

Computer Science Focus - 1

- Mathematical logic (Boolean logic and modeling logical queries)
- Number theory (Theory of proofs and heuristics in integers)
- Graph theory (Foundations for structures and searching algorithms)
- Type theory (Formal analysis of the types of data)
- Category theory (Math and computation synthesis)
- Computational geometry (The study of algorithms to solve problems stated in terms of geometry)
- Numerical analysis (Foundations for algorithms in discrete mathematics, as well as the study of the limitations of floating point computation, including round-off errors)

Computer Science Focus - 2

- Automata theory (Different logical structures for solving problems)
- Computability theory (Calculable with the models of computers)
- Computational complexity theory (Time and storage space)
- Analysis of algorithms (Complexity of algorithms)
- Algorithms (Formal logical processes for computation)
- Data structures (The organization of and rules for the manipulation of data)

Computer Science Focus - 3

- Compilers
  - The translating of computer programs, usually from higher level languages to lower level ones.
- Interpreters
  - A program that takes a computer program in as input and executes it.
- Programming languages
  - Formal language for expressing algorithms, and the properties of these languages.
    - For example: Pascal, C, Java

Software Engineering Focus - 1

- Algorithm implementation (Using ideas from algorithms to design solutions.)
- Computer programming (Using a programming language to implement algorithms.)
- Formal methods (Mathematical approaches for describing software designs.)
- Reverse engineering (The application of the scientific method to the understanding of arbitrary existing software.)
- Software development process (The principles and practice of obtaining requirements: architect, design, develop, test, release, as well as proper engineering practices to meet quality and business needs.)

