Royal University of Phnom Penh

Software Engineering

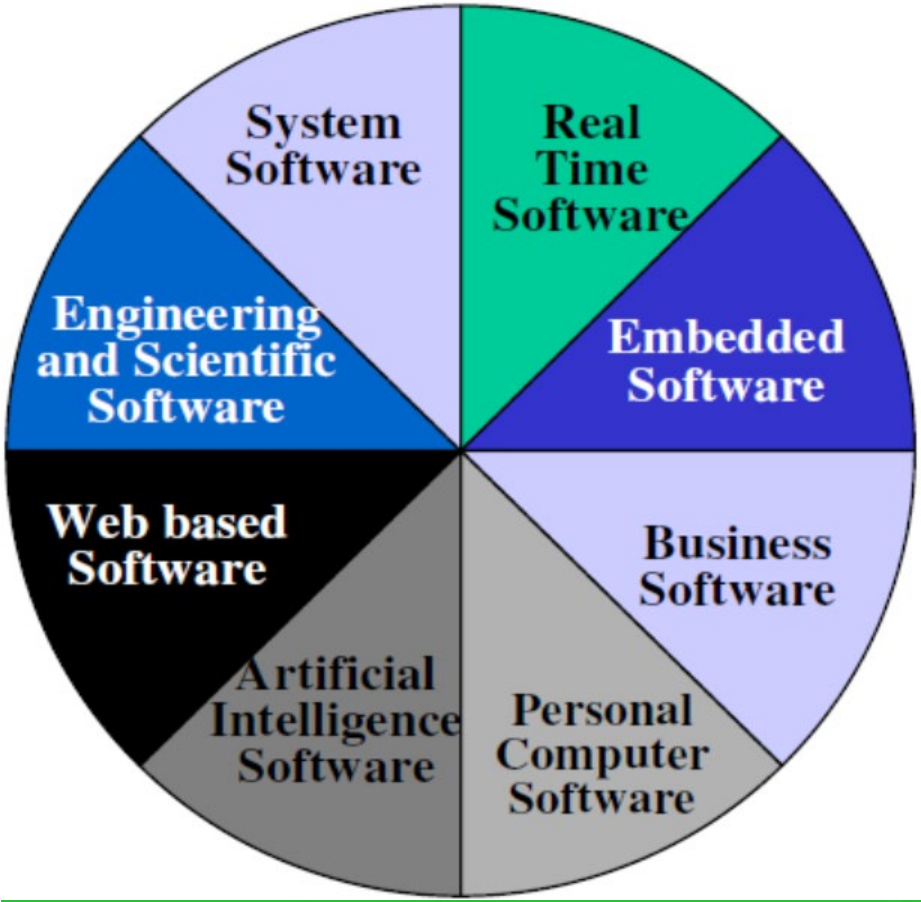
Facebook Lim Lyheng Pages ចែកចំណេះនិងបទពិសោធន៍

YouTube Lim Lyheng

Lecturer: Mr. Lim Lyheng Tel/Telegram: +855 11 855297 @LimLyheng

Chapter 1

Introduction to Software Engineering



Requirements & Grading Policy

- Attendance = **10%**
- Assignment= **15%**
- Mid-Term+ Homework + Quiz =**15%**
- Final Exam = **60%**
- **Note:** **Absent 2 time= -1% Permission 2 times=1 Absent**
- **How to submit Homework and Assignment**:
  - Return on time via my personal telegram
  - Homework must be an **individual** effort
  - Assignment will be done in group
  - Late homework/assignment will not be accepted without prior approval

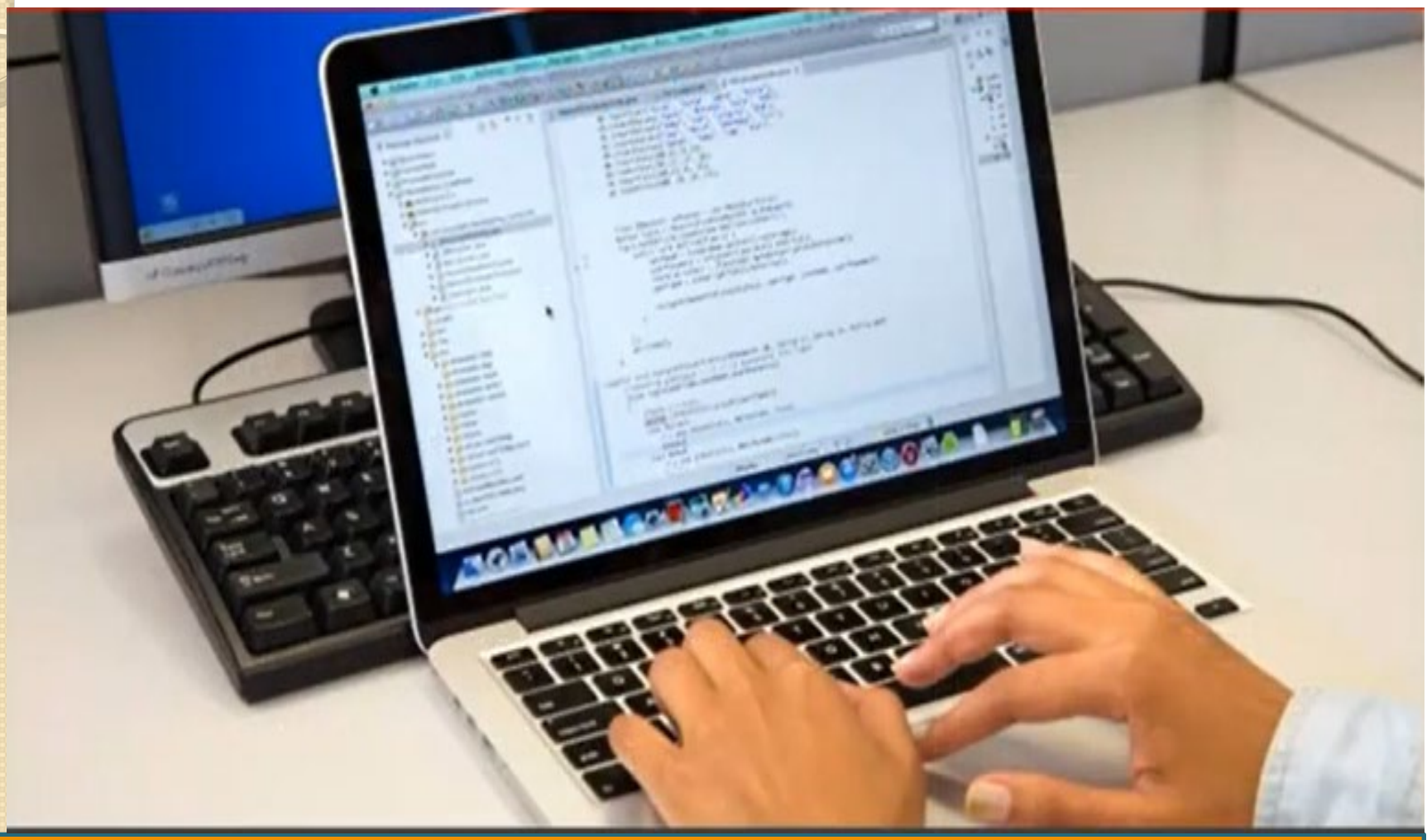
Why we need to study SE?

