SE103 INTRODUCTION TO INFORMATION TECHNOLOGY

Software Engineering Module

Prepared by:

Dr. Mohammad Malkawi

Dr. Moh’d Radaideh

Dr. Malik Qasaimeh

Dr. Hamza Al-Kofahi

Department of Software Engineering

Faculty of Computer and Information Technology

Jordan University of Science and Technology

October 29, 2022

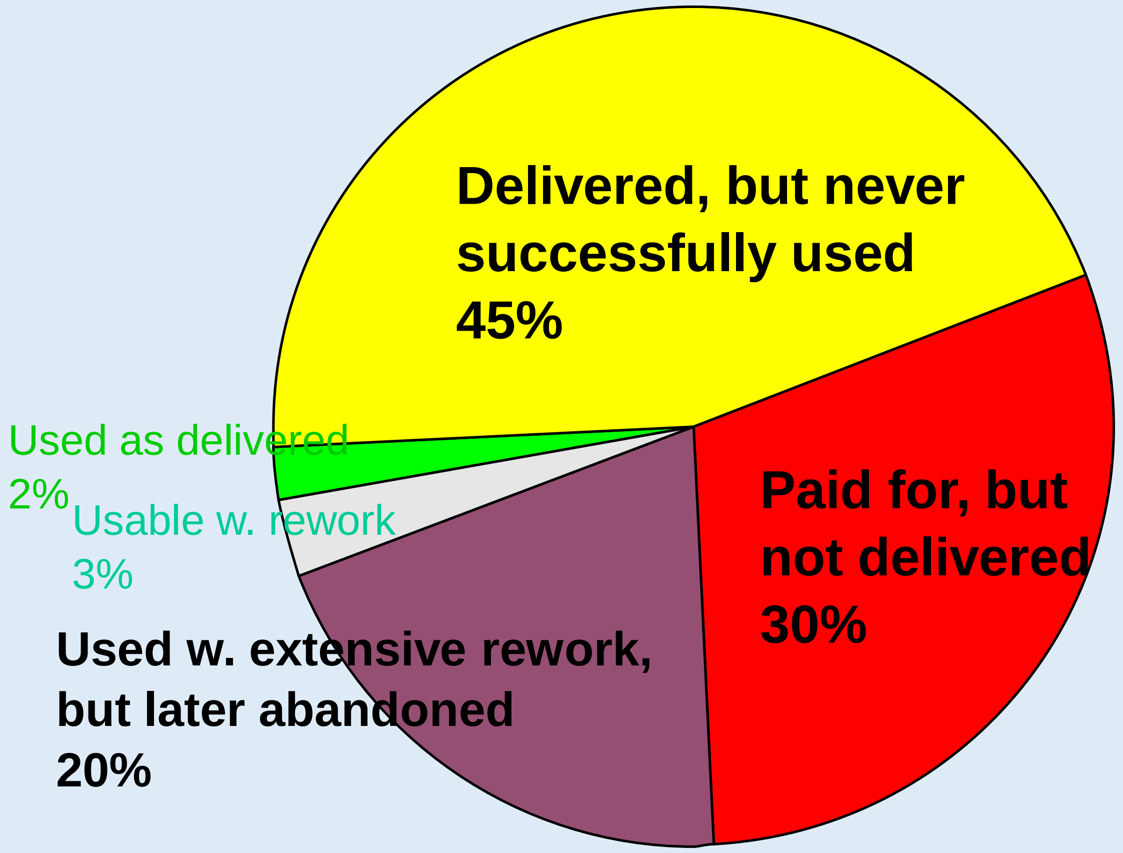
Part #1: Software Engineering Introductory View

Origin of Software Engineering: Software Crisis

- **Software Engineering** was spurred by the so-called Software crisis of the 1960s, 1970s, and 1980s, which identified many of the Problems of Software Development.
  - ✓ Many Projects ran over budget and schedule: The OS/360 Operating System was a classic example.
  - ✓ Some Projects caused property Damage. Software defects can cause property Damage. Poor Software Security allows hackers to steal identities, Costing Time, Money, and reputation.
  - ✓ The Software crisis was originally defined in terms of Productivity but evolved to emphasize Quality.
  - ✓ Some used the term Software crisis to refer to their inability to hire enough qualified Programmers.
  - ✓ Software defects can kill. Some embedded Systems used in radiotherapy machines failed so catastrophically that they administered lethal doses of radiation to patients. The most famous of these failures is the Therac-25 incident.

