Software Engineering

Lecture 1

Instructor : Alisha Farman (Alisha.farman@iqra.edu.pk)

Your Responsibility

- Attend classes regularly
- Ask question if you have any queries regarding course material or anything.
- Submit assignment in time.
- Don’t miss quizzes, assignments and examinationsCan get good result.
- No Plagiarism is allowed in any sort of a writer material – Write in your own words.

Course Policy

<u>Assignments:</u>

• Assignments are due at the beginning of class. • Late assignment will not be accepted.

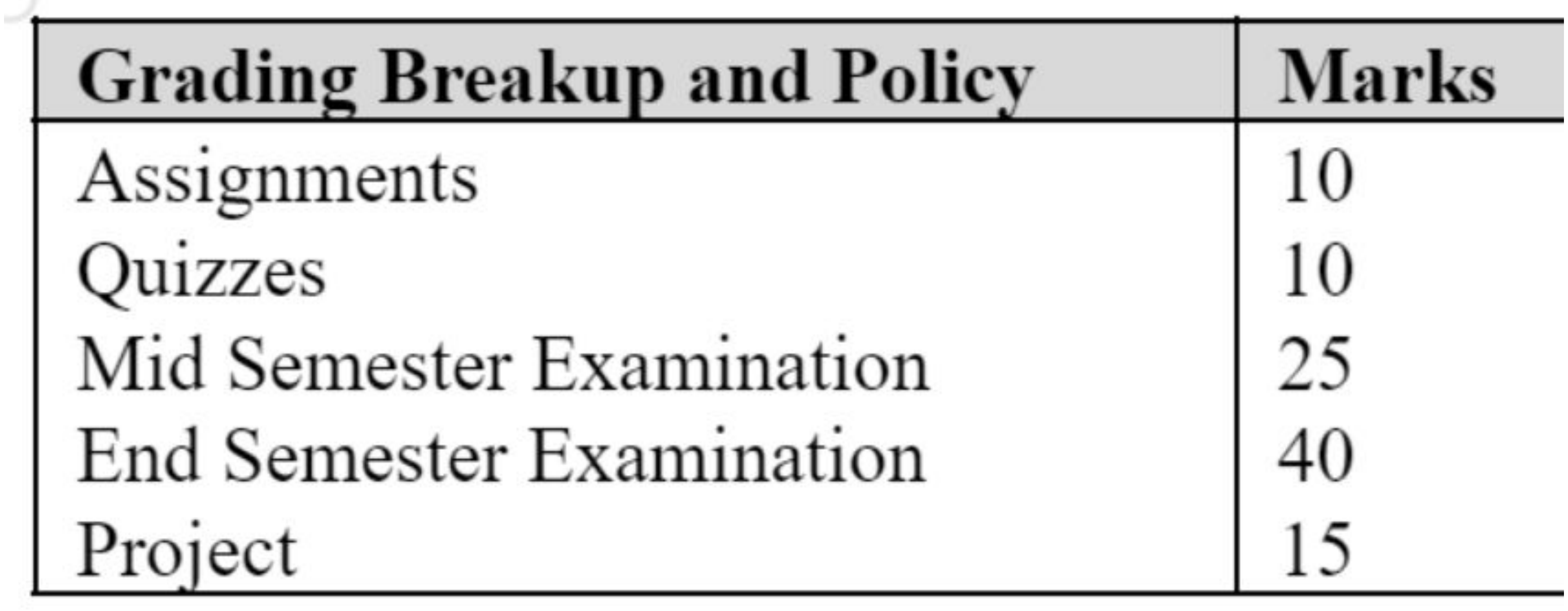
• All works have to be done independently except in case of group assignments.

• Students handing in similar assignments will receive a grade of 0 (Zero).

<u>Attendance:</u>

• Students are expected to attend all classes.

Grading Criteria

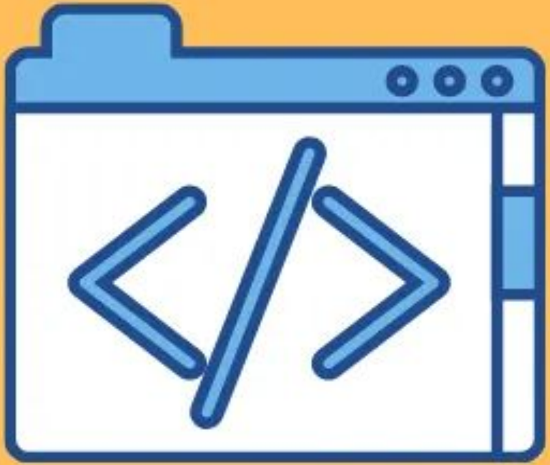


Grading Breakup and Policy	**Marks**
Assignments	10
Quizzes	10
Mid Semester Examination	25
End Semester Examination	40
Project	15

RECOMMENDED BOOKS:

1. **Software Engineering, Sommerville I., 10th Edition, Inc., Pearson 2014**
2. **Software Engineering, A Practitioner’s Approach, Pressman R. S.& Maxim B. R., 8th Edition, McGraw-Hill, 2015**

What is Software?

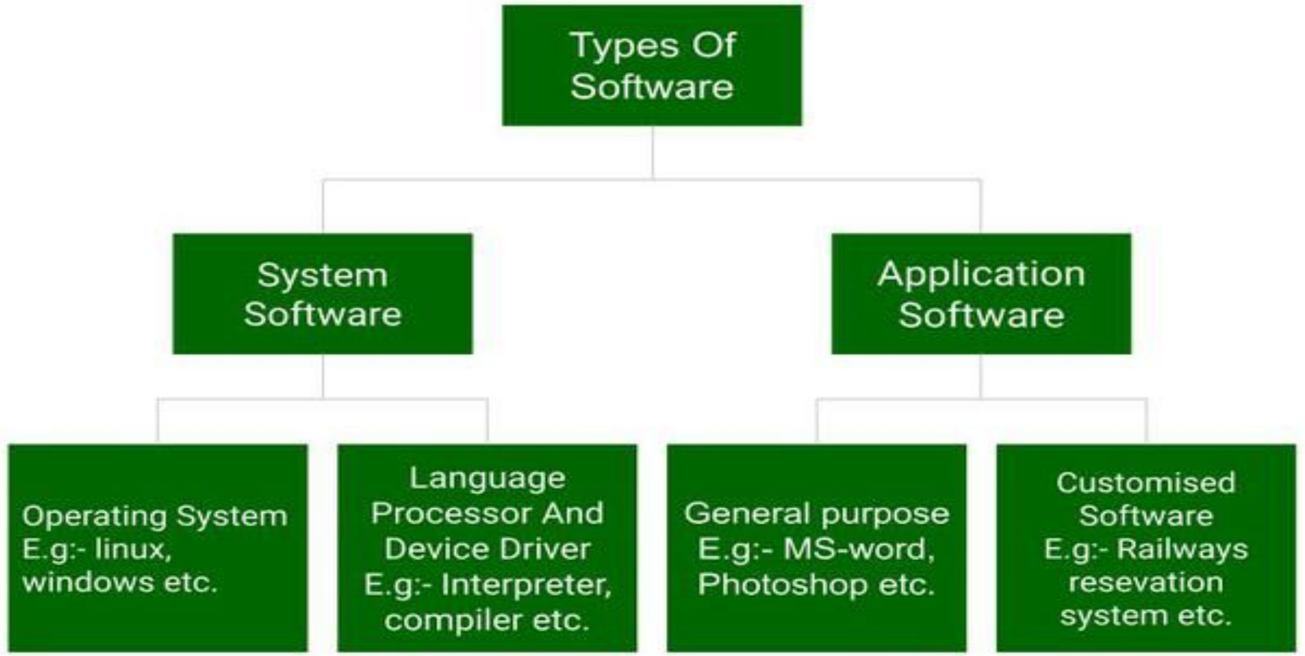


What is Software?

