OVERVIEW OF SOFTWARE ENGINEERING
(Tổng quan về Công nghệ Phần mềm)

PhD. Nguyen Thi Hanh

NỘI DUNG

1. Software
2. Software Engineering
3. Software Development Challenges
4. Software Engineer
5. Software Engineering Ethics

1. Software

- What is Software?
- Wear vs. Deterioration
- Software Products
- Application types
- Essential Attributes of Good Software

3

What is Software?

- 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) documentation that describes the operation and use of the programs.

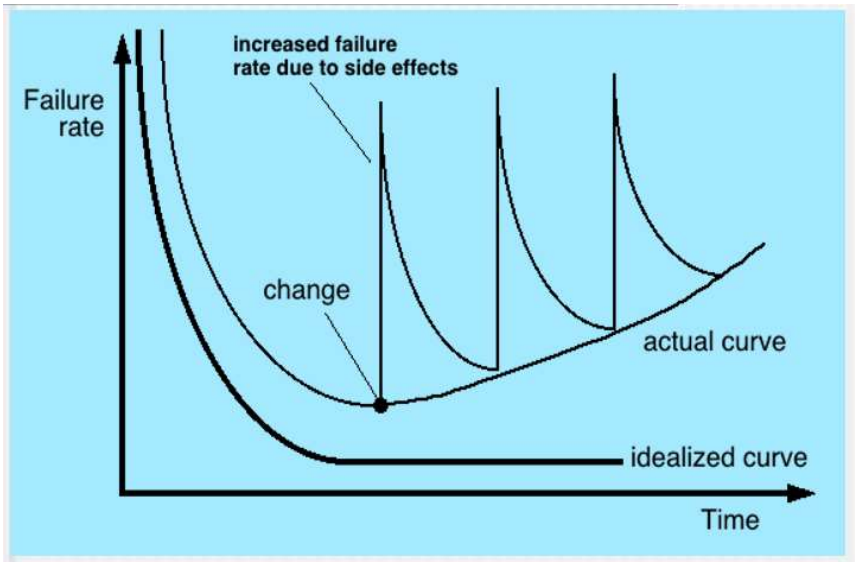
4

What is Software?

- **Software's Components**

5

Wear vs. Deterioration



Software Products

- Generic products
  - Stand-alone systems that are marketed and sold to any customer who wishes to buy them.
  - Examples – PC software such as graphics programs, project management tools; CAD software; software for specific markets such as appointments systems for dentists.
- Customized products
  - Software that is commissioned by a specific customer to meet their own needs.
  - Examples – embedded control systems, air traffic control software, traffic monitoring systems.

7

Application types

- Stand-alone applications
  - These are application systems that run on a local computer, such as a PC. They include all necessary functionality and do not need to be connected to a network.
- Interactive transaction-based applications
  - Applications that execute on a remote computer and are accessed by users from their own PCs or terminals. These include web applications such as e-commerce applications.
- Embedded control systems
  - These are software control systems that control and manage hardware devices. Numerically, there are probably more embedded systems than any other type of system

8

Application types

- Batch processing systems
  - These are business systems that are designed to process data in large batches. They process large numbers of individual inputs to create corresponding outputs.
- Entertainment systems
  - These are systems that are primarily for personal use and which are intended to entertain the user.
- Systems for modeling and simulation
  - These are systems that are developed by scientists and engineers to model physical processes or situations, which include many, separate, interacting objects.

9

Application types

- Data collection systems
  - These are systems that collect data from their environment using a set of sensors and send that data to other systems for processing.
- Systems of systems
  - These are systems that are composed of a number of other software systems.

10

Essential Attributes of Good Software

Product characteristic	Description
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
Dependability and security	Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.
Acceptability	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.

