Chapter 1- Introduction

CMPG224 - Software Engineering

15 – 07 - 2024

Acknowledgement

Ian Sommerville,

Software Engineering 10th Edition

Note: These are a slightly modified version of Ch1 slides available from the author's site http://iansommerville.com/software-engineering-book/

Topics covered

- Professional software development
  - Software engineering and its importance
  - Different types of software system and software engineering techniques
- Software engineering ethics
  - Ethical and professional issues that are important for software engineers

Introduction

- Software has become more critical, pervasive and the economies of all developed nations are dependent on software.
  - expenditure on software represents a significant fraction of GNP in all developed countries
- More and more systems are software controlled
  - Including entertainment, electrical products, financial systems, communication, healthcare, industrial manufacturing & distribution, etc
- Software costs often dominate computer system costs
- Therefore, Software Engineering (SE) is concerned with cost-effective software development
  - SE involves the theories, methods and tools for professional software development.

What is Software engineering?

- SE 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.
- This definition has two parts:
  - Engineering discipline
    - Using appropriate theories and methods to solve problems bearing in mind organizational and financial constraints.
  - All aspects of software production
    - Not just technical process of development, but also project management and the development of tools, methods etc. to support software production.

[Ian Sommerville]

What is Software Engineering?

- SE is a systematic and disciplined approach to software development that aims to create a high-quality, reliable, efficient and maintainable software solutions that meet specific needs
- It involves designing, developing, testing, deploying, and maintaining a software
  - Creating fault-free software:
    - that satisfies the customer's needs
    - developed within a given budget, and
    - delivered on or before a given deadline
- SE involves the principles, tools, and techniques used to develop professional software
  - Key aspects of SE
    - Systematic approach
    - Use of methodologies and tools
    - Focus on quality and efficiency

Importance of software engineering

- Software plays a crucial role in modern society, powering various aspects of our daily lives. Thus, we need to create high-quality software system economically and quickly that is:
  - Reliable - ensures that software performs consistently under specified conditions without errors or failures.
  - Efficient – use resource effectively, including time, memory, and processing power.
  - User needs - ensures that software meets the requirements and expectations of users.
  - Scalable - allows software to handle increasing amounts of work or to be easily expanded.
  - Maintainable - facilitates easy updates and modifications to the software.
  - Secure - protects software from vulnerabilities and threats.

- Building a large complex software product is hard and building it without discipline is crazy
  - Usually cheaper, in the long run, to use SE methods and techniques for software systems rather than just write the programs as if it was a personal programming project.
  - Failure to use SE method leads to higher costs for testing, quality assurance, and long-term and expensive maintenance

History of Software Engineering?

- Software Engineering term was first coined in the 1960s in first NATO SE conference seeking for solution to then "Software Crisis".
  - Programming was done using machine code and assembly language

- Software crisis was a period in SD history when the demand for software was significantly > the ability to produce it reliably and efficiently
  - SD projects became increasingly complex, leading to:
    - budgets not being met,
    - missing deadlines,
    - producing unreliable software and
    - several software projects canceled due to the inability to meet initial requirements.

- Software crisis acted as a catalyst for the emergence of SE as a discipline.
  - Highlighted the need for better software development practices.

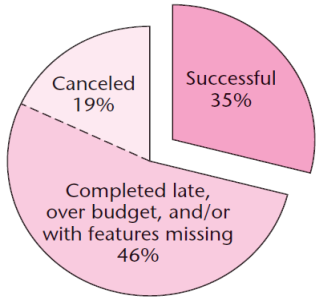
History of Software Engineering?