Software Engineering is the most well-paid Job

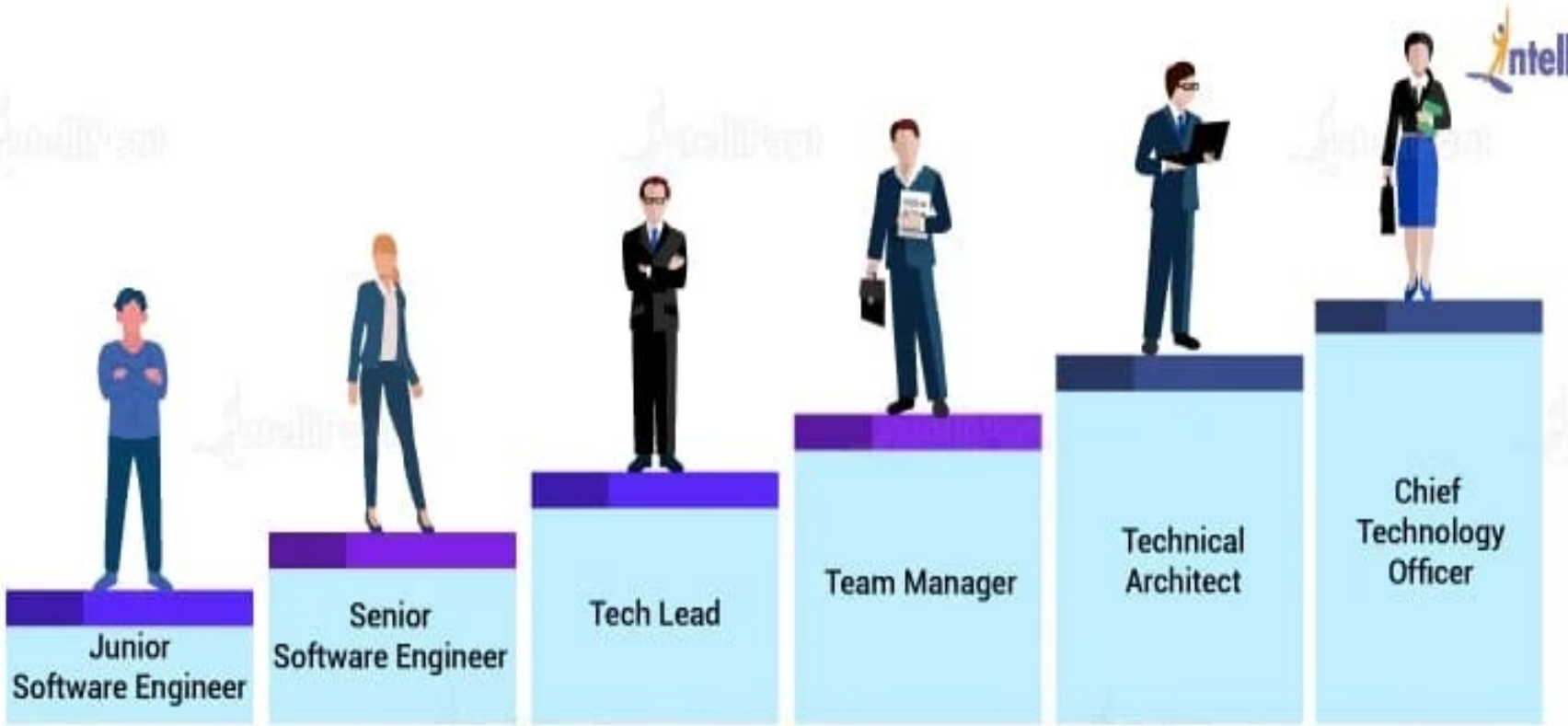
Computer Science Jobs

Excellent Job Opportunity

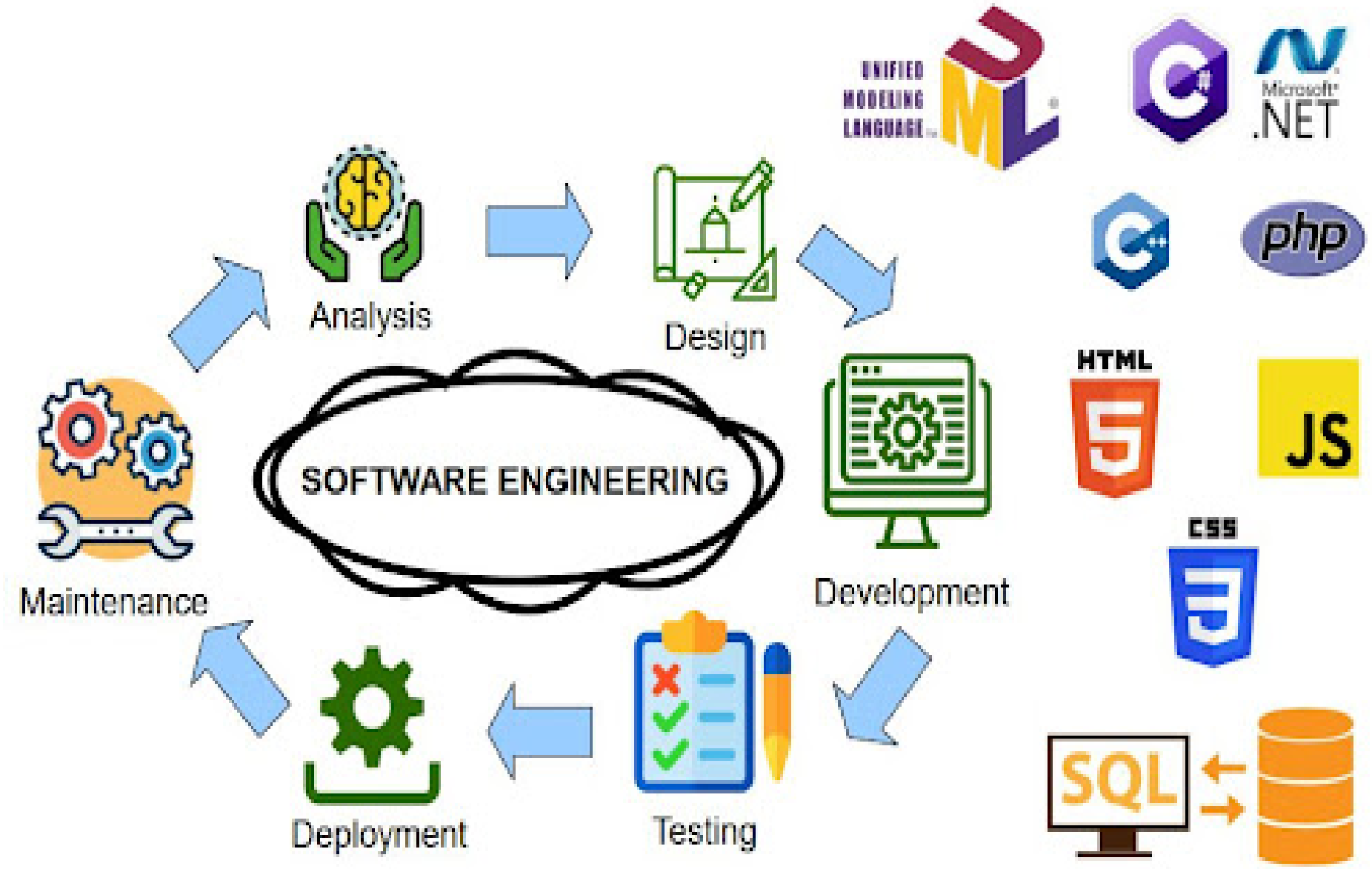
High demand Job across the Globe



Career Opportunities?



Software Engineering



A Day in the Life of a Software Engineer

Entry-Level Software Engineer

- Implement **new features**
- learn new **languages & frameworks** (React, Ruby, Python)
- Maintain **exicting code** (</CODE>)
- Pair programming
- Resolve **simple bugs / errors**
- Refactor **code**
- Practice **test driven development** (TDD)

Senior Software Engineer

- Decide which **framework & libraries** to use
- Architect API **endpoints & data models**
- **Mentor** junior engineers
- Attend sprint planning meetings with **product managers & designers**
- **Code reviews**
- Refactor **legacy code**
- Architect **scalable systems**