11

2. Software Engineering

- Software Engineering
- Importance of software engineering
- History of Coding and Software Engineering
- Software process activities
- General issues that affect most software
- A Layered Technology
- Software Development Challenges

12

Software Engineering

- Some realities:
  - a concerted effort should be made to understand the problem before a software solution is developed
  - design becomes a pivotal activity
  - software should exhibit high quality
  - software should be maintainable
- Software engineering is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use

13

Software Engineering

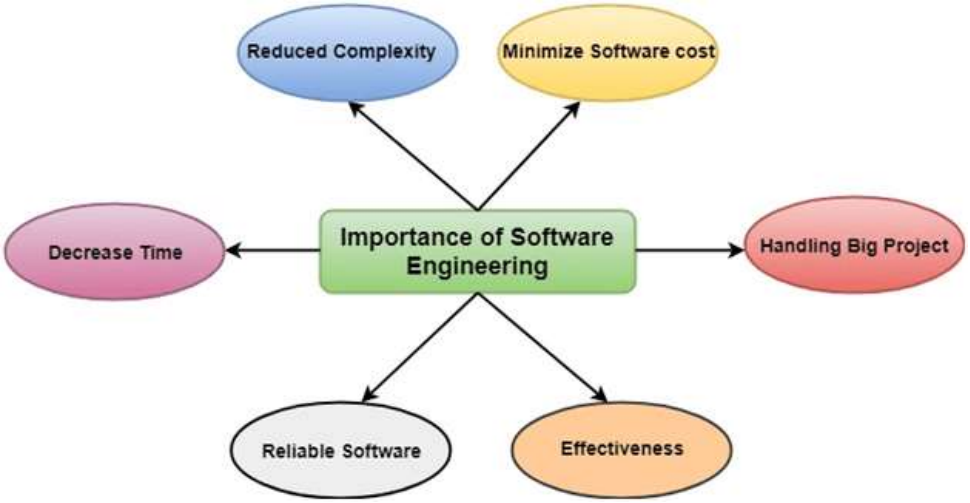
- The seminal definition:

  [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:

  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).

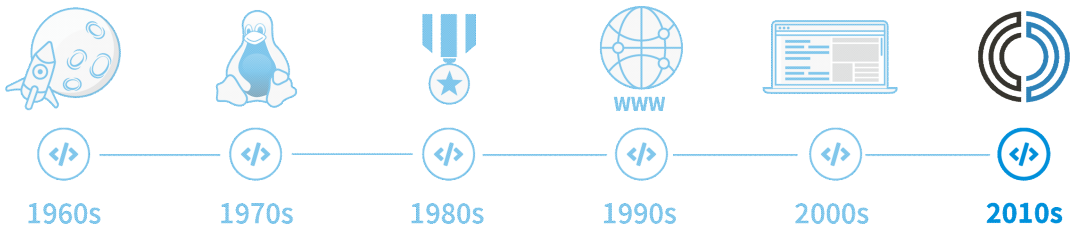
14

Importance of Software Engineering



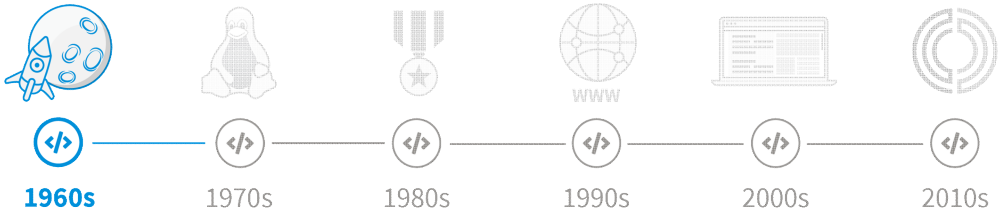
15

History of Coding and Software Engineering



16

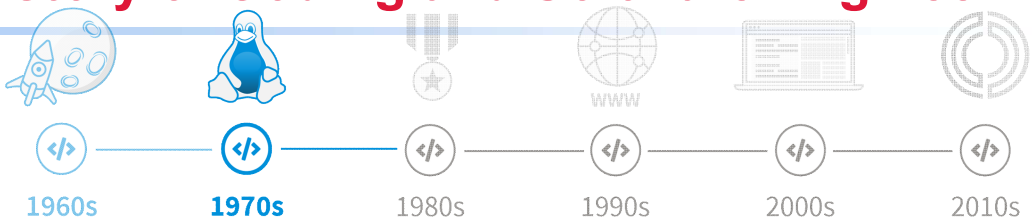
History of Coding and Software Engineering



- 1960s - were a time when software engineering was accepted as its own form of engineering but was also a time of trouble for the development of software. The software side struggled to keep pace with the hardware, which created problems for the field.