- Milestones
  - Development of Programming Languages:
    - 1950s-1960s: Introduction of high-level programming languages (e.g., FORTRAN, COBOL).
    - 1970s: Development of structured programming languages (e.g., C, Pascal).
    - 1980s-1990s: Emergence of object-oriented languages (e.g., C++, Java).
  - Methodologies:
    - 1970s: Waterfall model was introduced emphasizing a linear and sequential approach.
    - 1990s: Agile methodologies emerged, promoting iterative development and collaboration.
    - 2000s: Adoption of DevOps practices, integrating development and operations for continuous delivery.
  - Tools:
    - 1980s: Development of IDEs to streamline coding.
    - 2000s: Version control systems emerged (e.g., Git) to manage code changes.
    - Continuous evolution of tools for testing, deployment, and project management.
    - Etc.

Success Rate

- Post-software crisis era
  - Traditional models like the waterfall model proved inflexible and prone to failure
  - Iterative, incremental, and reuse-oriented lifecycles emerged as more effective alternatives
- Lifecycles and success:
  - Lifecycle models significantly impact project success, budget, and product quality.
    - A study by Standish Group in the late 2004 , only 28 % of software projects were considered success, 51 % were considered 'seriously late, over budget and lacking expected features.' while 18 % were cancelled outright [Hayes, 2004].
    - Standish Group in 2006 also analyzed SD projects as follows: [Rubenstein. 2007].

FIGURE 1.1
The outcomes of over 9,000 development projects completed in 2006 [Rubenstein, 2007].

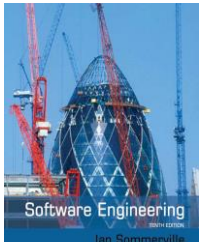


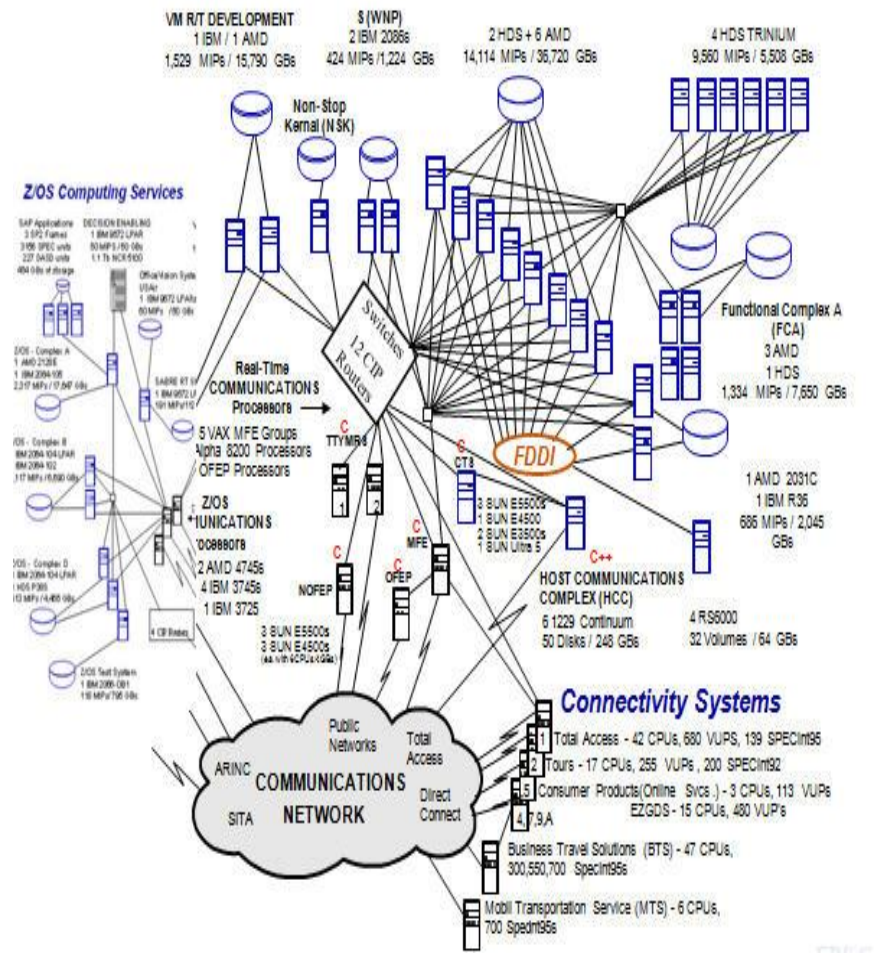
  - No universal model suits all software products equally well.
  - Customization and adaptation are essential based on project scope, size, and team expertise
- While SE has made significant advances post-crisis, challenges persist, and choosing the right lifecycle model remains fundamental for successful SD.