Top Skills for Software Engineers

1. Problem Solving

2. Attention to Detail

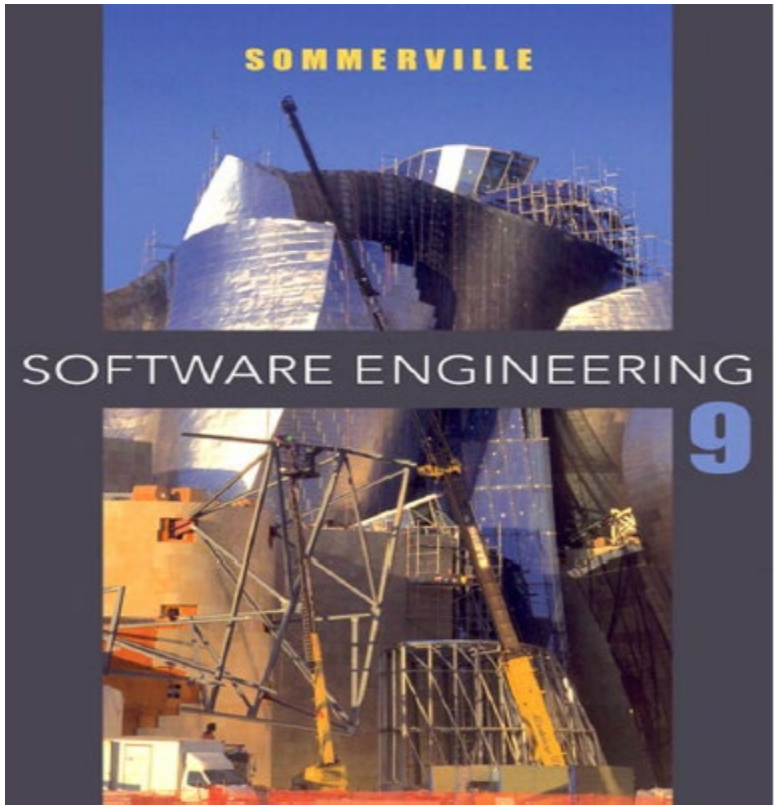
3. Clear Communication

4. Continual Learning

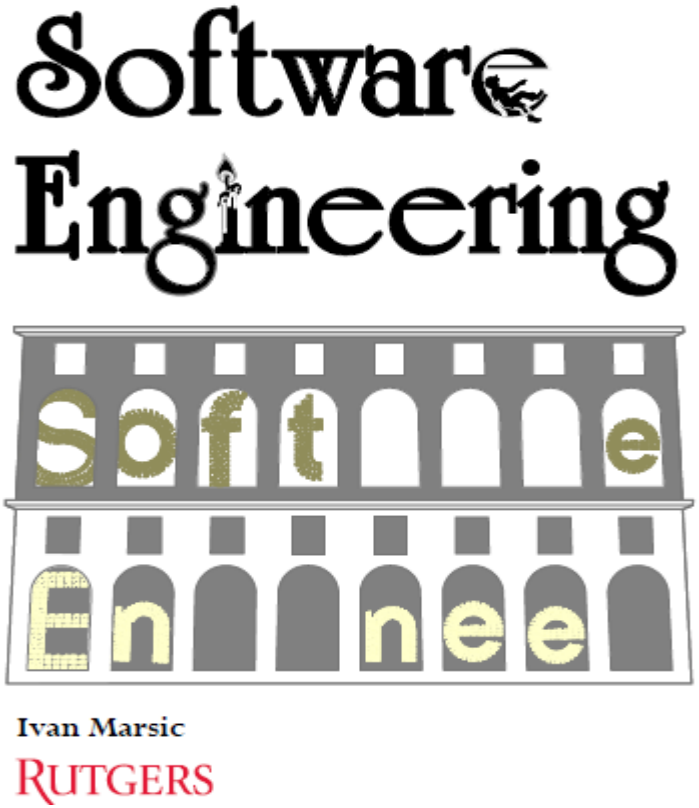
5. Teamwork

6. Empathize with End Users

Textbooks



"Software Engineering" by SimmelVill lan, 9th edition, Wiley, 2011, USA.



"Software Engineering" by Ivan Marsic, Rutger, 2012, USA.

McKinsey&Company

Software development handbook

Transforming for the digital age

Software Development January 2016

SCRUMstudy
Targeting success

A Guide to the

SCRUM BODY OF KNOWLEDGE (SBOK™ GUIDE)

2016 Edition

A Comprehensive Guide to Deliver Projects using Scrum

Key benefit of Software Engineering

- Reduces complexity
- Reduces the software cost
- Reduces time
- Handling big projects
- Reliable software