Why Software Engineering?

Software Projects totaling $96.7 million: Where The Money Went



Scientist vs. Engineer

- **Computer Scientist:**
  - ✓ Proves Theorems about Algorithms, Designs Languages, defines Knowledge representation schemes.
  - ✓ Has Infinite Time.
- **Engineer:**
  - ✓ Develops a solution for an Application-specific Problem for a Client.
  - ✓ Uses Computers & Languages, Tools, Techniques and Methods.
- **Software Engineer:**
  - ✓ Works in Multiple Application Domains.
  - ✓ Has 3 months.

Software Engineering - A Problem Solving Activity

- **Analysis**: Understand the nature of the Problem and break the Problem into pieces.
- **Synthesis**: Put the pieces together into a Large Structure.

For Problem solving we use:

- **Techniques (Methods):**
  - ✓ Formal Procedures for producing Results using some well-defined notation.
- **Methodologies:**
  - ✓ Collection of Techniques applied across Software Development and unified by a Philosophical Approach.
- **Tools:**
  - ✓ Instrument or Automated Systems to accomplish a Technique.

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?	1) Coping with increasing diversity, 2) demands for reduced Delivery Times, and 3) 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 analysable 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.

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.

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.

- **Engineering Discipline**: Using appropriate Theories and Methods to solve Problems bearing in mind Organizational and Financial Constraints.

- **Software Production**: Not just technical Process of Development. Also Project Management and the Development of Tools and Methods to support Software Production.

Importance of Software Engineering

- 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.
- 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 System, the majority of Costs are the Costs of changing the Software after it has gone into use.

General Issues That Affect Most Software

- **Heterogeneity**
  - ✓ Increasingly, Systems are required to operate as distributed Systems across Networks that include different types of Computer and Mobile Devices.
- **Business and Social Change**
  - ✓ Business and Society are changing incredibly quickly as emerging economies develop and new Technologies become available.
  - ✓ They need to be able to change their existing Software and to rapidly develop new Software.
- **Security and Trust**
  - ✓ As Software is intertwined with all Aspects of our lives, it is essential that we can trust that Software.

Part#2: Software Engineering Knowledge Areas

Software Engineering Knowledge Areas (SEKAs)

1- Software Requirements

2- Software Design

3- Software Construction

4- Software Testing

5- Software Maintenance

6- Software Configuration Management

7- Software Engineering Management

8- Software Engineering Process

9- Software Engineering Models and Methods

10- Software Quality

11- Software Engineering Professional Practice

12- Software Engineering Economics

13- Computing Foundation

14- Mathematical Foundation

15- Engineering Foundation

Software Requirements (SEKA#1)

- **A Software Requirement** is a property that must be exhibited by something in order to solve some Problem in the real world.
- **The Software Requirements Knowledge Area** is concerned with:

  1. The elicitation / Analysis / Specification / Validation of Software Requirements, and
  2. The Management of Software Requirements during the whole Software Product Life-Cycle.

Software Requirements (SEKA#1)

Software Requirements Classification

A. Product & Process Requirements

1. **Product Requirements:** Needs and Constraints on the Software (*e.g., the Software shall verify that a student meets all prerequisites before he or she can register for a course*).
2. **Process Requirements:** Constraints on the Software Development Process (*e.g., the Software shall be developed using an agile Process*).

B. Functional and Non-Functional Requirements

1. **Functional Requirements:** Describe the Functions in the Software.
2. **Non-Functional Requirements:** The Requirements that act to Constrain the solution. Also, known as Quality Requirements (*e.g., performance / maintainability / Safety / Reliability / Security / interoperability*).