Software projects failure

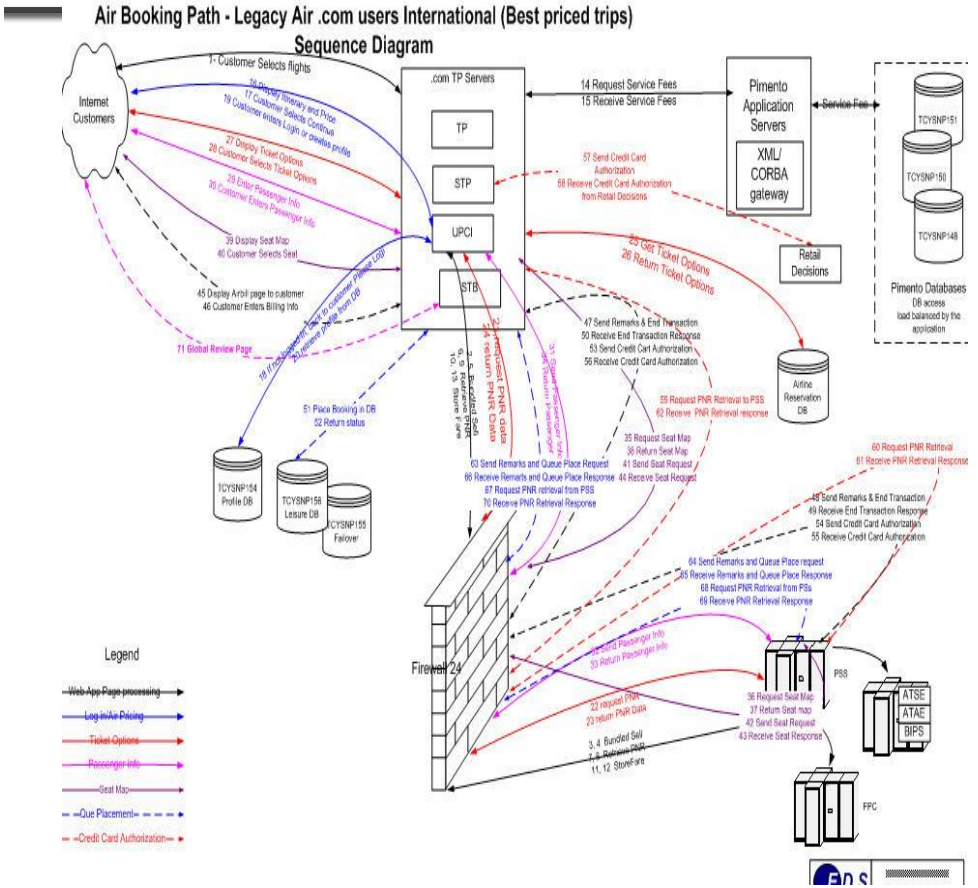
- Most SE projects still fail due to these important factors:
  - Increasing system size and complexity
  - Team-related issues such as lack of skilled personnel, poor team dynamics, etc.
  - Poor project management due to lack of clear objectives, insufficient planning, scope creep and ineffective risk management
  - Insufficient requirements analysis including unclear and changing requirements
  - Lack of stakeholder involvement such as poor communication and stockholders, conflicts
  - Inadequate Testing such as insufficient testing, delayed testing
  - Financial issues such as budget overruns and poor cost management
  - Lack of user involvement such as ignoring user feedback and poor user training and support
  - etc