17

History of Coding and Software Engineering

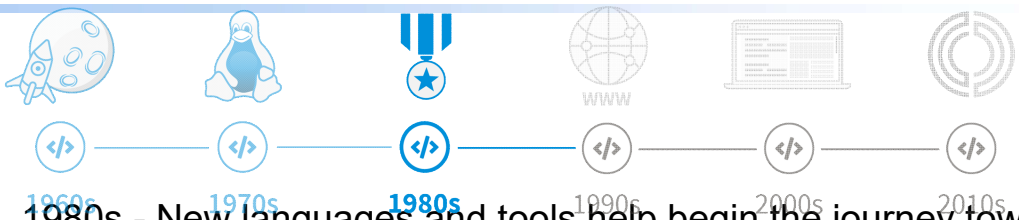


1970s - were a time when software engineering began its rise as new ideas, languages, and hardware were introduced.

- 1970 – Pascal Programming Language is introduced
- 1972 – C Programming Language would grow to become one of the most popular programming languages; the **Unix operating system** is recognized as one of the most important people in software technology
- **1975** – **The first PCs** begin to make their debut.
- **1979** – Seattle University begins offering a master's degree in computer engineering.

18

History of Coding and Software Engineering

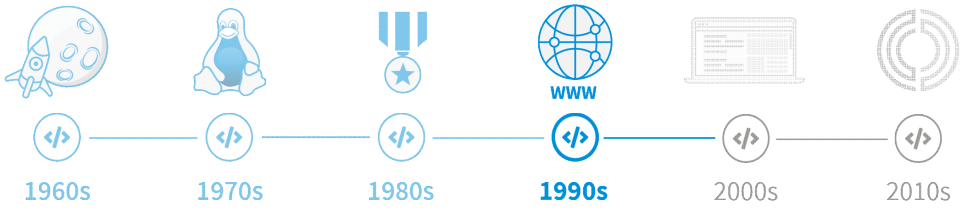


1980s - New languages and tools help begin the journey toward better engineering and the move toward object-oriented programming begins.

- 1980 – The Ada programming language
- 1982 – The first CASE tools - CASE tools
- **1985** – The C++ Programming Language is released, which has functional, generic, object-oriented, and procedural features
- **1989** – Companies begin to offer access to the internet. It's used mainly by scientists and the military

19

History of Coding and Software Engineering



1990s - object-oriented programming began to grow in popularity, the Internet made its debut, and a new approach to development was introduced.

- 1990 – WorldWideWeb, the first web browser, HTTP, HTML
- 1990 – use of the term "big data" begins to grow.
- 1991 – The Python programming language makes its debut

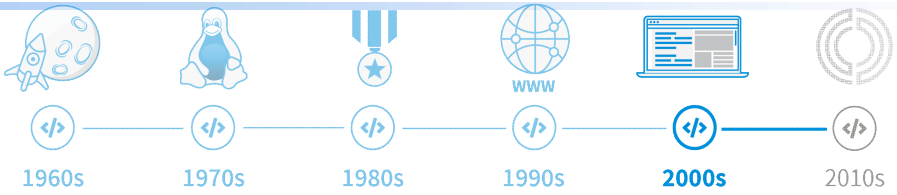
20

History of Coding and Software Engineering

- 1995 – The Java programming language, developed by James Gosling, is released, “Write Once, Run Anywhere”
- 1995 – JavaScript, a just-in-time, object-oriented programming language is introduced.
- 1996 – Rochester Institute of Technology - bachelor’s degree program in software engineering.
- 1998 – U.S. Naval Postgraduate School - doctorate program in software engineering.
- 1999 – Kent Beck - extreme programming, a type of agile software development that is designed to respond to the changing requirements of the user

21

History of Coding and Software Engineering

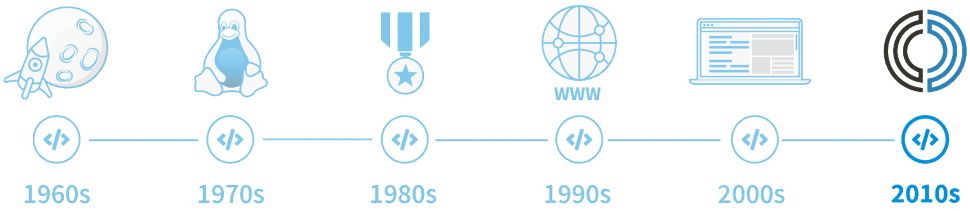


2000s - The bigger focus in the 2000s was on methodology as developers looked to make the process more responsive to customer needs, more profitable, and easier to create