Software is a set of instructions that tells a computer what to do. It's like the brain of the computer that makes everything work.

Software are all the programs used in a computer to perform certain tasks.

Software refers to a collection of instructions, data, or programs that enable a computer to perform specific tasks. It is the non-physical component of a computer system, as opposed to hardware, which comprises the physical parts.


Types Of Software
System Software
Application Software
Operating System E.g:- linux, windows etc.
Language Processor And Device Driver E.g:- Interpreter, compiler etc.
General purpose E.g:- MS-word, Photoshop etc.
Customised Software E.g:- Railways resevation system etc.

Types of Software

There are three major types of software:

- **System Software:** Includes operating systems, device drivers, and utilities that manage hardware and provide a platform for running application software.

- **Application Software:** Programs that perform specific tasks for the user, such as word processors, spreadsheets, and media players.

- **Embedded Software:** Software built into devices like phones, cars, and appliances to control their functions.

✅ Examples of Embedded Software:

Device	Embedded Software ka Role
Washing Machine	Wash cycle control karta hai (e.g. time, spin speed)
Microwave Oven	Heat time set karna, power level control karna
Digital Camera	Image capture, zoom control, storage handling
Smart TV	Interface, app control, remote ke inputs handle karna
ATM Machine	Card read karna, transactions process karna

Examples of Software

- Operating Systems: These are system software that manage the computer’s hardware and software resources.
  Examples: Windows, macOS, Linux, Android, iOS.
- Web Browsers: These are application software that allow users to browse the internet.
  Examples: Google Chrome, Mozilla Firefox, Safari
- Word Processors: These are applications for creating, editing, and formatting text documents.
  Examples: Microsoft Word, Google Docs
- Media Players: Software used to play audio and video files.
  Examples: VLC Media Player, Windows Media Player, iTunes.

Examples of Software

- **Photo Editors:** Software used to edit and enhance images.
  Examples: Adobe Photoshop, GIMP, Canva, Capcut
- **Messaging Apps:** Applications for sending messages or making calls.
  Examples: WhatsApp, Facebook Messenger, Slack.
- **Games:** Entertainment software that allows users to play digital games.
  Examples: Crazy taxi, Pubg, Call of Duty.
- **Development Tools:** Software used by programmers to write and test code.
  Examples: Visual Studio, Eclipse, Xcode, android studio.

What are the attributes of good software?

The attributes of good software describe what makes software reliable, effective, and user-friendly.

1. Maintainability

- The software should be easy to update and modify when necessary, whether it's for fixing bugs, adding new features, or adapting to new environments.

2. Reliability

- Good software performs consistently under normal conditions without crashing or producing incorrect results. It should work correctly and as expected.

3. Efficiency

- The software should use system resources (like memory and processing power) wisely, ensuring that it runs quickly and doesn't slow down the device.

4. Usability

- The software should be easy for users to understand and operate. A well-designed user interface (UI) and good user experience (UX) make it more accessible and enjoyable to use.

What are the attributes of good software?

5. Portability

- Good software can run on different platforms or environments (like Windows, macOS, or Linux) without needing major changes.

6. Security

- The software should protect user data and resources from unauthorized access, hackers, and other security threats. It should ensure privacy and data integrity.

7. Functionality

- The software must meet the needs of its users by providing the correct features and capabilities. It should perform the tasks it was designed for without errors or missing functionality.

What are the attributes of good software?

8. **Scalability**

- The software should be able to handle an increasing number of users, data, or tasks without a drop in performance. This is important for software that is expected to grow over time.