Example: Complex server connections and complex message flow





Complex server connections



complex message flow

Dealing with complexity

i. Abstraction, ii. Decomposition, iii. Hierarchy

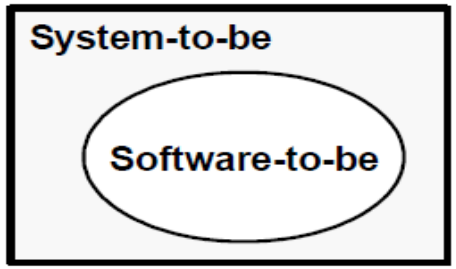
Professional software development

- SE is intended to support professional software development (SD) rather than individual programming
- Professional SD – involves a comprehensive and structured approach to creating, deploying, and maintaining software applications that meet user needs and technical specifications
  - Involves a combination of technical skills, methodologies, tools, and best practices to ensure that the software meets user needs, performs efficiently, and is maintainable over time.
- Key aspects of professional software development:
  - Software Development Life Cycle (SDLC) - writing programs to be used by someone else and following SDLC from requirement to deployment and maintenance, security, project management
    - Utilizing models such as waterfall, agile, DevOps, etc
    - Ensuing professionalism and ethics
  - It involved large-scale program and applications than can be several millions of lines code. E.g Windows 10 with 50 million lines, etc.
  - It involved teamwork with several hundred developers (Software Engineers)having various skills and knowledge in large projects.
- SD is a rewarding career path that offers many opportunities for growth and learning

SE Actors

Customer:

Requires a computer system to *achieve some business goals* by user interaction or interaction with the environment in a specified manner



Software Engineer's task:

To ***understand how*** the system-to-be needs to interact with the user or the environment so that customer's requirement is met and ***design*** the software-to-be

May be the same person

Programmer's task:

To ***implement*** the software-to-be designed by the software engineer

Frequently asked questions about SE

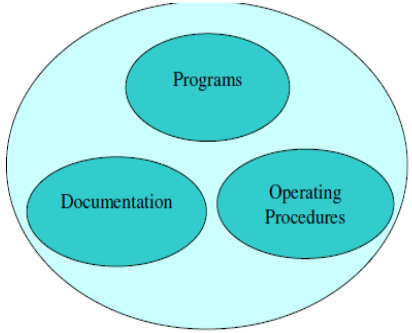
Question	Answer
What is software?	Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.
What are the attributes of good software?	Good software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable.
What is SE?	SE 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 SE and computer science?	Computer science focuses on theory and fundamentals; SE is concerned with the practicalities of developing and delivering useful software.
What is the difference between SE and system engineering?	System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. SE is part of this more general process.
What is the difference between SE and programming?	Programming is the process of writing code to implement a specific functionality or algorithm. SE is a broader term that includes programming as well as other aspects of software development, such as planning, analysis, design, testing, deployment, and maintenance

Frequently asked questions about SE

Question	Answer
What are the key challenges facing SE?	Coping with increasing diversity, demands for reduced delivery times and developing trustworthy software.
What are the costs of SE?	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 SE 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 SE?	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.

What is Software?

- Software is a collection of programs, operating procedures/libraries, associated documentation and data that perform specific tasks or operations on a computer system



Software=Program+Documentation+Operating Procedures

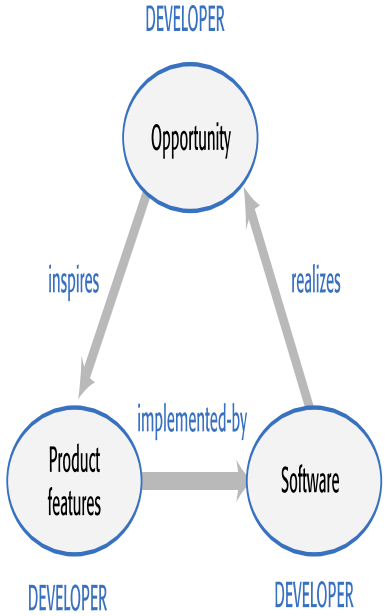
Components of software

  - It significantly affects nearly every aspect of our lives and pervasive in almost everything such as commerce, culture, etc.