- 2001 – The Manifesto for Agile Software Development is published.
- 2001 – Scrum, an agile process that uses iterative and incremental framework for complex software development, is introduced
- 2004 – The SE Body of Knowledge is introduced. The collaborative work addresses numerous software issues, including design, construction, maintenance, and more.

22

History of Coding and Software Engineering



2010s - While providing continuous improvements to languages and methods, the focus shifts again to address the need for software engineers with a new style of learning made to enhance traditional software engineering education

- **2010 –** Cloud computing begins its rise - software-as-a-service
- **2011 –** Coding bootcamps begin to pop up.

23

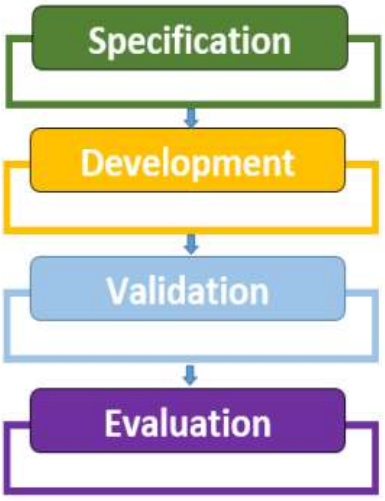
History of Coding and Software Engineering

- **2012 –** Tony Phillips starts **coding bootcamp** Hack Reactor with his brother Marcus Phillips and Shawn Drost. Their first class consisted of 16 students, all of whom found jobs.
- **2014 –** Hack Reactor launches the nation's first **online coding bootcamp**. Students from more than 20 countries use online bootcamps. Coding bootcamps at San Quentin correctional facility and developing the curriculum for a school in Kenya.
- **2015 – Hack Reactor**, as the school expands across America and creates Telegraph Academy to educate underrepresented minorities in software engineering.
- **2018 –** Hack Reactor is acquired by **Galvanize**, adding valuable product offerings, additional campuses, and services.

24

Software Process Activities

- Software specification, where customers and engineers define the software that is to be produced and the constraints on its operation.
- Software development, where the software is designed and programmed.
- Software validation, where the software is checked to ensure that it is what the customer requires.
- Software evolution, where the software is modified to reflect changing customer and market requirements.

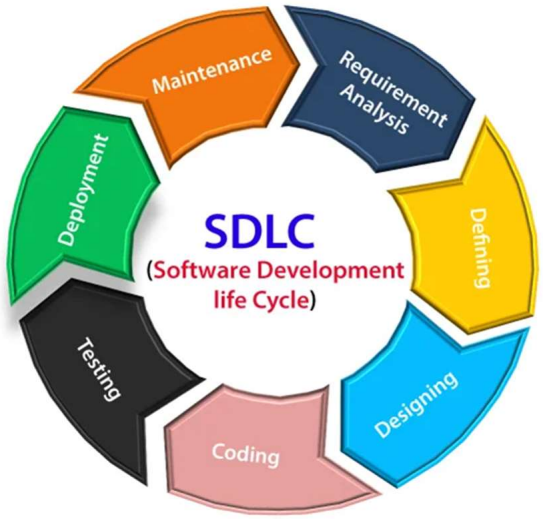


25

A Layered Technology

26

Software Development Life Cycle (SDLC)