- Effectiveness

Overall Aim of the Course

We assume that you are technically proficient.

You know a good deal about computing, can program reasonably, can learn more on the job.

When you leave RUPP, you are going to work on production projects where success or failure costs hundreds of thousands and millions of dollars.

Soon you will be in charge! It may be your money!

We want you to make your mistakes now and learn from your mistakes.

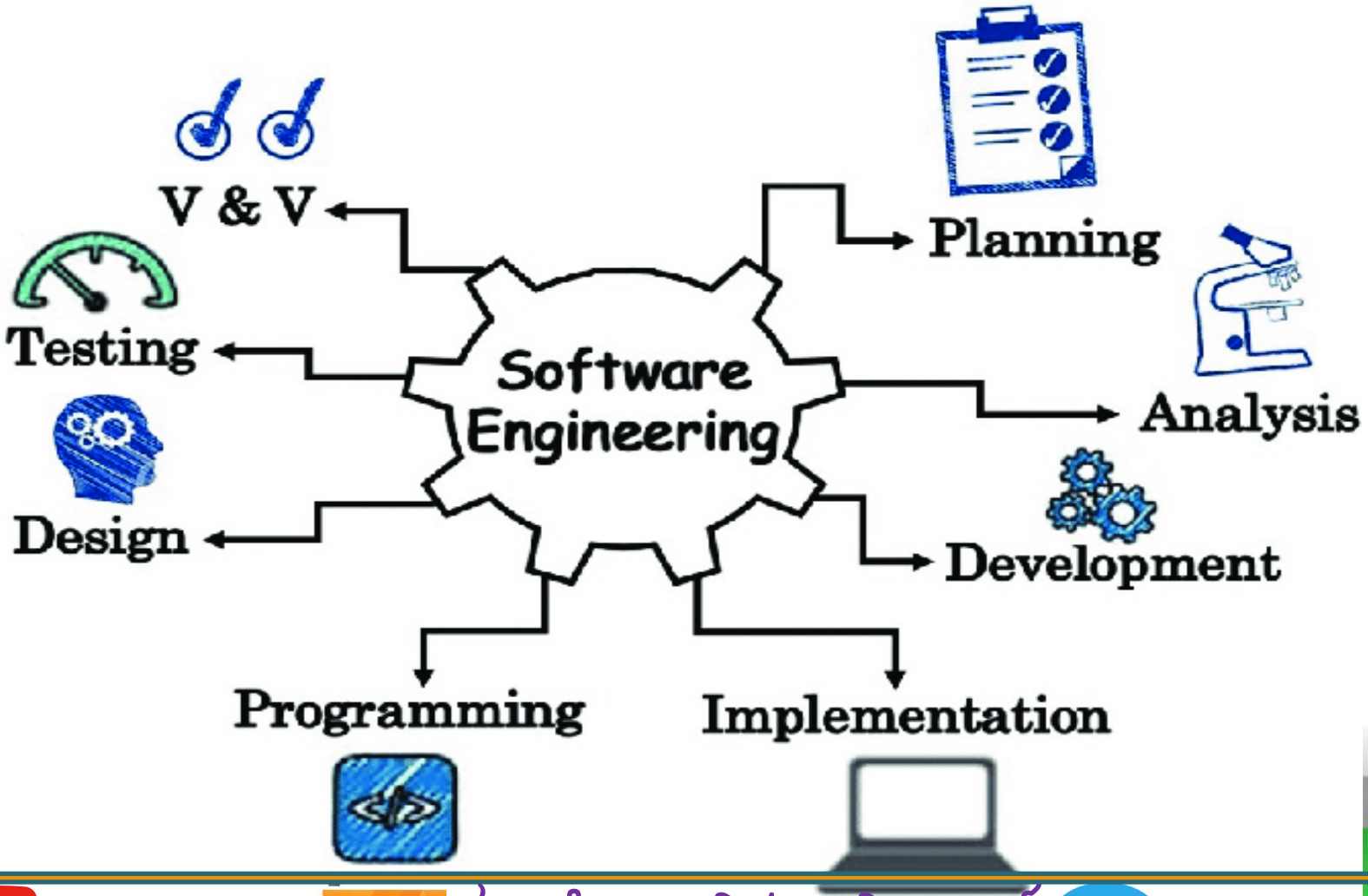
Learning Objective

- To introduce software engineering and to explain its importance
- Evaluate Software Engineering Techniques and Approaches
- To set out the answers to key questions about software engineering
- To introduce ethical and professional issues and to explain why they are of concern to software engineers

Learning Outcome?

