Assignment: Introduction to Software Engineering

Instructions:

Answer the following questions based on your understanding of software engineering concepts. Provide detailed explanations and examples where appropriate.

Questions:

Define Software Engineering:

It’s a discipline that aims to produce quality software development, software delivered in time, within the budget and that satisfies its requirement and the customer.

What is software engineering, and how does it differ from traditional programming?

Software Development Life Cycle (SDLC):

All Software engineering methodologies are alike – they have similar phases and similar artifacts – but there are many small differences. software engineering methodologies tend not to be too prescriptive: the developers are given some choice about whether they use a particular type of diagram. The benefits of software engineering Development are reduced time to market, greater product flexibility, and schedule predictability and the risks of them are performance and start-up costs. The software engineering phases are

Analysis

Construction

Testing

Uses UML and related diagrams– Unified Modelling Language (Use Case Diagram, Class diagram, sequence diagram and etc.)

There are a number of phases common to every development, regardless of methodology, starting with requirements capture and ending with maintenance. With the traditional approach, will be expected to move forward gracefully from one phase to the other. The list below describes the common phases in software development

Requirement

Analysis

Design

Specification

Implementation

Testing

Deployment and

Maintenance

Explain the various phases of the Software Development Life Cycle. Provide a brief description of each phase.

System Development Life Cycle is an approach developed and specifically used for system development and Information Systems. SDLC involves phases or stages that needs to be sequentially followed until the last step and these steps could be done in a circular approach until the system development project its done.

PROJECT INVISTIGATION:

System Investigation is the first phase in the traditional SDLC. The purpose is to identify potential problems and opportunities and consider them in light of goals of the company. The main purpose is to uncover answers to points like problems of the system, resources used to develop the system, feasibility, costs and time frames.

SYSTEM ANALYSIS:

The purpose of analysis is to gather information/data of the system, determining the requirements of for the new system, considering alternatives constrains and investigating further the feasibility and solutions

SYSTEM DESIGN:

The primary goal of the system design is to deep dive into the technical design that details system inputs, processes and outputs, interfaces, hardware, software, telecommunications, databases, personnel and procedures and how they relate to them.

SYSTEM IMPLEMENTATION:

This is the stage of system development that includes hardware acquisitions, software acquisitions and development, user preparation, hiring and training of personnel, site and preparation, installations, testing, start-up and user acceptance.

SYSTEM MAINTENANCE AND REVIEW:

This stage involves checking, changing and the enhancing the system to make it more useful in achieving users and organisational goal. It ensures the system operates as intended and modifies the system so that it continues to meet changing business needs.

Agile vs. Waterfall Models:

Compare and contrast the Agile and Waterfall models of software development. What are the key differences, and in what scenarios might each be preferred?

Waterfall methodology. Its primary benefits are as follows: It is a sequential, linear development approach. It is simple to use, provides a thorough description of the requirements, emphasizes the need of documentation, and can easily get past obstacles, but it is rigid and only produces partial deliverables. According to Winston W. Royce in 1970, he himself had introduced it as an example of a flawed software development method, that's vulnerable due to its many shortcomings (McCormick, M., 2021). The Waterfall Model was the first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. each phase must be completed before the next phase can begin and there is no overlapping in the phases (Ateeq, S.A.D.A.F. and Shuaib, M., 2014). The requirement specification, conceptualization, analysis, design, coding, testing and debugging, installation, and maintenance phases of software development are outlined in this model. The development team only moves on to the next step of development in this sequentially organized method when the preceding stage has been completed.

Agile refers to rapid movement. This software development methodology, often known as a lightweight and flexible model in software development, is founded on the incremental and iterative approaches of SDMs. According to Cohen et al., “Agile techniques vary in practices and emphasis, they share common characteristics, including iterative development and a focus on interaction, communication, and the reduction of resource-intensive intermediate artifacts”( Van Casteren, W., 2017:2). Customer happiness, adaptability, conducting product tests to guarantee quality, the importance of meeting deadlines, and producing high-quality products are the guiding elements of this approach. Scrum approach is presented by Agile Manifesto as a subset model. Its tactics include sprints, the product backlog, scrum (communication, feedback, and collaboration), and prioritizing features, enhancements, and defect resolution.

Requirements Engineering:

What is requirements engineering? Describe the process and its importance in the software development lifecycle.

Software Design Principles:

Requirement Engineering is a systematic process of developing requirements through an iteration co-operative process of analysing the problem, documenting the resulting observations in a variety of representation formats, and checking the accuracy of understanding gained (Macualay, L.A, 2012).