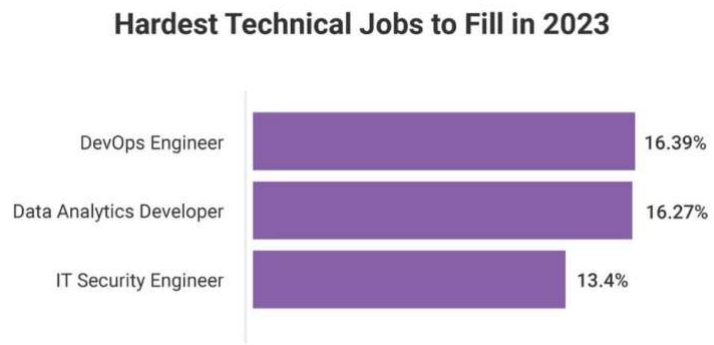
3. Software Development Challenges

- The new Reveal survey of software developers and IT professionals.
- These challenges include:
  - The shortage of developers
  - Finding time to build apps
  - An inability to keep pace with innovations in developer tools
  - Difficulty with third-party integrations
  - Workload and Project Management
  - Security threats
  - Increased client expectations

28

Developer Shortage

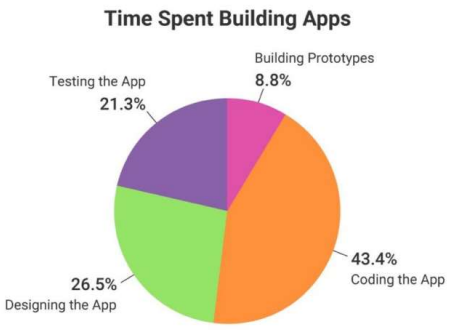
- Biggest challenge is recruiting qualified talent



29

Time Spent Building Apps

- Reveal's survey found that nearly half (43.4%) of a developer's time is spent coding an app.
- The best ways to reduce the coding time is to incorporate low-code/no-code development platforms that help development teams create applications with visual tools that require little to no up-front coding. n



30

Inability To Keep Pace With Innovations in Developer Tools

- New methodologies, programming languages, tools and technologies demand a developer's attention.
- But adopting a new language/technology/tool as often as those innovations are introduced to the software development space poses a challenge.
  - If you jump on it too soon, you may quickly discover that the innovation doesn't meet your expectations, and its failure could impede your projects.
  - if you wait too long, you might find yourself falling behind your competitors who have already become proficient with new innovations.

31

Difficulty with Third-Party Integrations

- Third-party API integrations are often necessary to make your product more beneficial and appealing to your users.
- Integrations have enormous value as they save on in-house development time and money, but you need to ensure that the API works as expected and won't impact the existing functionalities of your product.
- You also need to determine if the integration will create vulnerabilities and expose your product and users to security threats.

32

Security Threats

- API abuse, such as data breaches and losses, can seriously damage a company's reputation, not to mention the harm that can be done to the customers.
- Data breach methods are becoming more sophisticated, which means poor integration is very risky. Safely maintaining a connection with another system requires control and constant enhancement.
- You should evaluate the third-party application you want to integrate into your product. Make sure that the API is stable, scalable, and secure.

33

Workload and Project Management

- Many people are continuing to work remotely or in a hybrid environment for the long term. The difficulties they face managing workloads and projects can be overcome with a digital workplace productivity tool.
- These platforms are an integral part of successful digital transformation as they provide distributed teams with the tools to complete their tasks by managing workload and projects efficiently and effectively. Workplace productivity tools integrating project and content management, chat, and even data analytics in one software platform can increase productivity, enhance communication, foster innovation and drive business growth.

34

Workload and Project Management

- The software development challenges survey found that more than half (54.4%) of developers and IT participants want to use one tool where everyone can collaborate and resolve issues. Another 47.5% want to automate workflows and processes, and 43.7% prefer eliminating manual file sharing.

35

Increased Client Expectations

- Clients are directly impacted by the progress and success of a project, which is why developers and their product managers must ensure that their expectations are met.
- A quarter of the Reveal survey respondents (26.2%) have trouble managing their workload, and 26% feel client expectations are too high.
- In this high-pressure environment, developers must find ways to get the work done while maintaining client satisfaction.
- Workplace productivity tools that allow transparency, visibility and collaboration between developers and clients align everyone on progress, deadlines, and budget.