C. System Requirements & Software Requirements:

1. **A System** is a combination of Components (e.g. Hardware, Software, firmware, people, Information, Techniques, facilities, Services, Etc.) to achieve a pre-defined objective.
2. **System Requirements**: The ones for the System as a whole. In a System containing Software Components, Software Requirements are derived from System Requirements.

D. Emergent Properties (Requirements)

1. Requirements that cannot be addressed by a single Component, but depend on how the Software Components interoperate (*e.g., the throughput Requirement for a call-centre depends on how the telephone System, Information System, and the operators all interact under actual operating conditions*).

E. Quantifiable Requirements

1. Software Requirements should be stated clearly, Unambiguously, and quantitatively.
2. This is particularly important for non-Functional Requirements.

Software Design (SEKA#2)

- **Design** is the Process of defining the architecture, Components, interfaces, and other Characteristics of a System.
- **Software Design** is the Software Engineering Life-Cycle Activity, in which Software Requirements are analysed to produce a description of the Software internal Structure that will serve as the basis for its Construction.

Software Construction (SEKA#3)

- **Software Construction** is the creation of Working Software through a combination of Coding, Verification, Unit Testing, Integration Testing, and Debugging.

Software Testing (SEKA#4)

- **Software Testing** is a dynamic Verification to ensure that the Program under Testing behaves as expected on a finite set of Test Cases, which are selected from an Infinite Execution Domain.

Software Maintenance (SEKA#5)

- ❑ **Software Maintenance** is an Integral part of a Software Life-Cycle, but unfortunately is given less attention in comparison with the Software Development part that has a higher profile than **Software Maintenance** in most Organizations.
- ❑ However, things are now changing as more attention is being paid to the issue of maintaining Software Components that were developed by others.

Software Configuration Management (SEKA#6)

- **Software Configuration Management (SCM)** is a supporting Software Life-Cycle Process that benefits Project Management, Development and Maintenance Activities, and Quality Assurance Activities.

Software Engineering Management (SEKA#7)

- **Software Engineering Management** is all about managing Software Projects and Systems. It is referred to as Software **Project Management** and is composed of 6 Phases:
  1. Initiation and Scope Definition Phase
  2. Software Project Planning Phase
  3. Software Project Enactment Phase
  4. Review and Evaluation Phase
  5. Closure Phase
  6. Software Engineering measurement Phase
- PS., the traditional Project Management Approach is composed of 5 Phases:
  1. Initiation Phase
  2. Planning Phase
  3. Execution Phase
  4. Monitoring and Control Phase
  5. Closure Phase

Software Engineering Process (SEKA#8)

- **Software Engineering Processes** are concerned with Work Activities accomplished by Software Engineers to develop, maintain, and operate Software.
- They include Processes for **Requirements**, **Design**, **Construction**, **Testing**, **configuration Management**, and others.

Software Engineering Models and Methods (SEKA#9)

- **Software Engineering Models and Methods** impose Structure on Software Engineering to make it Systematic, Repeatable, and more Success-Oriented.
- **Models** provide an Approach to Problem-Solving, a Notation, and Procedures for Models Construction and Analysis.
- **Methods** provide an Approach to the Systematic Specification, Design, Construction, Test, and Verification of the end-item Software and its associated Work Products.

Software Quality (SEKA#10)

- **Software Quality** is concerned with the Software Engineering perspective of:
  1. Culture and Ethics,
  2. Value and Cost of Quality,
  3. Quality Models and Quality Characteristics,
  4. Quality Improvement,
  5. Software Safety,
  6. Software Quality Assurance,
  7. Software Quality Control,
  8. Software and Systems Verification and Validation,
  9. Software Reviews and Audits,
  10. Software Quality Requirements,
  11. Software defects Characterization,
  12. Software Quality Management Techniques,
  13. Software Quality Measurements, and
  14. Software Quality Tools.