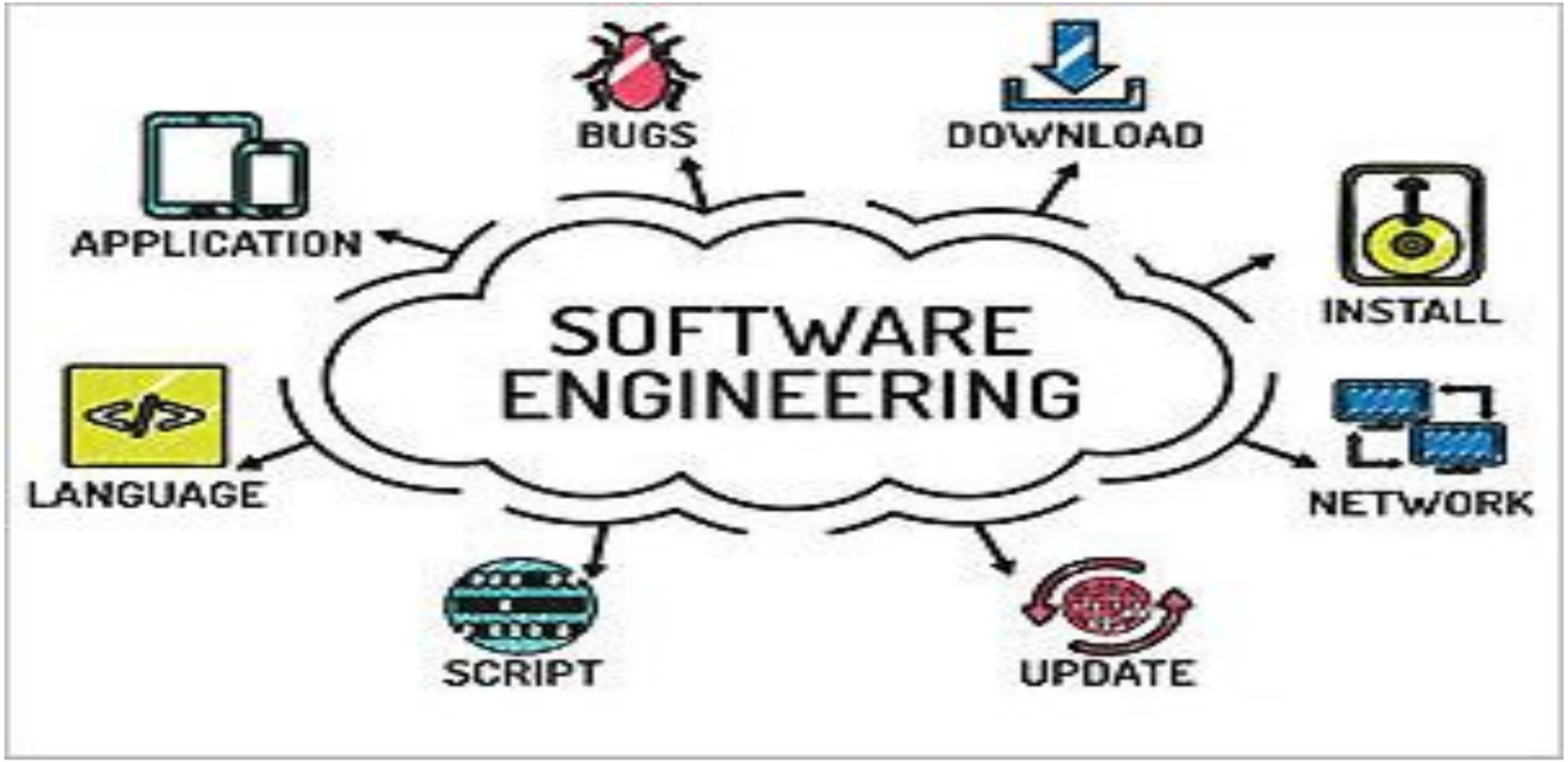
9. **Testability**

- It should be easy to test the software to find and fix any bugs or problems. Testing should be simple and thorough to ensure high quality.

What are the attributes of good software?

Good software has these features:

- **Easy to Update:** Changes can be made without much trouble (maintainable).
- **Reliable:** It works without crashing or failing.
- **Fast and Efficient:** Uses resources wisely so it doesn’t slow down your computer.
- **Easy to Use:** It’s user-friendly and simple to understand.
- **Secure:** Keeps your data safe from hackers.


BUGS
DOWNLOAD
APPLICATION
INSTALL
SOFTWARE ENGINEERING
LANGUAGE
NETWORK
SCRIPT
UPDATE

What is Software Engineering?

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

Software engineering is the process of designing, developing, testing, and maintaining software in a systematic, efficient, and reliable way.

It involves applying engineering principles to ensure that the software is high-quality, meets user needs, and can be easily maintained and updated.

1.1.1 Software engineering

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. In this definition, there are two key phrases:

1. *Engineering discipline* Engineers make things work. They apply theories, methods, and tools where these are appropriate. However, they use them selectively and always try to discover solutions to problems even when there are no applicable theories and methods. Engineers also recognize that they must work to organizational and financial constraints so they look for solutions within these constraints.

2. *All aspects of software production* Software engineering is not just concerned with the technical processes of software development. It also includes activities such as software project management and the development of tools, methods, and theories to support software production.

Software engineering is important for two reasons:

1. More and more, individuals and society rely on advanced software systems. We need to be able to produce reliable and trustworthy systems economically and quickly.

2. It is usually cheaper, in the long run, to use software engineering methods and techniques for software systems rather than just write the programs as if it was a personal programming project. For most types of systems, the majority of costs are the costs of changing the software after it has gone into use.

Difference between Computer science, System and software engineering

Computer Science (CS): Focuses on the theoretical foundations of information and computation. It includes studying algorithms, data structures, artificial intelligence, and more. Computer science covers a broad range of topics, from computing theory to hardware systems, but it is less concerned with the practicalities of building software products for specific users.

System Engineering (SE): Focuses on the overall design, integration, and management of complex systems over their life cycles. System engineering involves hardware, software, people, processes, and infrastructure to deliver functioning systems. It is a multidisciplinary field that spans engineering, project management, and software engineering.

Software Engineering: A subfield of system engineering that focuses solely on software products. It deals with the entire software development life cycle, including requirements analysis, design, implementation, testing, deployment, and maintenance.

📊 Comparison Table:

Feature	Computer Science	Software Engineering	System Engineering
🎯 Focus	Theory, algorithms, logic	Software design & development	Complete system (hardware + software)
🧑‍💻 Main Work	Problem-solving, research	Building applications	Managing large systems
🧪 Core Areas	AI, data science, networks	SDLC, testing, coding	Integration, system design
⚙️ Tools Used	Programming languages, compilers	IDEs, UML, testing tools	Modeling tools, simulation
📈 Output	Concepts, prototypes	Software products	Entire systems (e.g., aircraft, hospitals)

What are the fundamental software engineering activities?

The fundamental software engineering activities include

1. **Software specification (Software ki zaruratein likhna aur samajhna)**
2. **Software development (Actual software banana – coding wala part)**
3. **Software validation (Check karna ke software sahi kaam kar raha hai ya nahi)**
4. **Software evolution(Software ka waqt ke saath update hona)**

The systematic approach that is used in software engineering is sometimes called a software process. A software process is a sequence of activities that leads to the production of a software product. There are four fundamental activities that are common to all software processes. These activities are:

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

What are the Key Components of Software Engineering?

Software engineering includes multiple components which help in the overall software development process. Let’s understand each of the components in detail.

- **Requirements analysis:** You identify and understand the requirements of the end-users, technical, and functional constraints.
- **Software design:** Creating a detailed plan that defines the software’s architecture, components, interfaces, and data structures.
- **Implementation:** Writing the code and converting the design into a working software application.
- **Software testing:** Ensuring that the software functions as intended, and identifying and fixing any bugs or issues.
- **Deployment:** Distributing the software to end-users and making sure it works properly in the intended environment.
- **Maintenance:** Updating, improving, and fixing the software over time to meet changing requirements or address issues.

How Does Software Engineering Differ from Programming?

Software engineering and programming, both involve writing code. Computer programming focuses on the act of writing code to create software, whereas software engineering involves the entire software development lifecycle, from requirements analysis to maintenance.

In other words, software engineering is a more comprehensive approach to building software solutions. On the contrary, computer programming refers to only writing code for a software solution.

5 Core Challenges Faced in Software Engineering

1. **Complex and Evolving Requirements**
   As a software engineer, you'll face complex and ever-changing requirements. It's crucial to effectively gather, analyze, and prioritize features while accommodating changes during development. Failing to address these complexities can lead to delays, increased costs, and unmet user requirements.

2. **Scalability and Performance**
   You need to build software solutions that scale effectively and maintain high performance as the user base or data volume grows. Consider factors such as load balancing, data partitioning, and caching strategies. Ignoring scalability and performance issues can result in slow, unresponsive software that frustrates users and hinders growth.

3. **Integration With Existing Systems and Technologies**
   Your software often needs to interact with other systems, APIs, and technologies.The inability to integrate effectively with other systems may limit your software's usefulness and impact.