36

4. Software Engineer

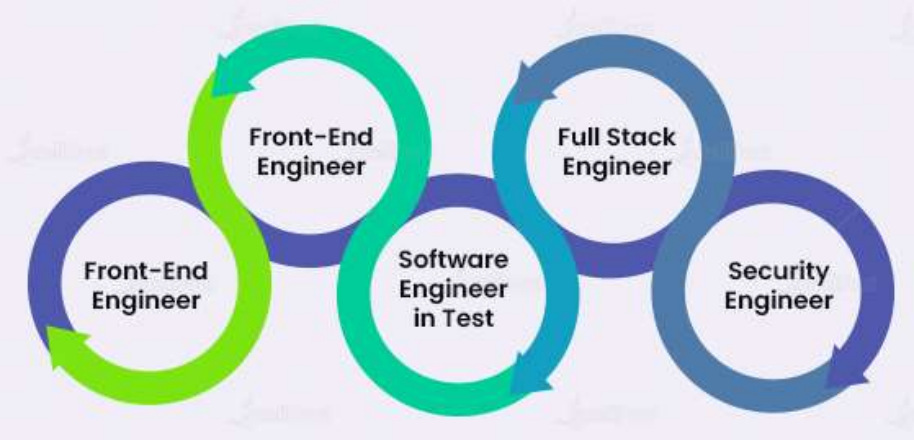
- Responsibilities of Software Engineers
- Types of Software Engineers
- Tech Skills Required by Software Engineers
- Soft Skills Required by Software Engineers
- Salary according to job position
- Title positions

37

Responsibilities of Software Engineers

38

Types of Software Engineers



39

Tech Skills Required by Software Engineers

40

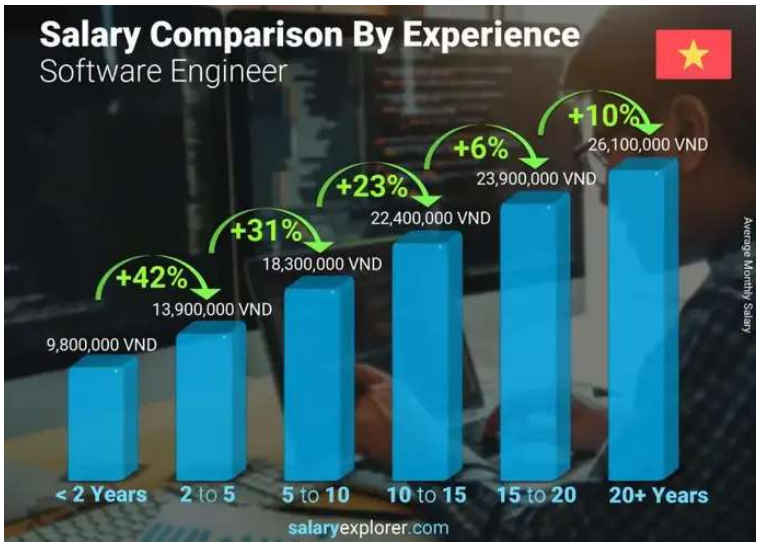
Soft Skills Required by Software Engineers