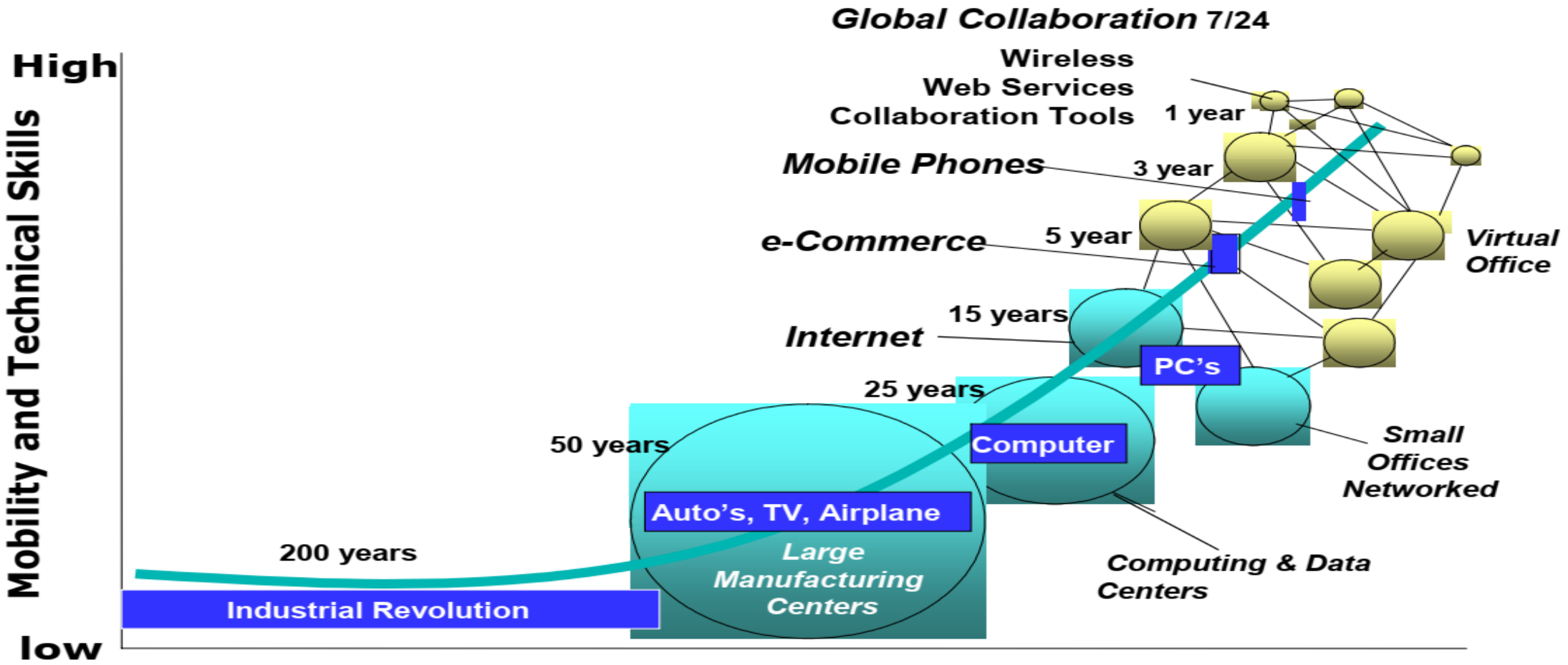
Software Engineering Focus - 2

- Software Engineering Disciplines:
  1. Requirements engineering
  2. Software architecture
  3. Software design
  4. Software construction
  5. Software verification and validation
  6. Software configuration management
  7. Software quality assurance
  8. Software project management
  9. Software portfolio management

Software Engineering Focus - 3

- Software Applications:
  1. System software
  2. Real-time software
  3. Business software
  4. Scientific software
  5. Embedded software
  6. Artificial intelligence software
  7. Personal computer software

The World Is Changing Faster



New Era, New Rules …

- The key factor in economic growth today is the development of the **knowledge industry** to create information and communication products.
- The defining measurement is **Speed** – the faster, the better.
- Moore's law: Computing power doubles every two years, technical knowledge changes every seven years.

What Is Changing In Software Fields?

1. Programming languages
2. Common elements
3. Design representations and models
4. Tools, process, and teams
5. Architecture, frameworks, libraries
6. New disciplines & new business processes

Programming Languages

Programming languages

- Language evolution:
  - Examples
    - C to C++
    - Fortran to Fortran 4
    - Cobol to 4GL and to App servers
    - C to Ada95, Java, C#
    - Java to Java 5
    - (JCL) to TCL, Perl, Python
    - HTML to JS, XHTML and to AJAX

Common Elements

Common elements:

- Rising tide of more abstraction
  - Intrinsic: what is built in
  - Definitions: what can be added by developers
- Conventionalization
  - Agreements: essential intrinsic, control flows
  - Abstraction and representation
  - *Examples:*
    - math libraries
    - graphics libraries
    - collections libraries
    - IO libraries, etc.

Design Representation & Models

- Higher-level design models:
  - Precision in models
  - Finite state models, state charts
  - Class diagrams
  - Structural models
  - Commonality in models
  - Diverse similar design notations: UML
  - New kinds of models
  - Explicit architecture
  - Requirements: use cases, domain objects, etc.
- Formal design models:
  - Precise models: finite state models
  - Attribute-specific models (SAL, JSR models)

Tools, Process and Teams

- Team-support tools widely adopted
  - Commercial and open source
- What is a "software organization"?
  - Geographically distributed teams
  - Technology support for collaboration
- Manage the software process:
  - Maturity Models (CMM, CMMI, etc.)
  - Rapid iteration, at scale: Plan-Driven, Agile, Xtreme

Disciplines, Processes

- Requirements engineering
  - Flexibility – iterative development
  - Adaptiveness – systems that can respond to change
  - Usability – human integration activity for ease of use
- Software architecture
  - Architectural frameworks
  - Data commonality: RDBMS, metadata, XML
  - System Integration
- Business processes
  - Information technology is the "central nervous system" of business. (i.e., ERP, CRM, SCM systems)
  - Integration across organizations - global enterprises

Issue: Obsolete Education System – 1

- India's software association, NASSCOM commissioned a survey in 2005 which found that:

1. Nearly 70% of graduates from India do not have the skills required for traineeships in the software industry.

2. Two out of every three graduates would be unable to make a satisfactory contribution to the industry for lack of the requisite analytical or basic business context – they only have basic programming skills.

Issue: Obsolete Education System – 2

- China Software Industry Association (CSIA) study in 2005 found:

1. Many graduates are focusing on the programming aspect but lack software engineering skills, resulting in poor quality of products & services.