5 Core Challenges Faced in Software Engineering

3. Security and Privacy

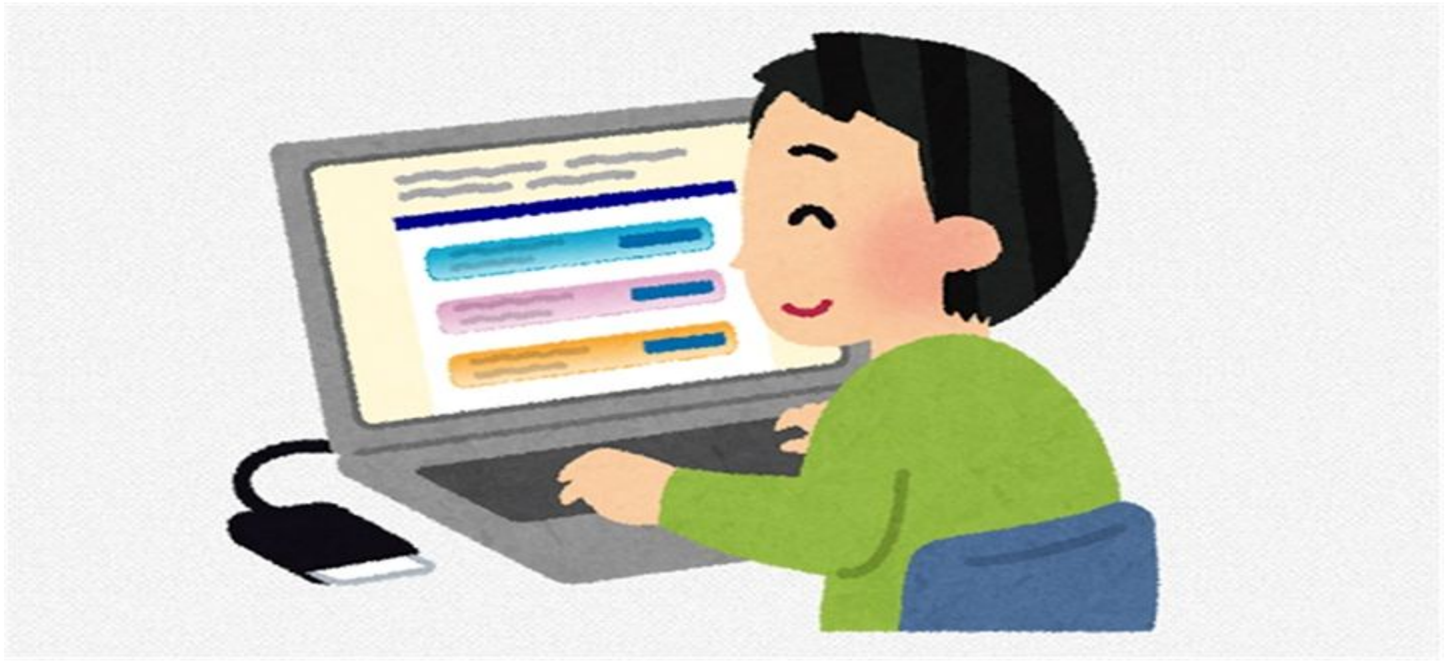
As a software engineer, it's essential to ensure your software is secure and protects user data. Employ safe coding practices, encryption, and authentication mechanisms, and stay updated on relevant regulations. Failure to address security and privacy challenges can lead to data breaches, loss of user trust, and legal consequences.

4. Effective Team Collaboration

Software projects involve diverse development teams, so facilitating effective communication and collaboration is crucial. Adopt agile methodologies, use collaboration tools, and foster a culture of open communication and shared ownership. Overcoming team collaboration challenges helps increase the likelihood of delivering successful, high-quality software products on time and within budget.

Who wants to be a Software Engineer?

Software engineers are concerned with developing software products (i.e., software which can be sold to a customer).



UNFINISHED WORK
FRIDAY EVENING
PERFECT! I'LL FINISH THIS ON MONDAY
MONDAY MORNING...
WHAT DOES THIS MEAN!?!?
MONKEYUSER.COM

Software Products

There are two kinds of software products:

1. **Generic products** These are stand-alone systems that are produced by a development organization and sold on the open market to any customer who is able to buy them. Generic products are versatile and widely applicable software solutions available for purchase by any customer.

Examples:

- Microsoft Office Suite: Includes applications like Word, Excel, and PowerPoint that can be used by anyone for various tasks.
- Adobe Photoshop: A widely used software for photo editing and graphic design, available to a broad audience.
- Web Browsers: Google Chrome, Mozilla Firefox, and Safari, which are available for anyone to use for browsing the internet.
- Video Conferencing Tools: Zoom, Microsoft Teams, and Skype, which offer video communication services for individuals and organizations.
- Project Management Tools: Software like Trello, Asana, or Microsoft Project that can be used by any team for task and project management.

Software Products

2. Customized (or bespoke) products These are systems that are commissioned by a particular customer. A software contractor develops the software especially for that customer. customized products are specifically designed and developed for the unique needs of individual clients or organizations. Each type serves different purposes and caters to different market segments.

Examples:

- **Mobile Applications:** A restaurant might hire developers to create a mobile app for ordering food and making reservations tailored to their specific menu and service
- **Healthcare Information Systems:** Custom systems developed for specific hospitals or clinics to manage patient data, appointments, billing, and compliance with health regulations.
- **Air Traffic Control Systems:** These systems are developed to meet the specific regulatory and operational requirements of a particular country's aviation authority.
- **Healthcare Management Systems:** Custom software created for a hospital or clinic to manage patient records, appointments, billing, and other functions tailored to their specific needs.
- **School Management System for an Educational Institution:** A private school commissions a software system to manage student enrollment, attendance, grade books, parent-teacher communication, and extracurricular activities specific to the institution's policies.

What is CASE?

Computer-aided software engineering (CASE) is the implementation of computer-facilitated tools and methods in software development. CASE is used to ensure high-quality and defect-free software. CASE ensures a check-pointed and disciplined approach and helps designers, developers, testers, managers, and others to see the project milestones during development.

Computer-Aided Software Engineering (CASE) refers to the use of software tools to assist in the software development process. These tools help automate various stages of software development, including planning, designing, coding, testing, and maintenance.

CASE tools woh software hote hain jo software banana aasan banate hain. Jaise Microsoft Word document likhne mein madad karta hai, waise hi CASE tools software banane mein madad karte hain.

Key Components of CASE

Tools and Software:

CASE encompasses a wide range of tools, including:

- **Modeling Tools:** Used for creating diagrams and models (e.g., UML diagrams, flowcharts) to represent system architecture and design.
- **Code Generators:** Automatically generate code from models or specifications, reducing manual coding effort.(Aapne UML class diagram banayi – us diagram ko ek tool (jaise StarUML) mein daala – tool ne Java ya C++ ka code automatically generate kar diya.)
- **Testing Tools:** Assist in automating testing processes, including unit testing, integration testing, and system testing.
- **Documentation Tools:** Help generate and manage project documentation, such as requirements specifications, design documents, and user manuals.
- **Project Management Tools:** Aid in planning, scheduling, and tracking project progress, resources, and risks.