Software Engineering Professional Practice (SEKA#11)

Professionalism	Group of Dynamics and Psychology	Communication Skills
1. Accreditation, Certification, and licensing, 2. Codes of Ethics and Professional Conduct, 3. Nature and Role of Professional Societies, 4. Economic Impact of Software, 5. Employment Contracts, 6. Legal Issues, 7. Documentation, and 8. Tradeoff Analysis.	1. Dynamics of Working in Teams, 2. Individual Cognition, 3. Dealing with Problem Complexity, 4. Interacting with Stakeholders, 5. Dealing with Uncertainty and Ambiguity, and 6. Dealing with Multicultural Environments.	1. Reading / Understanding / Summarizing Skills, 2. Writing Skills, 3. Team & Group Communication Skills, and 4. Presentation Skills.

Software Engineering Economics (SEKA#12)

- **Software Engineering Economics** is about making Business Decisions related to Software Engineering. It is concerned with aligning Software technical Decisions with the Business goals of the Organization.
- **Economics** is the study of value, Costs, Resources, and their Relationship in a given Situation.
- **SWE Activities** have Costs, but the resulting Software has Economic Attributes.
- **SWE Economics** involves Software Finance, Accounting, Controlling, Cash Flow, Valuation, Inflation, Depreciation, Taxation, Time-Value of Money, Efficiency, Effectiveness, and Productivity. It also involves the entire Life-Cycle of the Product, Project, Program, and Portfolio.

Computing Foundations (SEKA#13)

- **Computing Foundation** includes the following subjects:
  1. Problem Solving,
  2. Abstraction,
  3. Programming Fundamentals,
  4. Programming Basics,
  5. Debugging Tools and Techniques,
  6. Data Structure and Representation,
  7. Algorithms and Complexity,
  8. Basic Concept of Systems,
  9. Computer Organization,
  10. Operating Systems,
  11. Compiler Basics,
  12. Database Basics and Data Management,
  13. Network Communication Basics,
  14. Parallel Distributed Computing,
  15. Basic User Human Factors,
  16. Basic Developer Human Factors, and
  17. Secure Software Development and Maintenance.

Mathematical Foundations (SEKA#14)

- **Mathematical Foundations** include the following subjects:

  1. Sets / Relations / Functions,
  2. Proof Techniques,
  3. Graphs, and Trees,
  4. Finite State-Machines,
  5. Numerical Precision / Accuracy / Errors,
  6. Algebraic Structures,
  7. Basic Logic,
  8. Basics of Counting,
  9. Discrete Probability,
  10. Grammars, and
  11. Number Theory.

Engineering Foundations (SEKA#15)

- **Engineering Foundations** include the following subjects:
  1. Empirical Methods and Experimental Techniques,
  2. Statistical Analysis,
  3. Measurement,
  4. Engineering Design,
  5. Modeling, Simulation, and Prototyping,
  6. Standards, and
  7. Root cause Analysis.

Part #3: Software Engineering Job Opportunities, Careers and Applications

BECOMING A SOFTWARE ENGINEER

- What is a Software Engineer?
- How does the Software Engineer Role differ from the Software Developer Role?
- What is the job outlook of Software Engineers?
- What is the Average Salary for Software Engineers?
- What are the ways to Advance in the Software Engineering career?

BECOMING A SOFTWARE ENGINEER

What is a Software Engineer?

- A Software Engineer is a Computer Science Professional.
- Applies Engineering Principles and their Knowledge of Programming Languages to the Design Development, Testing, Evaluation and Maintenance of Computer Software.
- Have extensive Knowledge of Computer Operating Systems, Software Development and Programming Languages.
- Apply Engineering Principles at every stage during the Development Process to create customized Software Systems for Clients.

BECOMING A SOFTWARE ENGINEER

Software Engineers vs. Developers

- There's an overlap in some of the Responsibilities of Software Engineers and Software Developers.
- Some Employers use the two terms interchangeably. But there are some key differences, including:
  - ✓ Education,
  - ✓ Focus,
  - ✓ Skills, and
  - ✓ Salary

BECOMING A SOFTWARE ENGINEER
Software Engineers vs. Developers: Education

- ❑ Both careers have limited options for Professionals without a degree.
- ❑ A minimum of a bachelor's degree in Computer Science or a related field is common for SE and CS for Developers.
- ❑ Due to the increased demand for Software Engineers, an aspiring Engineer may also pursue an Advanced degree (M.Sc., Ph.D.).