1. Ability to develop future software solutions
2. Ability to counsel CEOs and political decision makers on investments in new software solutions based on your ability to assess user needs and technological possibilities
3. Ability to head the development and implementation of IT based on a thorough knowledge of software and its strengths and weaknesses in relation to a specific company.
4. Ability to work effectively in a team to analyze the requirements of a complex software system, and to create a design that satisfies these requirements
5. Ability to evaluate requirements for a complex software system
6. Ability to explain key aspects of models and processes for designing a complex software system

What is Software Engineering?



What is Software Engineering?

- Software engineering is an engineering branch associated with development of software product using well-defined scientific principles, methods and procedures.
- The outcome of software engineering is an efficient and reliable software product.

Software Overview

- Let us understand what Software Engineering stands for. The term is made of two words, software and engineering.
- **Software** is more than just a program code. A program is an executable code, which serves some computational purpose. Software is considered to be collection of executable programming code, associated libraries and documentations. Software, when made for a specific requirement is called **software product.**
- **Engineering** on the other hand, is all about developing products, using well-defined, scientific principles and methods. .

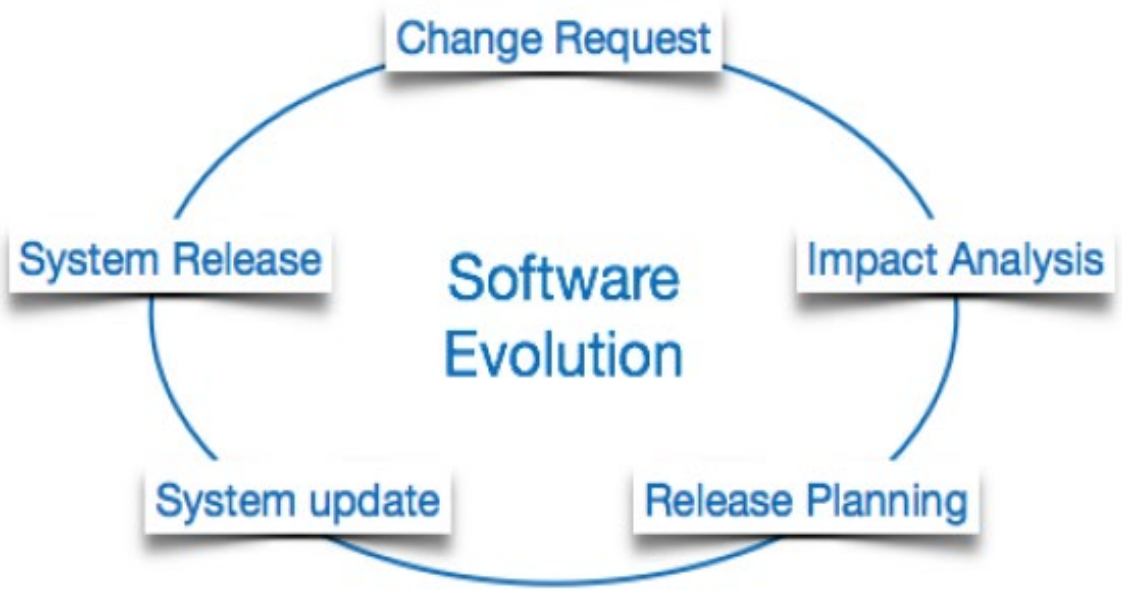
Then What is Software engineering in current situation?

Software engineering is an engineering discipline that is concerned with all aspects of software production which is

- Robust/Bug-free software
  - deliver on time
    - within budget plan
      - satisfy client's requirements …

Software Evolution

- The process of developing a software product using software engineering principles and methods is referred to as **Software Evolution.** This includes the initial development of software and its maintenance and updates, till desired software product is developed, which satisfies the expected requirements



Software Evolution

- Evolution starts from the requirement gathering process. After which developers create a prototype of the intended software and show it to the users to get their feedback at the early stage of the software product development. The users suggest changes, on which several consecutive updates and maintenance keep on changing too. This process changes to the original software, till the desired software is accomplished.
- Even after the user has the desired software in hand, the advancing technology and the changing requirements force the software product to change accordingly. Re-creating software from scratch and to go one-on-one with the requirement is not feasible. The only feasible and economical solution is to update the existing software so that it matches the latest requirements.

Software Evolution Law

Manny Lehman has given laws for software evolution. He divided the software into three different categories:

1. **Static-type (S-type) -** This is a software, which works strictly according to defined specifications and solutions. The solution and the method to achieve it, both are immediately understood before coding. The s-type software is least subjected to changes hence this is the simplest of all. For example, calculator program for mathematical computation.
2. **Practical-type (P-type) -** This is a software with a collection of procedures. This is defined by exactly what procedures can do. In this software, the specifications can be described but the solution is not obviously instant. For example, gaming software.
3. **Embedded-type (E-type) -** This software works closely as the requirement of real-world environment. This software has a high degree of evolution as there are various changes in laws, taxes etc. in the real world situations. For example, Online trading software.