Methodologies:

CASE often incorporates various methodologies and frameworks to guide the software development process, such as:

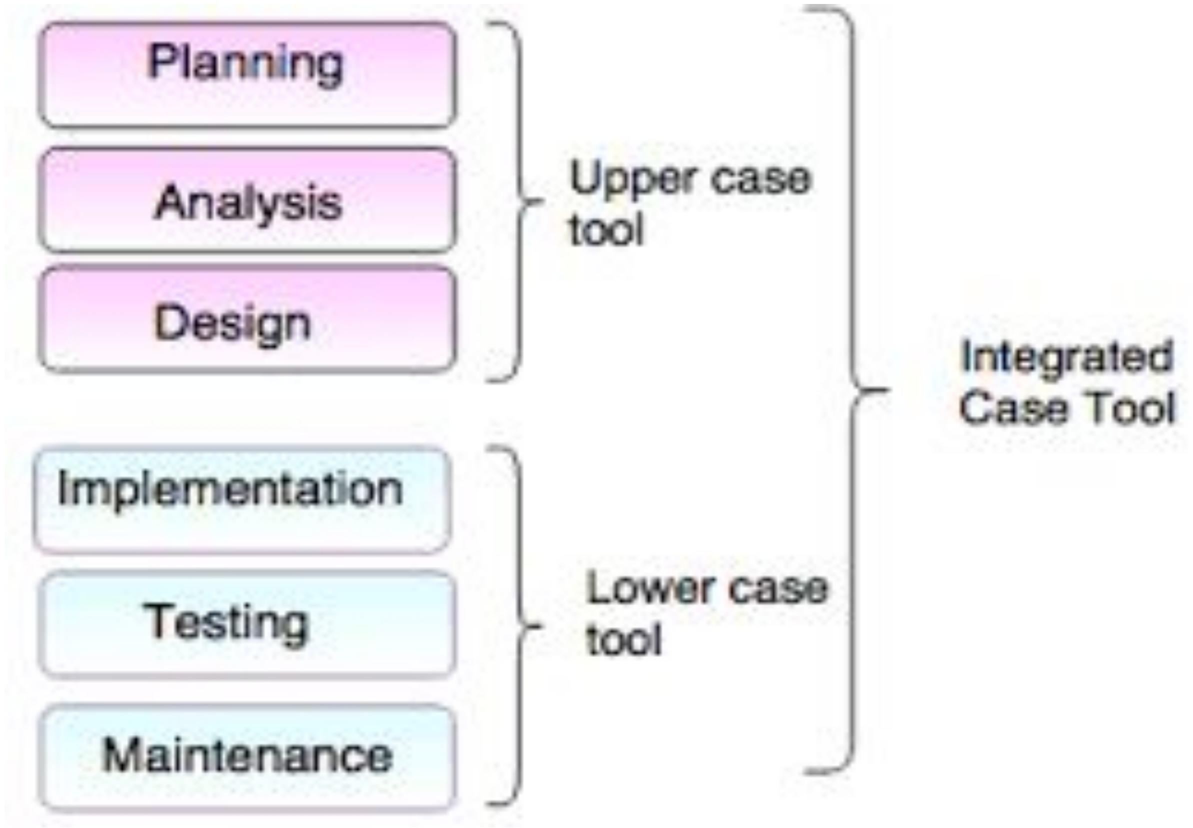
- **Agile:** Focusing on iterative development and collaboration.
- **Waterfall:** Following a linear, sequential approach.

CASE TOOLS

CASE tools are classified into two main categories:

Upper CASE tools: Focus on the early stages of software development, such as requirements analysis, system design, and architectural modeling.

Lower CASE tools: Focus on later stages, such as code generation, testing, debugging, and maintenance.


Planning
Analysis
Design
Upper case tool
Implementation
Testing
Maintenance
Lower case tool
Integrated Case Tool

Benefits of Using CASE Tools

Improved Productivity: Automation of tasks like design, coding, and testing speeds up the development process.

Consistency: Helps ensure that all parts of the software follow the same design principles and coding standards.

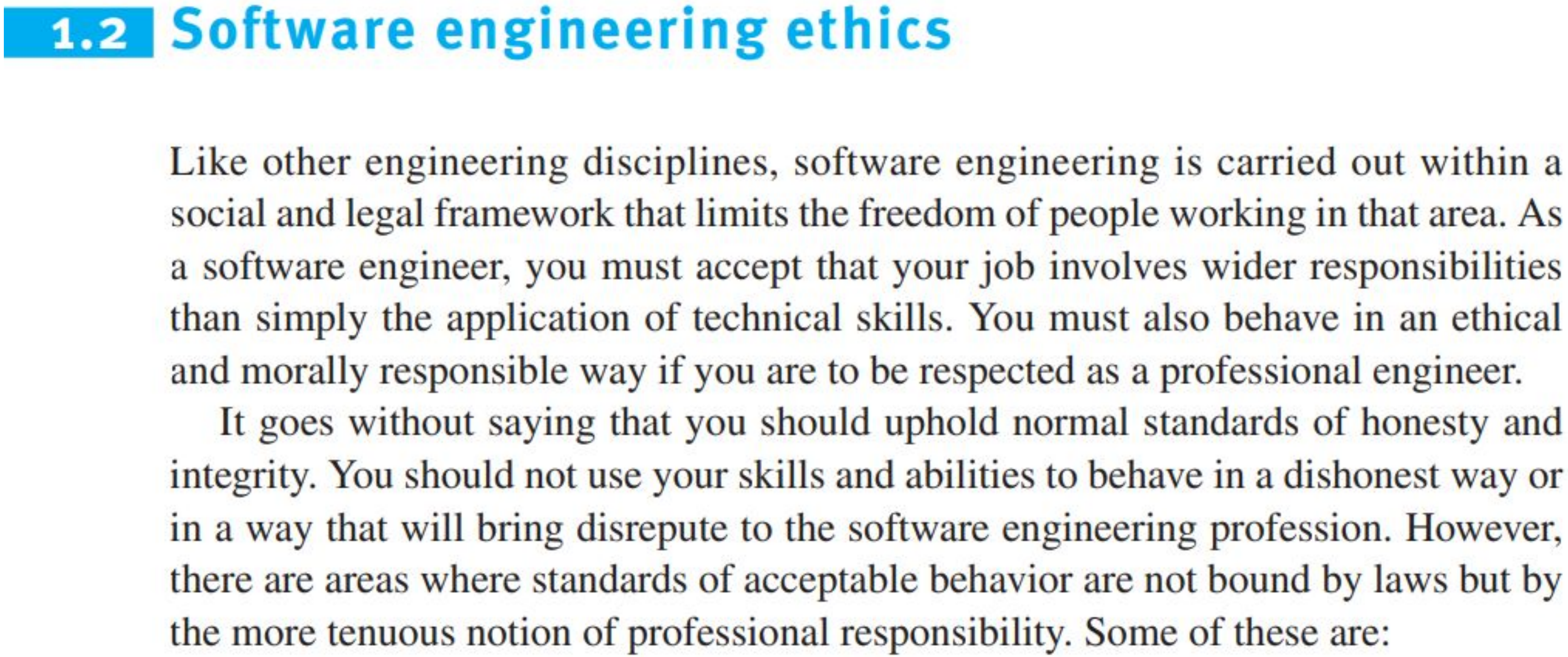
Error Reduction: Early detection of errors and inconsistencies, reducing bugs and rework during later stages.

Better Documentation: Automatically generates documentation for models, designs, and code.

Efficient Collaboration: Facilitates collaboration between teams by centralizing design and development efforts.

Quality Improvement: Enhances software quality by providing tools for testing and validation.