41

SE Pay Scale and Salaries in Vietnam

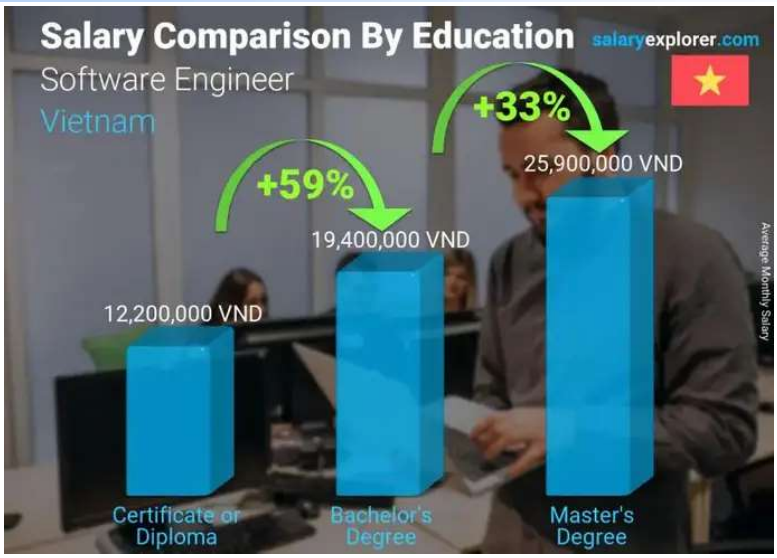
42

Salary Comparison by Years of Experience



43

Salary Comparison By Education / SE



44

Salary and Compensation Comparison By Gender / SE

45

Average Annual Salary Increment Percentage / Software Engineer

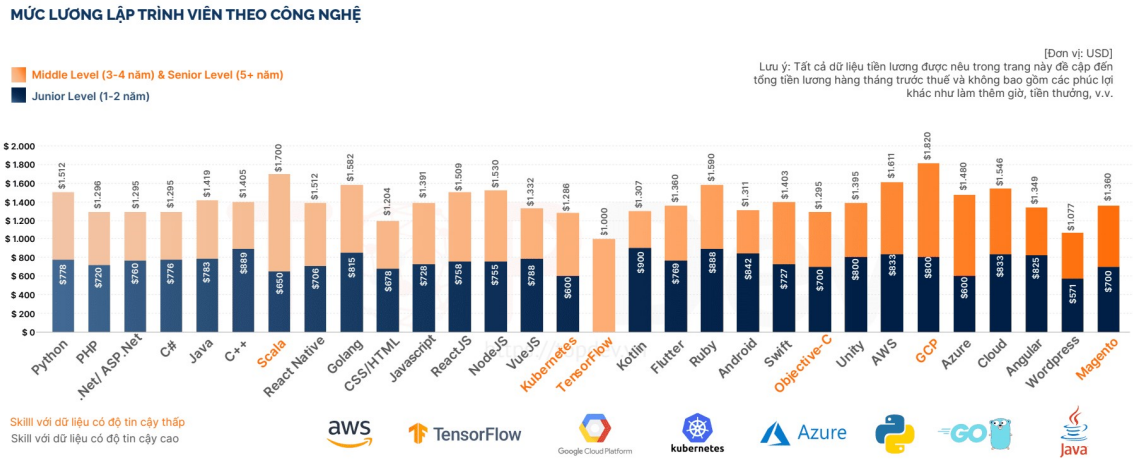
46

SE Average Salary in Vietnam 2023

- VietNam IT Market Report 2023 của TopDev

47

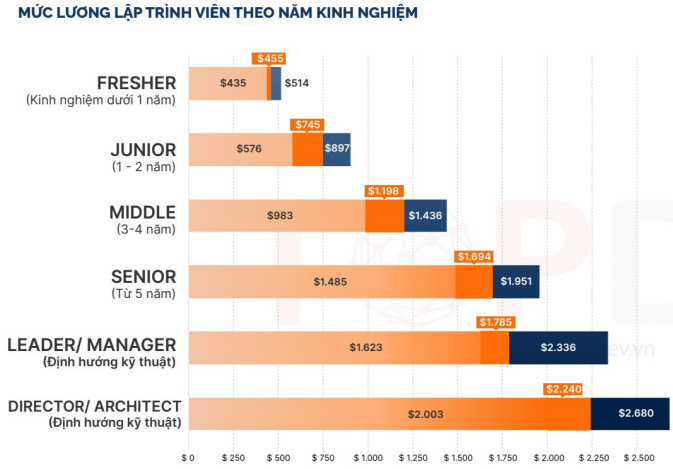
Salary According to Technology



48

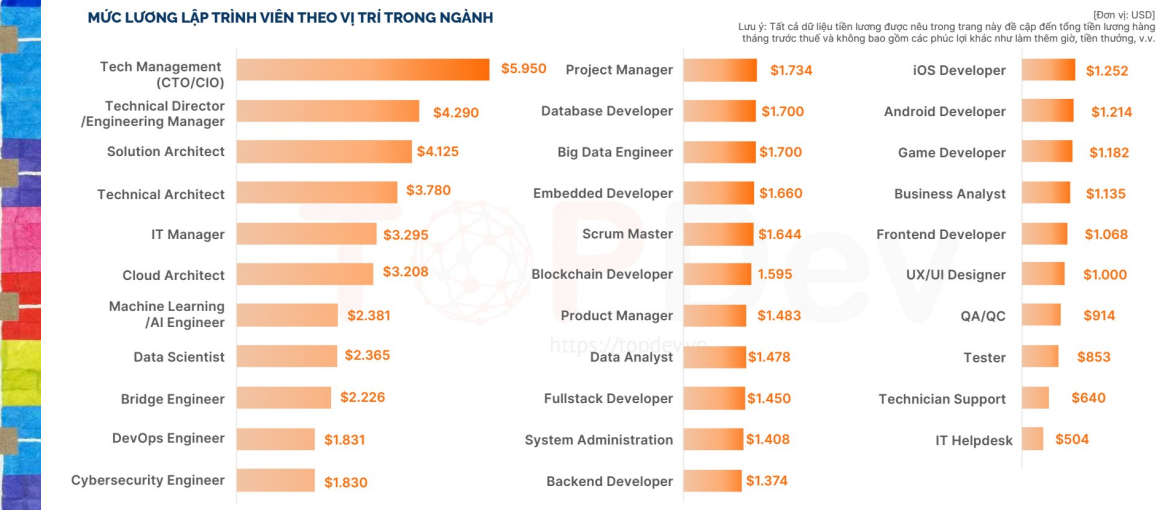
Salary based on Years of Experience

- **Years of Experience**



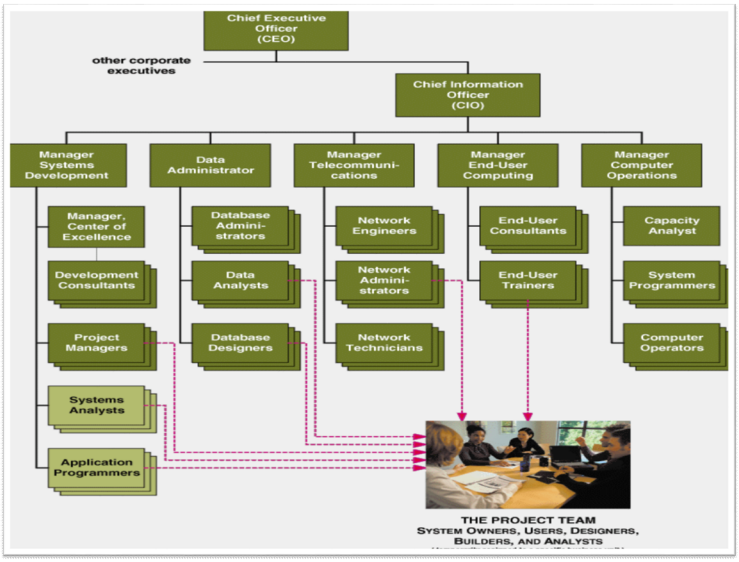
49

Salary According to Job Position



50

Job Titles in IT Field



51

Software Engineering Ethics

- 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** but involves following a **set of principles that are morally correct**.

52

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.

53

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).

54

ACM/IEEE Code of Ethics

- The professional societies in the US have cooperated to produce a code of ethical practice.
- Members of these organisations sign up to the code of practice when they join.
- The Code contains eight Principles related to the behaviour of and decisions made by professional software engineers, including practitioners, educators, managers, supervisors and policy makers, as well as trainees and students of the profession.

55

Laws related to the IT Field

- Luật công nghệ thông tin
  - Thông qua: 29/06/2006; Hiệu lực: 01/01/2007
- Luật sở hữu trí tuệ
  - Thông qua: 29/11/2005; Hiệu lực: 01/07/2006
- Luật giao dịch thương mại
  - Thông qua: 26/06/2023; Hiệu lực: 01/07/2024
- Luật an ninh mạng
  - Thông qua: 12/06/2018; Hiệu lực: 01/01/2019

56

Questions – Exercises

- What is software?
- What are the attributes of good software?
- What is software engineering?
- What are the fundamental software engineering activities?
- What is the difference between software engineering and computer science?
- What is the difference between software engineering and system engineering?
- What are the key challenges facing software engineering?
- What are the costs of software engineering?

57

Questions – Exercises

- What is the most important difference between generic software product development and custom software development? What might this mean in practice for users of generic software products?
- What are the four important attributes that all professional software should have? Suggest four other attributes that may sometimes be significant.
- Explain why there are fundamental ideas of software engineering that apply to all types of software systems.
- Explain how the universal use of the Web has changed software systems.

58