BECOMING A SOFTWARE ENGINEER
Software Engineers vs. Developers: Focus

- ❑ A major distinguishing Factor between the two career paths is the type of Work they commonly complete.
- ❑ Software Engineers are more likely to spend Time Working on Backend Software and solutions.
- ❑ While Developers often focus on Front-End Operation and the Client Experience.

BECOMING A SOFTWARE ENGINEER
Software Engineers vs. Developers: Focus

- ❑ A major distinguishing Factor between the two career paths is the type of Work they commonly complete.
- ❑ Software Engineers are more likely to spend Time Working on Backend Software and solutions
- ❑ While Developers often focus on Front-End Operation and the Client Experience.

BECOMING A SOFTWARE ENGINEER
Software Engineers vs. Developers: Skills

- Many Skills that Developers have also apply to Software Engineers. However, there are some key differences:
  - ✓ Software Developers often require higher levels of Interpersonal Skills
  - ✓ Because of their increased Interaction with Clients or Collaborators.
  - ✓ Software Engineers have the Skills of Design and Backend Capabilities

BECOMING A SOFTWARE ENGINEER

Software Engineers vs. Developers: Salary

- ❑ Compensation in both fields varies based on several Factors, including Experience, Professional Qualifications, Employer Size and Location.
- ❑ Generally, Software Engineers earn more than Developers, who make $78,061 per year to account for the increased Professional Requirements and Expectations.

BECOMING A SOFTWARE ENGINEER
<u>Job Outlook for Software Engineers</u>

- ❑ Software Engineers fit within a broader Category of Computer Development Professions in the United States Bureau of Labor Statistics (BLS) data.
- ❑ The BLS expects employment for Software Developers, Quality Assurance Analysts and Testers to grow by 25% from 2021 to 2031.
- ❑ This represents a significantly higher Projected Growth than the Average for all Occupations.

REASONS FOR THE DEMAND FOR SOFTWARE ENGINEERS

Need for Innovative Software

- ❑ As Technology becomes more important for Businesses across many Industries, the need for innovative Software and Engineers to develop it grows as well.
- ❑ When a Company identifies an Area for Improvement, it may hire Software Engineers to help in the Development of Systems and Software to take advantage of the Opportunity.
- ❑ This creates a broad range of career options for individuals trained as Software Engineers.
- ❑ Read more: 10 Business Plan Software Systems (With Pros and Cons) https://www.indeed.com/career-advice/career-Development/Business-plan-Software

REASONS FOR THE DEMAND FOR SOFTWARE ENGINEERS

Limited Life Span of Code

- Another important consideration for Companies relying on Software for their daily Operations is the need to keep their Code updated.
- This helps to maximize the Efficiency and capability of Work Software.
- It's also important to keep Software updated for Security concerns, as it allows a Company to respond to new threats and provide the required protection.
- Read more: What Is Patch Management and How Does It Work? https://www.indeed.com/career-advice/career-Development/patch-Management

REASONS FOR THE DEMAND FOR SOFTWARE ENGINEERS

Accelerated Growth in Technology

- The modern Technological field continues to expand rapidly.
- Companies Working in Industries that previously used minimal Technological conveniences may benefit from modern Technology in a variety of ways.
- This expanded demand for Technology creates a demand for Professionals who are capable of creating and maintaining it, such as Software Engineers.
- Read more: 10 Inventory Management Software Options (With Features) https://www.indeed.com/career-advice/career-Development/inventory-Management-Software

REASONS FOR THE DEMAND FOR SOFTWARE ENGINEERS

Increased Complexity of Projects

- As Technology grows more Advanced, it often becomes more Complex as well.
- This further adds to the value and demand for Software Engineering Professionals.
- More Complex Technology requires Employees with more Complex Training, such as the Specialized Training of a Software Engineer.