- Software product is a software developed to meet specific requirements for end user
  - Software products may be developed for a particular customer (bespoke)or may be developed for a general market (generic).

Software Classes

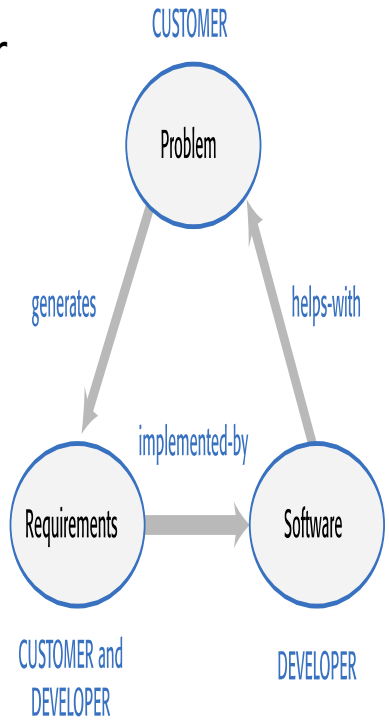
- Generic software (Market-driven software):
  - Pre-built or commercial off-the-shelf software (COTS) designed to provide general functionalities that is useful to a range of users and general purposes
    - typically produced, marketed and sold by software development companies to any customer who wishes to buy them.
  - Specification of what the software should do is owned by the software developer or organization producing the software and decisions on software change are made by them.
  - Key features: (advantage) cost-effective, mass-market availability, general functionality, community support and updates (disadvantage) limited customization, scalability issues and integration challenges
  - Examples – PC software such as project management tools, word processors, spreadsheets, operating systems, computer games etc.



Software Classes

- Customized Software (Bespoke software)
  - Custom-built software applications specifically developed to address unique needs and requirements of a particular customer (individual, organization, or business).
    - created from scratch, starting with a detailed analysis of the client's specific processes, workflows, and challenges.
  - Specification of what the software should do is owned by the customer for the software and they make decisions on software changes that are required.
  - Key features: (advantages) personalized solution, customization and flexibility, integration with existing systems, enhanced efficiency and productivity, competitive advantage

    (disadvantages) not cost-effective, takes longer to develop and implement, requires ongoing support and updates, which can be costly



- Examples – air traffic control software, traffic monitoring systems, process-control systems, hospital-management systems, etc

Essential attributes (quality) of good software

- Software quality is the degree to which a software product meets specific requirements and satisfies the needs of its users.
  - It is essential to ensure that the software delivers value to its users and functions as expected without significant defects or issues.

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.

SE Stakeholders vs Quality

- Software produced at anytime should be efficient, reliable, usable, modifiable, portable, testable, reusable, maintainable, interoperable, and correct

- What software quality means to different stakeholders

Customer: solves problems at an acceptable cost in terms of money paid and resources used

people who make the decisions about ordering and paying for the software

User: easy to learn; efficient to use; helps get work done

people who will use the software

Quality software

people who develop and maintain the software, many of whom may be called software engineers

Developer: easy to design; easy to maintain; easy to reuse its parts

People who run the organization that is developing the software

Development manager: sells more and pleases customers while costing less to develop and maintain

Software process activities

- In SE, we use Software development life-cycle (SDLC) to facilitate the development of large software product in a systematic, well-defined, and cost-effective way.
  - SDLC goal is to optimize the development process by reducing errors, risks, and costs so that the software meets customer expectations
- SDLC is structured into several phases or activities into tasks you can assign, complete, and measure:
- Software specification
  - **Objective**: customers and software engineers to identify the problems - gather and document the functional and non-functional requirements
  - **Activities**: Stakeholder meetings, requirement elicitation, and analysis.
- Software development - the software is designed and implemented
  - Design
    - **Objective**: analysed requirements are used to create a blueprint of software, outline its structure, components and interactions - architecture and design of the software.
    - **Activities**: High-level design and low-level design, design reviews.
  - Implementation
    - **Objective**: write the code by converting the design specifications into executable code using programming languages and tools
    - **Activities**: Coding, code reviews, and integration.