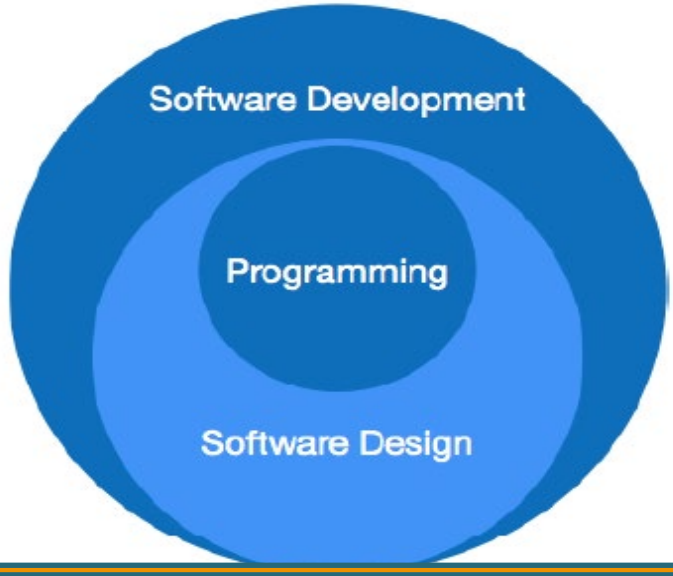
E-Type software evolution

Manny Lehman has given eight laws for E-Type software evolution -

1. **Continuing change -** An E-type software system must continue to adapt to the real world changes, else it becomes progressively less useful.
2. **Increasing complexity -** As an E-type software system evolves, its complexity tends to increase unless work is done to maintain or reduce it.
3. **Conservation of familiarity -** The familiarity with the software or the knowledge about how it was developed, why was it developed in that particular manner etc., must be retained at any cost, to implement the changes in the system.
4. **Continuing growth-** In order for an E-type system intended to resolve some business problem, its size of implementing the changes grows according to the lifestyle changes of the business.
5. **Reducing quality -** An E-type software system declines in quality unless rigorously maintained and adapted to a changing operational environment.
6. **Feedback systems-** The E-type software systems constitute multi-loop, multi-level feedback systems and must be treated as such to be successfully modified or improved.
7. **Self-regulation -** E-type system evolution processes are self-regulating with the distribution of product and process measures close to normal.
8. **Organizational stability -** The average effective global activity rate in an evolving E-type system is invariant over the lifetime of the product.

Software Paradigms

- Software paradigms refer to the methods and steps, which are taken while designing the software.
- There are many methods proposed and are implemented. But, we need to see where in the software engineering concept, these paradigms stand. These can be combined into various categories,
- Programming paradigm is a subset of Software design paradigm which is further a subset of Software development paradigm.



Software Development Paradigms

- Software Development Paradigm
  - Requirement gathering
  - Software design
  - Programming
- Software Design Paradigm
  - Design
  - Maintenance
  - Programming
- Programming Paradigm
  - Coding
  - Testing
  - Integration

Why Software engineering?

- The economies of ALL developed nations are dependent on software.
- More and more systems are software controlled
- Software engineering is concerned with theories, methods and tools for professional software development.
- Expenditure on software represents a significant fraction of GNP(Gross Nation Products) in all developed countries.

Why Software engineering?

- Software costs often dominate computer system costs. The costs of software on a PC are often greater than the hardware cost.
- Software costs more to maintain than it does to develop. For systems with a long life, maintenance costs may be several times development costs.
- Software engineering is concerned with cost-effective software development.

What are the need of Software Engineering?

The need of software engineering arises because of higher rate of change in user requirements and environment on which the software is working. Following are some of the needs stated:

1. **Large software -** It is easier to build a wall than a house or building, likewise, as the size of the software becomes large, engineering has to step to give it a scientific process.
2. **Scalability-** If the software process were not based on scientific and engineering concepts, it would be easier to re-create new software than to scale an existing one.
5. **Quality Management-** Better process of software development provides better and quality software product.
3. **Cost-** As hardware industry has shown its skills and huge manufacturing has lower down the price of computer and electronic hardware. But, cost of the software remains high if proper process is not adapted.
4. **Dynamic Nature-** Always growing and adapting nature of the software hugely depends upon the environment in which the user works. If the nature of software is always changing, new enhancements need to be done in the existing one. This is where the software engineering plays a good role.

Why apply SE to Systems?

- To predict time, effort, and cost!
- To improve software quality!
- To improve maintainability!
- To meet increasing demands!
- To lower software costs!
- To successfully build large, complex software systems!
- To facilitate group effort in developing software!

Why apply SE to Systems?

- Provide an understandable process for system development.
- Develop systems and software that are maintainable and easily changed.!
- Develop robust software and system.!
- Allow the process of creating computing based systems to be repeatable and manageable.

Characteristics of good software

- A software product can be judged by what it offers and how well it can be used.
- This software must satisfy on the following grounds:
  1. *Operational*
  2. *Transitional*
  3. *Maintenance*
- Well-engineered and crafted software is expected to have the following characteristics:

Characteristics of good software

Operational	Transitional	Maintenance
This tells us how well the software works in operations. It can be measured on: 1. Budget 2. Usability 3. Efficiency 4. Correctness 5. Functionality 6. Dependability 7. Security	This aspect is important when the software is moved from one platform to another: 1. Portability 2. Interoperability 3. Reusability 4. Adaptability	This aspect briefs about how well the software has the capabilities to maintain itself in the ever-changing environment: 1. Modularity 2. Maintainability 3. Flexibility 4. Scalability

Thoughts about Project Selection

Projects

- Target must be a production system (not research)
- Client should be one or two designated people -- client should be prepared to meet with you regularly and attend the presentations

Team

- Teams need many strengths -- organizational, technical, writing, etc.
- Consider appointing a leader to coordinate the effort

Variety of Software Products

Examples

Real time:	air traffic control
Embedded systems:	digital camera, GPS
Data processing:	telephone billing, pensions
Information systems:	web sites, digital libraries
Sensors:	weather data
System software:	operating systems, compilers
Communications:	routers, mobile telephones
Offices:	word processing, video conferences
Scientific:	simulations, weather forecasting
Graphical:	film making, design

etc., etc., etc., ...

Client

- **Client (XYZ Customer)**

The client gives requirements and provides resources and expects some product in return.

- ***Client satisfaction is the primary measurement of success in a software project.***

Question: Who is the client for your Product?

Categories of Software Product

Categories of client and software product:

- Generic (e.g., Microsoft Excel)
- Packages (e.g., Mathematica)
- Customized versions of generic packages (e.g., Cornell's payroll system)
- Bespoke (customized) (e.g., IRS internal system)
- Demonstration, prototype, research, ...

Who is the client for each category of product?