REASONS FOR THE DEMAND FOR SOFTWARE ENGINEERS

Software Engineer Salary

- ❑ The Average annual Salary for a Software Engineer is $93,959 per year.
- ❑ Senior Software Engineers make an Average of $115,741 per year.
- ❑ Some of the primary Factors that Impact Salary for Software Engineers include years of Experience, Educational level, Specialization and Geographic Location.
- ❑ Among the highest-paying cities for Software Engineers are San Francisco, San Jose, New York City, Seattle and Chicago.
- ❑ 10 Best Cities for Software Engineers https://www.indeed.com/career-advice/finding-a-job/best-city-for-Software-Engineers

HOW TO ADVANCE IN YOUR CAREER AS A SOFTWARE ENGINEER?

- ❑ Here are some steps you can take to Advance your career as a Software Developer:
  - ✓ Practice Coding.
  - ✓ Get Experience at Large Companies.
  - ✓ Develop your Soft-Skills.
  - ✓ Pursue Leadership Experience.

HOW TO ADVANCE IN YOUR CAREER AS A SOFTWARE ENGINEER?

Practice Coding

- You can learn to Code through Practical Experience.
- Completing Project prompts, either of your own creation or that you find in other sources, presents you with Software challenges to complete.
- This can allow you to develop new Skills, Troubleshoot Problems to learn new Information and build a Professional Portfolio.
- Related: 12 Elements of a Career Portfolio https://www.indeed.com/career-advice/resumes-cover-letters/career-Portfolio

HOW TO ADVANCE IN YOUR CAREER AS A SOFTWARE ENGINEER?

<u>Get Experience at Large Companies</u>

- ❑ Consider applying for an entry-level Position as a Programmer, Developer or related Role.
- ❑ Companies commonly look for Experienced Professionals to fill their Software Engineer Positions, and these entry-level jobs often list lower Work and Education Requirements.
- ❑ This allows you to gain Practical Experience, which can enable you to seek more Advanced Positions in the future.
- ❑ Related: 8 Entry-Level Software Developer Jobs (With Typical Duties) https://www.indeed.com/career-advice/finding-a-job/entry-level-Software-Developer-job

HOW TO ADVANCE IN YOUR CAREER AS A SOFTWARE ENGINEER?

Develop Your Soft-Skills

- ❑ It's important to build your Professional Skills to become a well-rounded Employee.
- ❑ Besides having highly technical Skills, having strong Soft-Skills such as Communication, Organization and Teamwork can be Valuable as a Software Engineer.
- ❑ By Working on your Skill set and improving it, you may make yourself a more appealing Employee to a potential Employer.
- ❑ Related: 10 Important Skills for Computer Engineers https://www.indeed.com/career-advice/career-Development/Computer-Engineer-Skills

HOW TO ADVANCE IN YOUR CAREER AS A SOFTWARE ENGINEER?

Pursue Leadership Experience

- ❑ You may request the Opportunity to take Leadership Roles on Projects.
- ❑ Doing so allows you to develop your Skills as a Team leader and may show Professional initiative to potential Employers.
- ❑ This can be Valuable when you're seeking Advancement into a Position such as a Software Engineer. https://www.indeed.com/career-advice/resumes-cover-letters/Leadership-Skills

TIPS FOR WORKING AS A SOFTWARE ENGINEER

- If you're interested in a career as a Software Engineer, keep these tips in mind:
  - ✓ Follow your Passion
  - ✓ Keep your Skills Updated
  - ✓ Build your Professional Network
  - ✓ Get Certified

TIPS FOR WORKING AS A SOFTWARE ENGINEER

Follow Your Passion

- ❑ Software Engineers have the potential to Work across a broad range of Industries.
- ❑ Thus, this Role provides significant Freedom to choose a field you're passionate about.