Macualay, L.A, (2012) continues to explain that the fundamental importance of Requirement Engineering is “concern with what needs to be designed rather than how it is to be designed, also concern with with future sisturations like input to systems, designs shown as the user’s present job, and technological options and output shown as the future system.

Requirement Engineering describes two types of process. First is the problem analysis, where the analyst, project manager and the team spends time brainstorming, interviewing experts on the subject and identifying constrains of the problem

Second activity is product description, this includes the research and analysis on what product, feasibilities and choice of option modelling and requirement documentations.

Explain the concept of modularity in software design. How does it improve maintainability and scalability of software systems?

Testing in Software Engineering:

Dennis, J.B., (1975), says modular means “ constructing with standardized units or dimensions for flexibility and variety in use. Applied to software engineering, modularity referrers to the building of software systems by putting together parts called program modules.” Modularity is one of the approaches to achieve frameworks and techniques that may help in downsizing knowledge. it denotes the possibility to perceive large knowledge repositories (be it an ontology or database) as a set of modules. It is used in computer science disciplines like algorithms, software engineering and programming languages (Parent, C., and Spaccapietra, S., 2009). Modular’ improves maintainability and scalability by performing tasks intended to interact with other modules in a large software architecture, representation of encapsulation, easiness of understanding and potential for reuse of approaches.

Describe the different levels of software testing (unit testing, integration testing, system testing, acceptance testing). Why is testing crucial in software development?

Version Control Systems:

UNIT TESTING

Unit testing, also known as module testing. Unit testing, tests individual programs. Module or component testing is important to identify and remove faults that otherwise can be difficult to identify and expensive to remove later in the development phase. It includes structural and functional testing.

Structural testing can be derived from module specifications and using appropriate functions and module-based testing techniques. Functional testing is complemented by structural testing by covering the code and coping with cases not included in functional testing.

INTERGRATION TESTING

Integration faults are caused by incomplete specifications or faulty implementation of interface, resource usage, or required properties. Integration testing is the quality of single module necessary but not sufficient enough to guarantee the quality of the final system. The failure of low quality module leads to system failures that are often difficult to diagnose, hard and expensive to remove (Barezzi, L., and Pezze, M., 2006).

SYSTEM AND ACCEPTANCE TESTING

System and acceptance testing consider the behaviour of the overall system functionalities and non-functionalities aspects. Module and integration testing are based on the internal of the software, which are hardly accessible to the users. They focus on verification activities that provides useful results without requiring the deployment of the whole system and the presence of the user. It’s activities forces on functional properties. Some non-functional properties, like modularity, maintainability and testability can be enforced through design rules and check with simple static analysis tools during development and but not extensive testing (Barezzi, L., and Pezze, M., 2006).

System testing verifies the correspondence between the overall systems and specifications, while acceptance testing verifies correspondence between the system and the user’s expectations(Barezzi, L., and Pezze, M., 2006)..

What are version control systems, and why are they important in software development? Give examples of popular version control systems and their features.

Version control (also known as revision control or source control) is the management of multiple revisions of the same unit of information. It is most commonly used to manage ongoing development of digital documents like source code, art resources or electronic models and other critical information that may be worked on (Milentijevic, I., Ciric, V. and Vojinovic, O., 2008) Version control system (VCS) is a software application that helps in collaborative software development of software projects.

Centralized and distributed version control systems are the two most widely used varieties. Distributed version control systems store files across several repositories, whereas centralized version control systems store all files in a single repository.

Software Project Management:

The Software Project Management are tools applied to a large family of tools whose primary purpose is sophisticated scheduling for projects with hundreds or even thousands of overlapping and partially interdependent tasks. These tools are able to drop down to very detailed task levels, and can even handle the schedules of individual workers.

Discuss the role of a software project manager. What are some key responsibilities and challenges faced in managing software projects?

Software Maintenance:

However, the family of project management tools is general purpose in nature and does not include specialized software sizing and estimating capabilities as do the software cost estimating tools. Neither do these general project management tools deal with quality issues such as defect removal efficiency. Project management tools are useful, but software requires additional capabilities to be under full management control.

Example of Software Project Management tool:

CHECKPOINT

Constructive Cost Model (COCOMO)

CostXpert,

GANTT CHART

PERT

Parametric Review of Information for Costing and Evaluation – Software (PRICE-S)

Software Evaluation and Estimation of Resources – Software Estimating Model (SEER-SEM) and

Software Life Cycle Management (SLIM).