2. Most graduates do not have good communication skills, especially in foreign languages (English).

3. Over 75% of graduates will NOT be able to work in the software industry without additional training.

Issue: Obsolete Education System – 3

- Research from accreditation of undergraduate programs in U.S and U.K found that:

1. Many graduates do not have the skills to do work required by the software industry.
2. Many university's curricula have changed very little although technology has changed dramatically.
3. Knowledge of contemporary issues is seldom discussed.
4. Many curricula are still focusing on programming not on Software Engineering which is needed in the global economic, environmental and social context.

Software Industry Issues

- 45% to 68% of software project costs can be correlated to people and organization factors.
- People Factors:
  - Knowledge & Skills
  - Experience
  - Performance & Motivation
- Organization Factors:
  - Compensation
  - Career Development
  - Training

Software Industry vs. Academia

- Education institutions are in the training business where students are taught general knowledge and theory related to their field.
- Industry that hires graduated students must train them on specific applications and business processes where they apply their academic knowledge to the specific industrial job.
- Depending on the industry, job training can last between 6 months to 2 years. Workers are not productive until they master their skills.
- Industry must invest (time and money) in workers during this transition period.

Software Industry Requirements

- Today, education institutions are shaped by economic forces to provide skilled workers to industry.
- Industry exerts influence on education institutions as to what should be taught and how students must learn so they will be ready to do work productively once they graduate.
- Industry will hire workers that can meet their skill requirements. Students will go to schools that have a strong reputation in fields that have high placement rates.
- A school's reputation is a key for career and job placement.