TIPS FOR WORKING AS A SOFTWARE ENGINEER

Keep Your Skills Updated

- ❑ Just as the Technological field is constantly changing, it's important for Professionals within it to build on and develop their Skills.
- ❑ Remaining updated on the latest Developments and keeping your Programming and Engineering Skills strong can be beneficial as a Software Engineer.

TIPS FOR WORKING AS A SOFTWARE ENGINEER

Build Your Professional Network

- ❑ Your Network may help you find an Opportunity within their Organization.
- ❑ Serve as a reference when you're seeking an Opportunity with another Employer or help you with Issues you encounter during your Work.

TIPS FOR WORKING AS A SOFTWARE ENGINEER

Get Certified

- Certification shows a potential Employer that you've completed Advanced study on a Topic, such as a Programming Language or Style of Software Engineering.
- This might make you a more appealing Candidate for a Position.

Software Engineering Applications 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 & manage Hardware Devices. There are more embedded Systems than any other type of System.
- **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.
- **Modelling and Simulation Systems:** These are Systems that are developed by scientists and Engineers to Model Physical Processes or Situations, which include many, separate, interacting objects.
- **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.

Part #4: Software Engineering Certifications

Software Development International Certifications

- ❑ **Amazon Web Services (AWS®) Certification**:
  - ✓ AWS Certified DevOps Engineer – Professional
- ❑ **IEEE Computer Society:**
  - ✓ The Certified Software Development Professional (CSDP)
- ❑ **National Initiative for Cybersecurity Careers and Studies(NICCS®)**:
  - ✓ Certified Secure Software Lifecycle Professional (CSSLP®)

Software Testing International Certifications

- **International Software Testing Qualifications Board(ISTQB®)**:
  - ✓ **Core Foundation (level 1):**
    1. Certified Tester Foundation Level (CTFL)
  - ✓ **Core Advanced (level 2):**
    1. Certified Tester Advanced Level Test Analyst (CTAL-TA)
    2. Certified Tester Advanced Level Test Manager (CTAL-TM)
  - ✓ **Specialist (Level 3):**
    1. Certified Tester AI Testing (CT-AI)
    2. Certified Tester Game Testing (CT-GaMe)

Software Security International Certifications

- **The International Council of Electronic Commerce Consultants (EC-Council):**
  - ✓ Certified Ethical Hacker (CEH): a Qualification given by EC-Council and obtained by demonstrating Knowledge of Assessing the Security of Computer Systems by looking for Weaknesses and Vulnerabilities in Target Systems.
- **The International Information System Security Certification Consortium (ISC):**
  - ✓ Certified Information Systems Security Professional (CISSP)

Project Management and Software Project Management International Certifications

- **Project Management Institute (PMI):**

1. Project Management Professional (PMP)
2. Certified Associate in Project Management (CAPM)
3. PMI Agile Certified Practitioner (PMI-ACP)
4. Professional in Business Analysis (PMI-PBA)
5. Program Management Professional (PgMP)
6. Portfolio Management Professional (PfMP)
7. PMI Risk Management Professional (PMI-RMP)
8. PMI Scheduling Professional (PMI-SP)
9. PMI Project Management Ready
10. Disciplined Agile Certifications:
    a. Disciplined Agile Scrum Master (DASM)
    b. Disciplined Agile Senior Scrum Master (DASSM)
    c. Disciplined Agile Coach (DAC)
    d. Disciplined Agile Value Stream Consultant (DAVSC)

Project Management and Software Project Management International Certifications

❑ SCRUM.ORG Certifications:

1. Professional Scrum Master I (PSM I)
2. Professional Scrum Master II (PSM II)
3. Professional Scrum Master III (PSM III)
4. Professional Scrum Product Owner I (PSPO I)
5. Professional Scrum Product Owner II (PSPO II)
6. Professional Scrum Product Owner III (PSPO III)
7. Professional Scrum Development (PSD)
8. Professional Agile Leadership (PAL)
9. Professional Agile Leadership – Evidence-Based Management (PAL EBM)
10. Scaled Professional Scrum (SPS)
11. Professional Scrum with Kanban (PSK)
12. Professional Scrum with User Experience (PSU)