Software process activities

- Software validation - the software is checked to ensure that it is what the customer requires.
  - Testing
    - **Objective**: Ensure the software meets the requirements and is free of defects.
    - **Activities**: Unit testing, integration testing, system testing, acceptance testing.

- Software evolution - the software is modified to reflect changing customer and business/market requirements.
  - Maintenance
    - **Objective**: Updating, fixing, and improving the software over time
    - **Activities**: Bug fixes, updates, and performance improvements.

Software engineering fundamental Principles

- Some fundamental principles apply to all types of software system, irrespective of the development techniques used:
  - Systems should be developed using a managed and understood development process.
    - Of course, different processes are used for different types of software.
  - Dependability and performance are important for all types of system.
  - Understanding and managing the software specification and requirements (what the software should do) are important.
  - Where appropriate, you should reuse software that has already been developed rather than write new software.

Software engineering tools

- SE relies on various tools and techniques to enhance productivity and quality.
  - Programming languages : Python, Java, C++, etc.
  - Development tools: IDEs (Visual Studio Code, NetBeans, IntelliJ IDEA, PyCharm, etc)
  - Version control systems: Git, GitHub/GitLab, etc.
  - Other Essential Tools:
    - Build tools: Maven, Gradle for automating the build process.
    - Testing frameworks: JUnit, pytest for writing and running tests.
    - Containerization: Docker for creating, deploying, and running applications in containers.
    - Continuous Integration/Continuous Deployment (CI/CD): Jenkins, Travis CI for automating the software delivery process.
- Several different types of software system exist and there is no universal set of software techniques/methods and tools that is applicable to all of these systems.
- However, SE tools and techniques used depend on the:
  - type of software/application being developed
  - requirements of the customer
  - organization developing the software, and background of the development team

General issues that affect software

- Rapid advancement of technology:
  - Keeping up with the ever-evolving tech landscape can be challenging. New tools, frameworks, and languages emerge constantly, requiring engineers to adapt quickly.
- Growing customer demands:
  - Software projects must meet diverse client needs - understanding business concepts and incorporating necessary functionality is crucial, even for seemingly simple applications
- Time constraints:
  - Tight deadlines put pressure on development teams - working across different time zones can further complicate the process.
- Limited infrastructure/resources:
  - Insufficient resources or IT infrastructure can hinder successful project execution

General issues that affect software

- Software testing conflicts:
  - Balancing comprehensive testing with project timelines can be tricky - ensuring quality without compromising delivery is essential
- Changing requirements:
  - As projects progress, requirements often evolve. Engineers must adapt and accommodate these changes
- Security challenges:
  - Protecting software from vulnerabilities, cyber threats, and data breaches remains a critical concern

SE skills

- SE is a demanding and rewarding profession that requires a combination of technical skills and soft skills to succeed.
  - **Technical skills** - knowledge and abilities related to programming, software development, and computer science.
    - It include data structures and algorithms, databases, programming languages, frameworks and tools, version control testing, etc.
  - **Soft skills** - personal attributes and abilities related to communication, teamwork, problem-solving, creativity, etc.
    - It includes communication skills, teamwork skills, problem-solving skills, analytical skills, logical thinking skills, creativity skills and management skills.

SE skills

- Career options and opportunities
  - Software engineers can work in various domains and industries, such as
    - web development, mobile development, gaming, artificial intelligence, data science, cybersecurity, embedded systems, etc.
  - Software engineers roles and positions include:
    - **Entry-Level**: Junior developer, technical support specialist.
    - **Mid-Level**: Software engineer, system analyst.
    - **Senior-Level**: Senior developer, technical lead, architect, software tester.
    - **Leadership**: Project manager, engineering manager
  - Software engineers can also pursue further education or certification to advance their skills and knowledge, such as master's degree, doctoral degree, professional certification, etc.