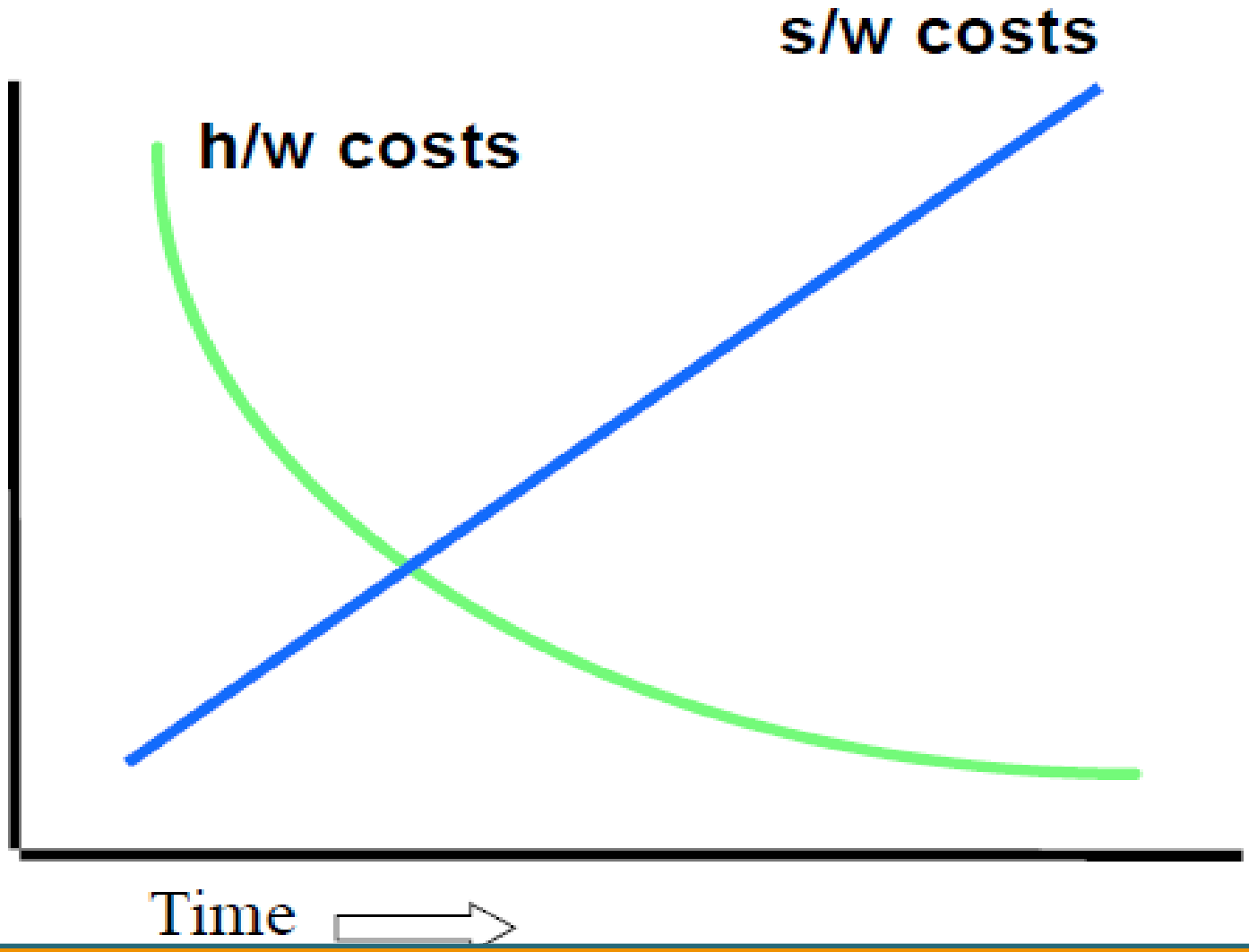
Observations

- Most software development is by **teams**
  - Effectiveness of team determines success
- Most large software projects are **built on older ones**
  - It is rare to start a new suite of programs from scratch
  - Building on the work of others is a fundamental skill of software development

Why Software is so expensive?

- Software is expensive!
- The major costs are salaries (your salaries)
- More and more demand from clients
- Every software project has a trade-off between:
  1. Functionality
  2. Resources (cost)
  3. Timeliness

Hardware Cost Vs Software Cost?



Frequently asked questions about software engineering

Question	Answer
What is software?	Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.
What are the attributes of good software?	Good software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable.
What is software engineering?	Software engineering is an engineering discipline that is concerned with all aspects of software production.
What are the fundamental software engineering activities?	Software specification, software development, software validation and software evolution.
What is the difference between software engineering and computer science?	Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.
What is the difference between software engineering and system engineering?	System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. Software engineering is part of this more general process.

Frequently asked questions about software engineering

Question	Answer
What are the key challenges facing software engineering?	Coping with increasing diversity, demands for reduced delivery times and developing trustworthy software.
What are the costs of software engineering?	Roughly 60% of software costs are development costs, 40% are testing costs. For custom software, evolution costs often exceed development costs.
What are the best software engineering techniques and methods?	While all software projects have to be professionally managed and developed, different techniques are appropriate for different types of system. For example, games should always be developed using a series of prototypes whereas safety critical control systems require a complete and analyzable specification to be developed. You can't, therefore, say that one method is better than another.
What differences has the web made to software engineering?	The web has led to the availability of software services and the possibility of developing highly distributed service-based systems. Web-based systems development has led to important advances in programming languages and software reuse.

Programmer	Software Engineer
	
A computer programmer is aware of the way to code and will have the technical skills required to create significant merchandise.	A software engineer follows a scientific method of understanding necessities, operating with stakeholders and developing an answer that fulfills their needs.

Programmer	Software Engineer
	
A programmer tends to work alone.	A software engineer is an element of a bigger team.

What are the attributes of good software?

- The software should deliver the required functionality and performance to the user and should be maintainable, dependable and acceptable.
- **Maintainability**
  - Software must evolve to meet changing needs;
- **Dependability**
  - Software must be trustworthy;
- **Efficiency**
  - Software should not make wasteful use of system resources;
- **Acceptability**
  - Software must accepted by the users for which it was designed. This means it must be understandable, usable and compatible with other systems.

Why software development is so difficult?

- Communication!
  - Between customer and developer!
  - Poor problem definition is largest cause of failed software projects!
- Within development team!
  - More people = more communication!
  - New programmers need training!
- Project characteristics!
  - Novelty!
  - Changing requirements!
    - 5 x cost during development!
    - up to 100 x cost during maintenance!
  - Hardware/software configuration
  - Security requirements
  - Real time requirements
  - Reliability requirements

Why software development is so difficult?

- Personnel characteristics
  - Ability!
  - Prior experience!
  - Communication skills!
  - Team cooperation!
  - Training!
- Facilities and resources!
  - Identification
  - Acquisition
- Management issues
  - Realistic goals!
  - Cost estimation!
  - Scheduling!
  - Resource allocation!
  - Quality assurance!
  - Version control!
  - Contracts!