Software engineering ethics

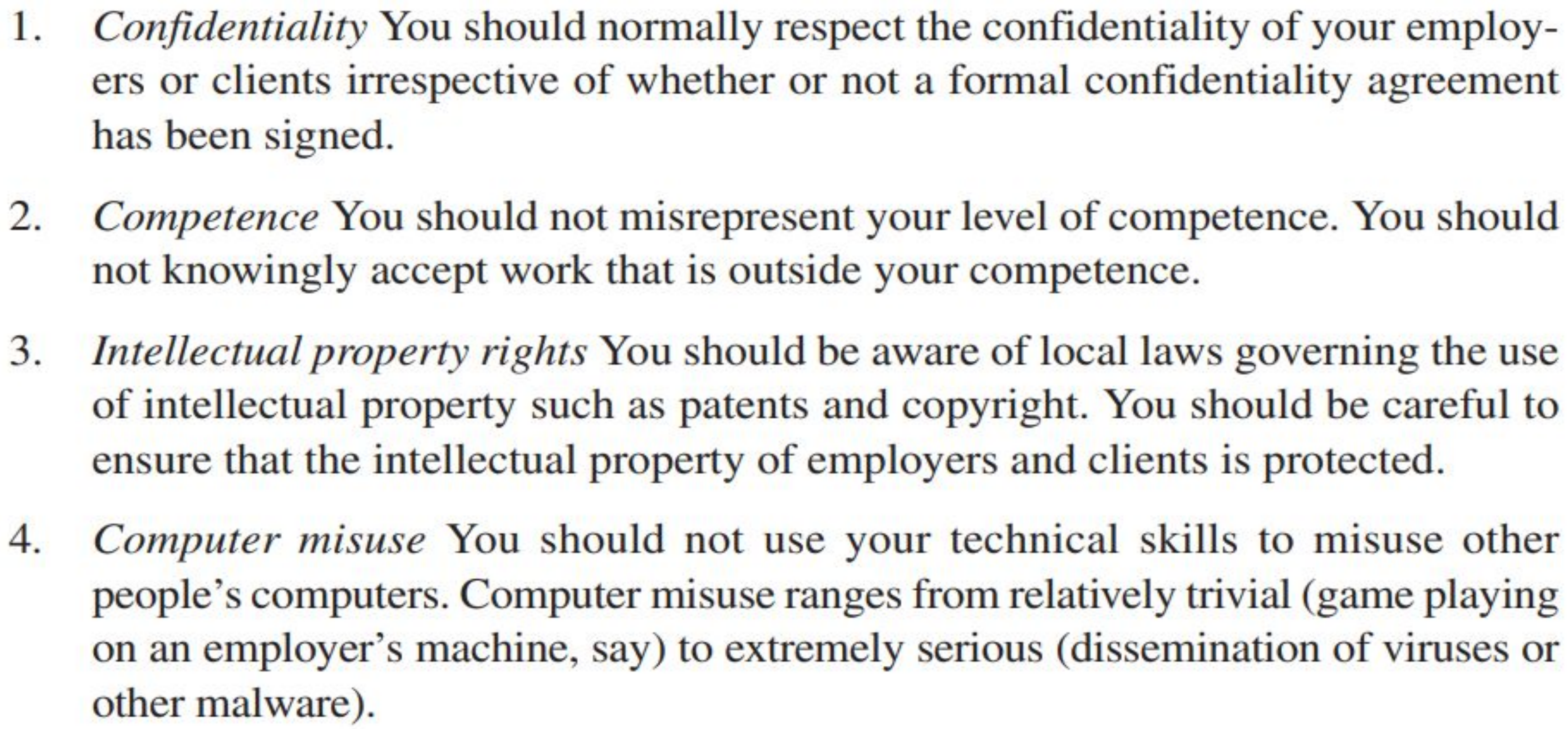


1.2 Software engineering ethics

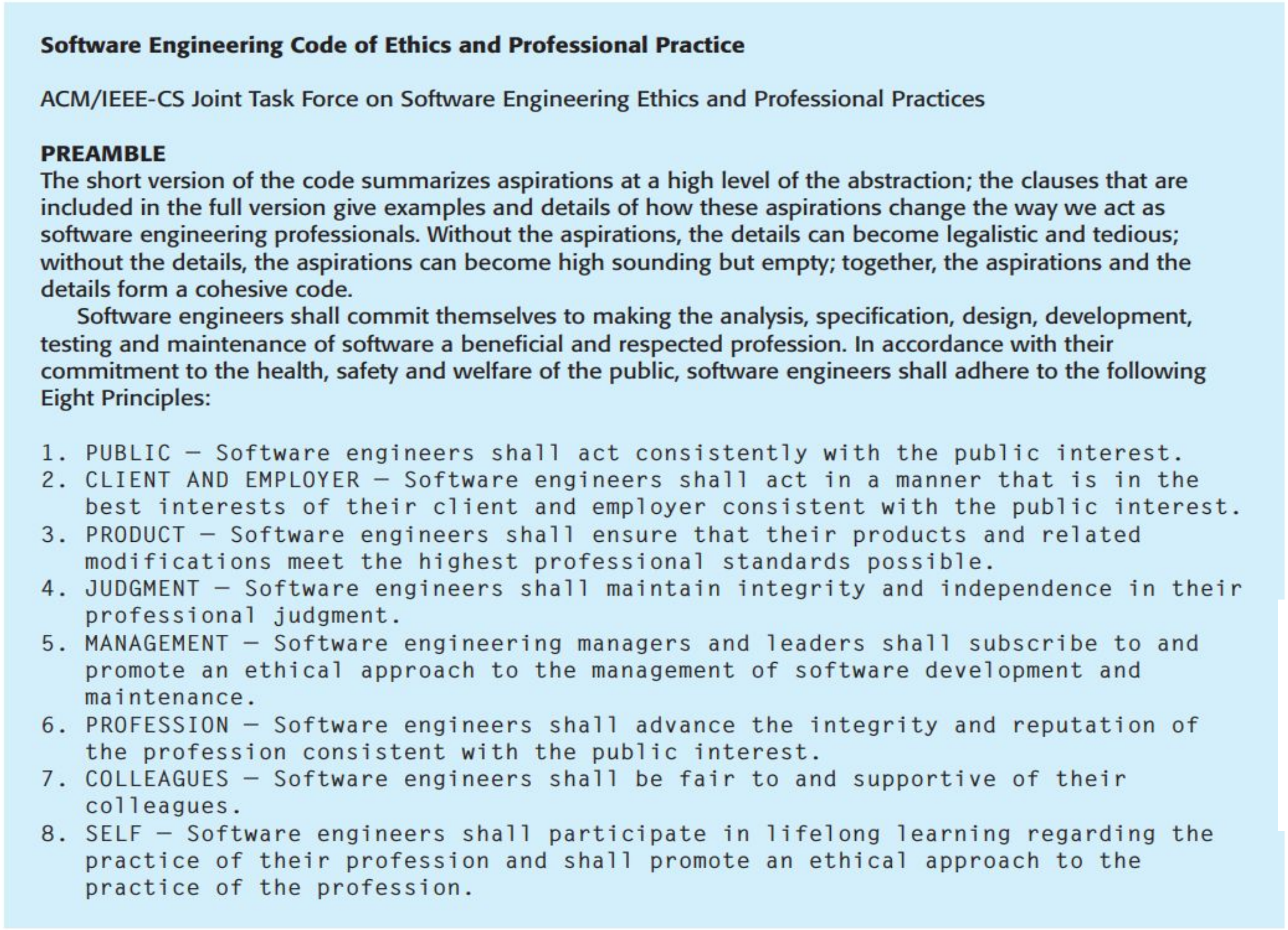
Like other engineering disciplines, software engineering is carried out within a social and legal framework that limits the freedom of people working in that area. As a software engineer, you must accept that your job involves wider responsibilities than simply the application of technical skills. You must also behave in an ethical and morally responsible way if you are to be respected as a professional engineer.