Tips to up your SE skills

- SE is a dynamic and evolving field that requires continuous learning and improvement.
- Some of the tips for learning more about SE:
  - You have to stay updated with the latest trends and developments in SE by reading blogs, newsletters, podcasts, magazines, etc.
  - Practice your coding skills by solving problems or challenges on online platforms such as HackerRank, LeetCode, CodeChef, etc.
  - Build your portfolio by creating projects or applications using different tools and technologies and showcasing them on platforms such as GitHub, etc.
  - Network with other software engineers by joining online communities or forums such as Stack Overflow, Reddit, Quora, etc., or attending events or meetups such as Hackathons, Conferences, Workshops, etc.
  - Seek feedback or mentorship from experienced software engineers by asking questions or seeking advice on platforms such as Stack Overflow, Reddit, Quora, etc., or finding mentors or coaches on platforms such as MentorCruise, Codementor, Experfy, etc.
- Important SE resources
  - Register for online courses at your own pace and convenience – popular online platforms that offer courses in SE are Coursera, edX, Udemy, Udacity, Pluralsight, etc.
  - Read important SE books such as: “Code Complete”, “Clean Code” “Design Patterns: Elements of Reusable Object-Oriented Software”, “The Pragmatic Programmer: From Journeyman to Master” etc.

Emerging technologies in Software Engineering

- SE is a rapidly evolving field and there have been significant advances in recent years in addition to web-based systems and cloud computing via internet such as:

- Artificial Intelligence (AI) - transforming software development by automating tasks, enhancing code quality, and predicting potential issues.
  - Applications: AI-powered tools like GitHub Copilot assist in code generation, debugging, and testing

- Machine Learning (ML) - its algorithms enable systems to learn from data and improve over time without explicit programming.
  - Applications: Used in predictive analytics, natural language processing, and recommendation systems

- Blockchain - decentralized ledger technology that ensures secure and transparent transactions.
  - Applications: Widely used in cryptocurrencies, smart contracts, and supply chain management

Future directions in Software Engineering

- Low-code/No-code development - platforms that allow users to create applications with minimal or no coding.
- Cybersecurity - increasing focus on securing software against cyber threats.
- Remote work - the shift towards remote and hybrid work environments.
- AI and automation - continued integration of AI and automation in software development.
- Sustainability - emphasis on developing sustainable and energy-efficient software.

SE ethical consideration

- SE 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 such as:
  - Privacy - ensure the protection of user data and respecting privacy rights.
  - Security – design and develop software with robust security measures to prevent unauthorized access and data breaches.
  - Intellectual Property - respect copyright and intellectual property rights when using or incorporating others' work into software projects.
  - Quality – ensure the software meet quality standards.
  - Social Impact - consider the broader impact of software on society and taking steps to minimize negative consequences.

Professional responsibilities and Code of Conduct

- Professionalism – maintain a high level of professionalism in all interactions and communications with clients, colleagues, and stakeholders.
- Competence – continue to improve your technical skills and staying up to date with industry best practices.
  - Don't misrepresent your level of competence or knowingly accept work which beat your competence.
- Honesty and integrity - always be truthful and acting with integrity in all professional activities.
- Collaboration – work effectively as part of a team, respecting diverse perspectives and promoting a positive work environment
- Confidentiality - safeguard or respect confidential information obtained during work irrespective of whether or not a formal confidentiality agreement has been signed.
- Compliance – always adhere to legal and regulatory requirements related to software development and usage.
- Computer misuse – don't use your technical skills to misuse other people's computers.
  - ranges from relatively trivial (game playing on an employer's machine, say) to extremely serious (dissemination of viruses).

Thank You

Next is Software Processes and Models