Goal of Software Engineering

To produce the reliable high-quality software at low cost

Development of high quality software is difficult

1. How to solve a difficult software problem?
2. How to divide the work among hundreds of developers?
3. How long is it going to take?
4. How much is it going to cost?
5. How to make it reliable, robust, or safe-critical?
6. How do we ensure the quality of the software that we produce?!
7. How do we meet growing demand and still maintain budget control?!
8. How do we avoid disastrous time delays?!

Objective of Software Engineering

1. **Maintainability**- the ease which changes in a functional unit can be performed in order to meet prescribed requirement
2. **Correctness**-the extent to which software meets its specified requirements.
3. **Reusability**- the extent to which a module can be used in multiple applications

Objective of Software Engineering

4. **Testability**- the extent to which software facilitates both the establishment of test criteria and the evaluation of the software with respect to those criteria
5. **Reliability**- an attribute of software quality. The extent to which a program can be expected to perform its intended function, over an arbitrary time period.

Objective of Software Engineering

6. **Portability-** the ease which software can be transferred from one computer system or environment to another
7. **Adaptability-** the ease which software allows differing system constraints and user needs to be satisfied by making changes to the software.

What is Software Problem?

- Software cannot be built fast enough to keep up with !
  - H/W advances
  - Rising expectations
  - Feature explosion
- Increasing need for high reliability software!

What is Software Problem?

- Software is difficult to maintain! "aging software"
- Difficult to estimate software costs and schedules!
- Too many projects fail
  - Arianne Missile
  - Denver Airport Baggage System
  - Samsung galaxy note7

Software Crisis

- The construction of new software, which is pleasing to both user and buyer and does not contain errors, is an unexpectedly hard problem.
- It's perhaps the most difficult problem in engineering today.. Referred to as the **" Software Crisis",** it has become the longest continuing crisis in the engineering world, and it continues unabated.

What is causing Software Crisis?

Software crisis is characterized by inability to develop the desired Software Project because of such problems:

- Projects running over-budget.
- Project running over-time.
- Software is inefficient.
- Software is of low quality
- Software does not meet user/customer needs
- Project is unmanageable/Code difficult to maintain
- Standards are not used or documented
- Software Development Progress is not clear/unknown

How to Solve Software Crisis?

Software Engineering

We can successfully build large reliable systems. Software Engineering has worked

Software Engineering Characteristics

1) Software is engineered or developed, it's not manufactured in the classical sense
2) Software doesn't wear out
3) Software is complex
4) Software is intangible
5) Most software is custom-built, rather than being assembled from existing components

The Characteristics of Good Software Engineer

1. Passionate
2. Determined
3. Team Player
4. Confident
5. Up-to-date
6. Efficient Time Management
7. Coolheaded and Open Minded
8. Competitive
9. Creative
10. Strategist
11. Original
12. Industrious
13. Realist
14. Independent
15. Nerves of Steel

What are the key challenges facing software engineering?

- Heterogeneity, delivery and trust.
- **Heterogeneity**
  - Developing techniques for building software that can cope with heterogeneous platforms and execution environments;
- **Delivery**
  - Developing techniques that lead to faster delivery of software;
- **Trust**
  - Developing techniques that demonstrate that software can be trusted by its users.

Professional and ethical responsibility

- 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.
- Ethical behaviour is more than simply upholding the law.

Issues of professional responsibility

- **Confidentiality**
  - Engineers should normally respect the confidentiality of their employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.
- **Competence**
  - Engineers should not misrepresent their level of competence. They should not knowingly accept work which is outwith their competence.

Ethics vs Law

- **Ethics** are personal code of behaviour
- **Law** is minimum standard imposed by society
- **Law** represents will of majority (Violation punishable by government)
- **Ethics** are suggested, not mandated (Violation punishable by government)

Issues of professional responsibility

- **Intellectual property rights**
  - Engineers should be aware of local laws governing the use of intellectual property such as patents, copyright, etc. They should be careful to ensure that the intellectual property of employers and clients is protected.
- **Computer misuse**
  - Software engineers should not use their technical skills to misuse other people's computers. Computer misuse ranges from relatively trivial (game playing on an employer's machine, say) to extremely serious (dissemination of viruses).

Code of ethics- Principle

1. **Public:** Software Engineers shall act consistently with the public interest
2. **Client and Employer:** Software Engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
3. **Product:** software Engineers shall ensure that their products and related modifications meet the highest professional standards possible

Code of ethics- Principle

4. **Judgement:** Software Engineers shall maintain integrity and independence if their professional judgement
5. **Management:** Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance
6. **Profession:** Software Engineers shall advance the integrity and reputation of the profession consistent with the public interest.

Code of ethics- Principle

7. **Colleagues:** Software Engineers shall be fair to and supportive of their colleagues
8. **Self-Learning:** Software Engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

Ethical Dilemmas

1. Disagreement in principle with the policies of senior management.
2. Your employer acts in an unethical way and release a safety-critical system without finishing the testing of the system
3. Participation in the development of military weapons systems or nuclear systems.

Key Points

- Software engineering is an engineering discipline that is concerned with all aspects of software production.
- Software products consist of developed programs and associated documentation. Essential product attributes are maintainability, dependability, efficiency and usability.
- The software process consists of activities that involved in developing software products. Basic activities are software specification, development, validation and evolution.

Key Points

- Methods are organised ways of producing software. They include suggestions for the process to be followed, the notations to be used, rules governing the system descriptions which are produced and design guidelines.
- Software engineers have responsibility to the engineering profession and society. They should not simply be concerned with technical issues.
- Professional societies publish codes of conduct which set out the standards of behaviour expected of their members.

Summary

- Software engineering developed out of necessity to handle large software projects that cannot be handled by few individuals using ad hoc methods.
- Goal is to develop software systems that:
  - Satisfy technical requirements, user needs, and acquirer expectations.
  - Are developed on time and on budget
  - Are easy to modify and maintain
  - Are developed with pride and personal satisfaction
  - Fulfils all ethical and legal considerations
- Good project management and software engineering process are required.

Royal University of Phnom Penh

Pages

ចែកចំណេះនិងបទពិសោធន៍

YouTube Lim Lyheng

End of Chapter 1

Thank You