It goes without saying that you should uphold normal standards of honesty and integrity. You should not use your skills and abilities to behave in a dishonest way or in a way that will bring disrepute to the software engineering profession. However, there are areas where standards of acceptable behavior are not bound by laws but by the more tenuous notion of professional responsibility. Some of these are:

Software engineering ethics



1. *Confidentiality* You should normally respect the confidentiality of your employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.
2. *Competence* You should not misrepresent your level of competence. You should not knowingly accept work that is outside your competence.
3. *Intellectual property rights* You should be aware of local laws governing the use of intellectual property such as patents and copyright. You should be careful to ensure that the intellectual property of employers and clients is protected.
4. *Computer misuse* You should not use your 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 or other malware).



Software Engineering Code of Ethics and Professional Practice

ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices

PREAMBLE

The short version of the code summarizes aspirations at a high level of the abstraction; the clauses that are included in the full version give examples and details of how these aspirations change the way we act as software engineering professionals. Without the aspirations, the details can become legalistic and tedious; without the details, the aspirations can become high sounding but empty; together, the aspirations and the details form a cohesive code.

Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession. In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to the following Eight Principles:

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.
4. JUDGMENT — Software engineers shall maintain integrity and independence in their professional judgment.
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.
7. COLLEAGUES — Software engineers shall be fair to and supportive of their colleagues.
8. SELF — 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.

Figure 1.3 The ACM/IEEE Code of Ethics (© IEEE/ACM 1999)

Examples of Ethical Issues in Software Engineering:

Data Breaches: Failing to implement adequate security measures, leading to unauthorized access to sensitive user data.

Algorithmic Bias: Developing algorithms that perpetuate discrimination or bias against certain groups based on race, gender, or socioeconomic status.

Intellectual Property Theft: Using proprietary code or technology without permission, violating copyright laws and ethical standards.

Misleading Software Claims: Making exaggerated or false claims about the capabilities or performance of software products.

SOFTWARE ENGINEERING ETHICS

Here are some key aspects of software engineering ethics:

Professional Conduct

- **Integrity:** **Software engineers should act with honesty and integrity in their professional work. This includes accurately representing their qualifications, capabilities, and the potential of the software they develop.**
- **Competence:** **Engineers should only undertake work that they are competent to perform and should seek to improve their skills and knowledge continuously.**

Respect for Privacy

- **Data Protection:** **Software engineers must respect user privacy and ensure that personal data is collected, stored, and processed securely. They should comply with relevant data protection regulations.**
- **User Consent:** **Users should be informed about how their data will be used and should give explicit consent before data collection.**

Software engineering ethics

Transparency and Accountability

- **Disclosure:** Software engineers should disclose any potential conflicts of interest, as well as limitations or risks associated with the software they develop.
- **Accountability:** Engineers should take responsibility for their work and the consequences of software failures or misuse.

Quality Assurance

- **Commitment to Quality:** Software engineers should strive to produce high-quality software that is reliable, secure, and free from defects. This includes proper testing and validation.
- **Ethical Testing:** Engineers should avoid using deceptive practices in software testing and ensure that the software behaves as expected.

Impact on Society

- **Social Responsibility:** Software engineers should consider the societal impact of their work, including potential harm or benefits to individuals, communities, and the environment.

Software Engineering Ethics

Compliance with Laws and Regulations

- **Legal Standards:** Software engineers must adhere to relevant laws and regulations governing software development, intellectual property, and cybersecurity.
- **Ethical Use of Software:** Engineers should avoid creating or supporting software that promotes illegal or unethical activities, such as hacking, piracy, or discrimination.

Professional Development and Mentorship

- **Continuous Learning:** Engineers should commit to lifelong learning and professional development, staying updated on best practices, technologies, and ethical considerations in software engineering.
- **Mentorship:** Experienced engineers should mentor and guide junior professionals, promoting ethical behavior and professional growth in the field.

Collaboration and Respect

- **Teamwork:** Engineers should work collaboratively with colleagues, respecting diverse perspectives and contributions. They should communicate effectively and foster a positive team environment.
- **Conflict Resolution:** When conflicts arise, engineers should approach them professionally and ethically, seeking resolution through dialogue and compromise.

Discussion

The “Drone Revolution” is currently being debated and discussed all over the world. Drones are unmanned flying machines that are built and equipped with various kinds of software systems that allow them to see, hear, and act. Discuss some of the societal challenges of building such kinds of systems.

Common Myths of Software Engineering

Myth: Software Development is Just Coding

- **Reality:** Software engineering encompasses much more than just writing code. It includes requirements gathering, design, testing, and maintenance.

Myth: Adding More Developers Speeds Up Development

- **Reality:** Adding more developers to a late project can slow it down due to increased coordination and communication overhead (Brooks' Law).

Myth: Requirements Are Always Clear and Fixed

- **Reality:** Requirements often evolve over time as stakeholders gain a better understanding of their needs. Agile methodologies address this by allowing for flexibility.

Myth: Testing is an Afterthought

- **Reality:** Testing should be integrated throughout the development process (test-driven development), not just at the end, to catch defects early.

Common Myths of Software Engineering

Myth: Good Software Can Be Built Quickly

- **Reality:** Quality software requires time for planning, design, development, testing, and refinement. Shortcuts can lead to technical debt and poor quality.

Myth: More Features Mean Better Software

- **Reality:** Adding unnecessary features can complicate software and detract from its core functionality. Focus on delivering essential features first.

Myth: Documentation is Unnecessary

- **Reality:** Proper documentation is crucial for maintaining software, onboarding new team members, and ensuring that knowledge is preserved over time.

Myth: Software Development is a Linear Process

- **Reality:** Software development is often iterative and can involve revisiting earlier phases (requirements, design, testing) based on feedback and changes.