Thus any new project can be compared against similar projects to judge the validity of schedules, costs, quality, and other important factors. The most useful measurements for projects in the 10,000- function point domain include measures of the following:

• Accumulated effort.

• Accumulated costs.

• Development productivity.

• Volume and rate of requirements changes.

• Defects by origin.

• Defect removal efficiency.

Measures of effort should be granular enough to support work breakdown structures. Cost measures should be complete and include development costs, contract costs, and costs associated with purchasing or leasing packages.

However bulk of schedule delays and cost overruns tends to occur during testing and is caused by excessive defect volumes, it can be hypothesized that lack of effective quality control on large systems is a major contributor to both cost and schedule overruns.

Define software maintenance and explain the different types of maintenance activities.

Software maintenance involves checking, changing and enhancing systems to make it more useful in achieving the user and organisational goals.

Maintenance Activities

**Slipstream upgrade**: a minor upgrade, typically a code adjustment or fixing bugs and defects.

**Patches**: Change or correction of a problem or system enhancement.

**New Release:** it’s a significant program or system change that require changes in the documentation of the software.

**New version**: it’s a major program change, encompassing many features to the original version of the software.

Why is maintenance an essential part of the software lifecycle?

Maintenance is important for users, groups, organisation and they have dedicated teams for maintaining. After software development and implementation, the software or system will need maintenance. Maintenance prevent system failures.

Ethical Considerations in Software Engineering:

What are some ethical issues that software engineers might face?

The practicing software engineer engages in a social process during software production and thereby acquires obligations to: users, clients, customers, colleagues, supervisors, the organization for which he works, and the discipline of software engineering. The ethical problems faced by the software engineer involve: the end product, the process of developing that product, and the human interactions in the development of the product. Ethics in Software Engineering includes every conceivable misuse of computers and the general discipline. It’s the illegal or malicious acts one can engage in with computers. These acts are sometimes described as conundrums and leave one with the impression that there can be no progress in ethics.

These principles should influence you to consider broadly who is affected by your work; to examine if you and your colleagues are treating other human beings with due respect; to speculate on how the public would view your decision if they were reasonably well informed; to analyse how the least empowered will be affected by your decision; and to consider whether your acts would be judged worthy of the ideal professional working as a software engineer.

How can software engineers ensure they adhere to ethical standards in their work?

Product

Software engineers shall, insofar as possible, assure that the software on which they work is useful and of acceptable quality to the public, the employer, the client, and the user, completed on time and at reasonable cost, and free of error.

Public

Software engineering shall, in their professional role, act only in ways consistent with the public safety, health, and welfare. In particular Be fair and truthful in all statements, particularly public ones, concerning software or related documents. Not put self-interest, the interest of an employer, the interest of a client, or the interest of the user ahead of the public’s interest and Accept full responsibility for their own work.

Profession

Software engineers shall, in all professional matters, advance both the integrity and reputation of their profession as is consistent with public health, safety, and welfare. In particular Ensure that clients, employers, and supervisors know of the software engineer’s commitment to this code of ethics, and their own responsibility under it. Take responsibility for detecting, correcting, and reporting errors in software and associated documents on which they work. Obey all laws governing their work, insofar as such obedience is consistent with the public health, safety, and welfare. Exercise professional responsibility to society by constructively serving in civic affairs.

Self

Software engineers shall, throughout their careers, strive to enhance their own abilities to practice their profession as it should be practiced. In particular Further their knowledge of developments in the design, development, and testing of software and related documents, together with the management of the development process. Improve their understanding of the software and related documents on which they work and of the environment in which they will be used. Improve their knowledge of this code, its interpretation, and its application to their work.

Jones C. Software project management practices: Failure versus success. CrossTalk: The Journal of Defense Software Engineering. 2004 Oct;17(10):5-9.

Gotterbarn, D., 1991, May. Ethical considerations in software engineering. In *Proceedings of the 13th international conference on Software engineering* (pp. 266-274).

Gotterbarn, D., Miller, K. and Rogerson, S., 1997. Software engineering code of ethics. *Communications of the ACM*, *40*(11), pp.110-118.

Munassar, N.M.A. and Govardhan, A., 2011. Comparison between traditional approach and object-oriented approach in software engineering development. *International Journal of Advanced Computer Science and Applications*, *2*(6).

Submission Guidelines:

Your answers should be well-structured, concise, and to the point.

Provide real-world examples or case studies wherever possible.

Cite any references or sources you use in your answers.

Submit your completed assignment by [due date].