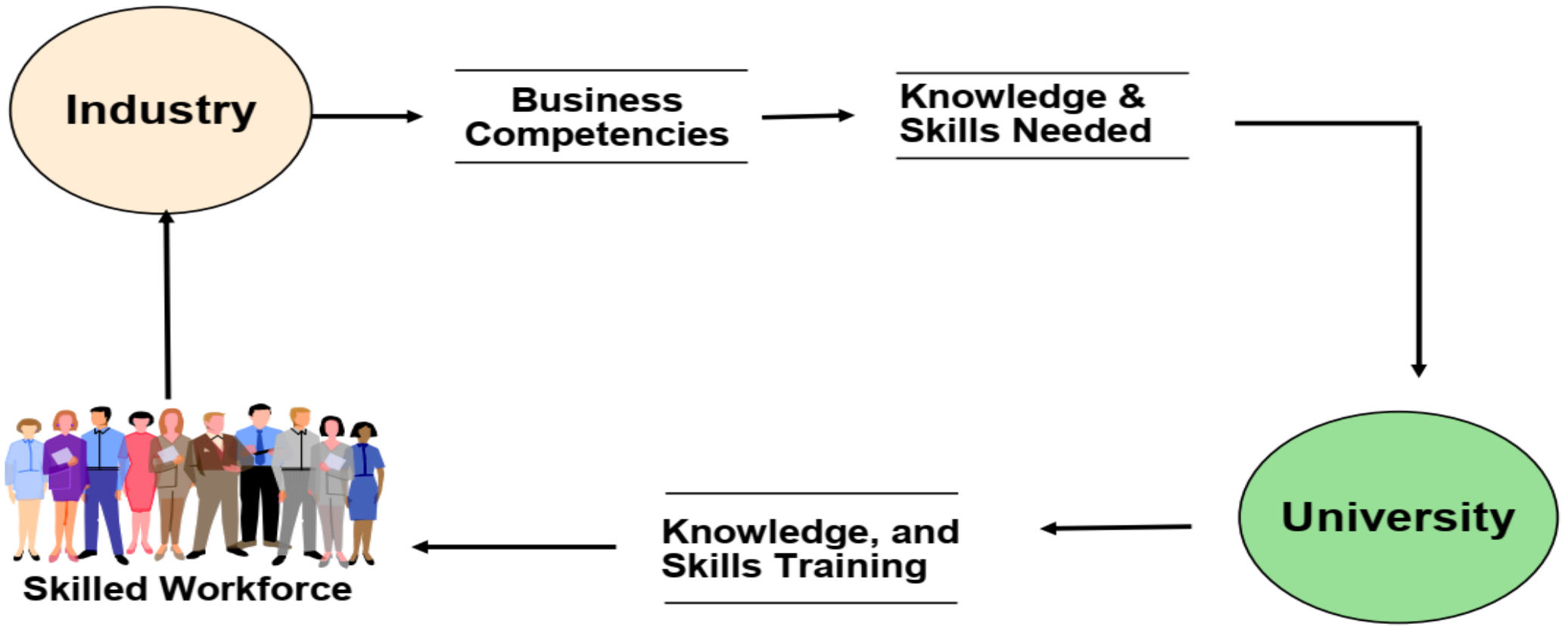
Software Industry's Requirements

- Curriculum must focus on a **disciplined approach** that covers the entire life cycle or software activities from the conceptual idea to product release and maintenance service.
- Although programming languages are the foundation, management and the application of tools and methods to solve problems must be also be **emphasized,** for new graduates, to be better prepared to contribute to the business with appropriate experience and knowledge.
- Written and oral communication skills must be required as supporting courses.

Software Industry's Requirements

- **Students:**
  - Must receive knowledge and skills training necessary to work as soon as they graduate.
- **Curriculum:**
  - Must be redesigned to guide students in the application of software technology in the business context.
- **Technology:**
  - Must be selected that will enhance the educational process and industry's needs.

Software Industry & University Collaboration



Education New Direction

- Driven by these changes, fundamental change is taking place in many education institutions.

- Many education institutions are adjusting their curriculum to address industry's needs:

1. Transition from personal learning to team learning.
2. Transition from theoretical to practical.
3. Transition from independent systems to integrated systems.
4. Transition from personal computing to network computing.

New Curriculum Guidelines

- In 2004, the IEEE (Institute of Electrical and Electronics Engineers) and ACM (Association for Computing Machinery) established a set of curriculum guidelines, known as SE 2004, aimed at better equipping undergraduate students for a career in technology.

- The SE 2004 provides guidelines for universities regarding the knowledge they should teach in undergraduate programs for Computer Sciences, Software Engineering, Computer Engineering, Information Systems.

- This guideline is now being used to improve the curriculum at many universities in the U.S., UK, Europe and Asia. Some schools have adopted it entirely, but many are only adjusting their existing curriculum to reflect some necessary changes.

Importance of Software

- The importance of software in global economy:
  - The market power of software knowledge
  - The unique characteristics of software as a building material
  - The knowledge age:
    - Assets = knowledge & skills
    - Trends: Globalization, outsourcing
    - Speed: Business transaction 7/24
    - Change: Software is adaptable to change

Why Study Software Engineering?

- According to the U.S. Bureau of Labor: "Software engineers are projected to be one of the fastest growing occupations over the 2005-15 period, with the highest starting salaries and clearly highest in demand."

- In Asia, there are a few important trends:
  - Governments are funding large-scale economic initiatives focusing on high technology, especially software.
  - Corporations need increasingly complex enterprise-wide integrated software solutions.
  - Corporations in the U.S. and western Europe are rapidly increasing the amount of work outsourced to Asia.

Why Study Software Engineering?

- According to recent studies, the current IT workforce in Asia is insufficient to meet global demands. Most Asian software workers are lacking the skills required to design and implement large-scale software systems.

- It is easy to find people with programming skills in C++, Java, or .NET, but difficult to find people with the knowledge necessary for engineering large software systems.

- At the same time, 83% of the world's largest companies are outsourcing work in the amount of 60 to 120 billion dollars annually. Most outsourced work is going to India, where the demand for software people is already outpacing the supply. Clearly, there are opportunities in southeast Asia for well-trained IT graduates.

Software Engineering Perspective

- The history of software development is one of increasing scale and complexity.
- In the beginning, a few people could hand craft small software programs. The work grew, needing teams of ten or twenty ……then a few hundred as the scale of work continued to grow, but the success was mixed.
- Today large software projects require many teams of hundreds of people, with the complexity outpacing the ability to intuitively solve problems.
- What is required is a more structured and disciplined approach to software development and management….. and the world needs: Software Engineering.