Common practices in software engineering

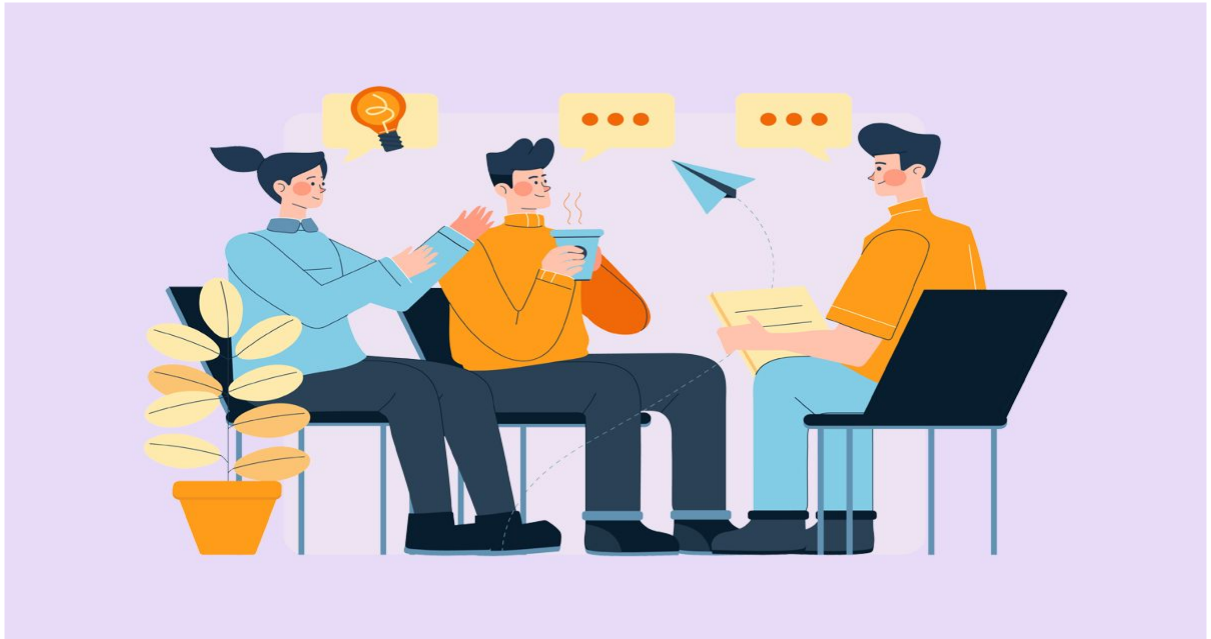
1. Requirement Analysis
2. Software Design
3. Methodologies
4. Testing and Quality Assurance
5. Version Control
6. Documentation
7. Code Reviews

KEY POINTS

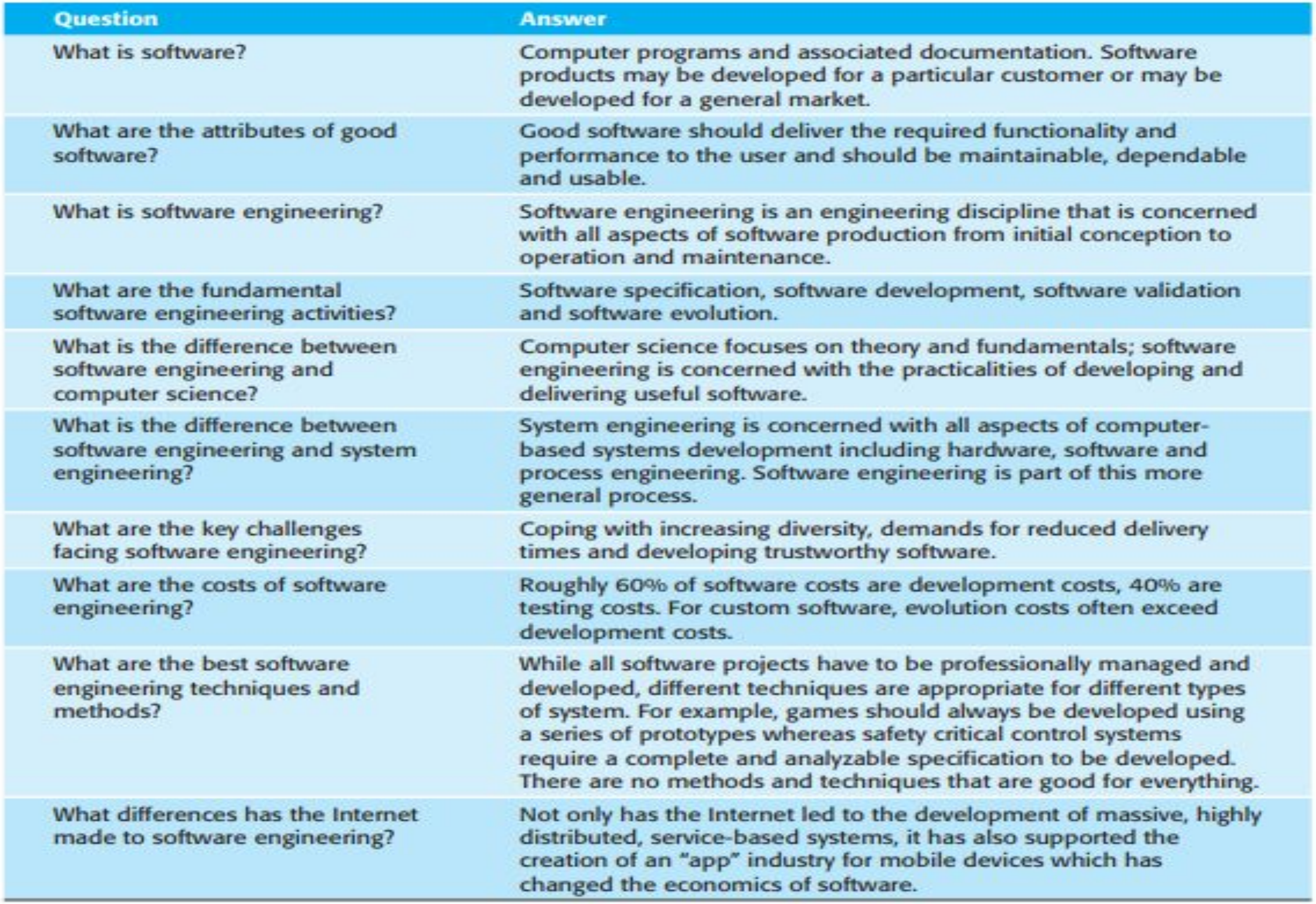
- Software engineering is an engineering discipline that is concerned with all aspects of software production.
- Software is not just a program or programs but also includes documentation. Essential software product attributes are maintainability, dependability, security, efficiency, and acceptability.
- The software process includes all of the activities involved in software development. The high-level activities of specification, development, validation, and evolution are part of all software processes.
- The fundamental notions of software engineering are universally applicable to all types of system development. These fundamentals include software processes, dependability, security, requirements, and reuse.
- There are many different types of systems and each requires appropriate software engineering tools and techniques for their development. There are few, if any, specific design and implementation techniques that are applicable to all kinds of systems.
- The fundamental ideas of software engineering are applicable to all types of software systems. These fundamentals include managed software processes, software dependability and security, requirements engineering, and software reuse.
- Software engineers have responsibilities to the engineering profession and society. They should not simply be concerned with technical issues.
- Professional societies publish codes of conduct that set out the standards of behavior expected of their members.

DISCUSSION

Discuss whether professional engineers should be certified in the same way as doctors or lawyers.



No, software engineers do not necessarily need to be certified in the same way as doctors or lawyers. However, certification may be beneficial in specific, safety-critical fields.



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 from initial conception to operation and maintenance.
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.
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. There are no methods and techniques that are good for everything.
What differences has the Internet made to software engineering?	Not only has the Internet led to the development of massive, highly distributed, service-based systems, it has also supported the creation of an "app" industry for mobile devices which has changed the economics of software.