|  |
| --- |
| **Introduction to Software Engineering (SE1001)** |
| Date: April 3rd 2024 |
| **Course Instructor(s)** |
| Ms. Kiran Khurshid |
| **Sessional-II Exam** | |
| **Total Time (Hrs):** | **1** |
| **Total Marks:** | **60** |
| **Total Questions:** | **9** |

|  |
| --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Roll No Section Student Signature |

**Do not write below this line**

**Attempt all the questions on the provided answer sheets. No need to copy the questions on the answer sheet. Just write the question number and write its answer.**

***CLO #3: Discuss key Software Engineering Principles***

**Q1:** Which of the following is a process quality and which is a product quality? [10 marks]

1. Reliability Product
2. Understandability Product
3. Correctness Product
4. Visibility Process
5. Evolvability Product
6. Timeliness Process
7. Verifiability Product
8. Robustness Product
9. Repairability Product
10. Maintainability Product

***CLO #3: Discuss key Software Engineering Principles***

**Q2:** Legacy systems are difficult to maintain and modify. Write down two techniques that are aimed at uncovering the structure of such systems and restructuring to improve them. Explain each technique in one sentence only. [4 marks]

**Reverse engineering** and **reengineering techniques and technologies** are aimed at uncovering the structure of legacy software and restructuring or in some way improving it.

Reengineering is the process through which an existing system undergoes an

alteration, to be reconstituted in a new form. Usually, the process includes two distinct

phases. In one phase, the software engineer proceeds backwards, from the existing system to some abstract representation that allows him or her to understand exactly the

structure of the existing system and how to modify it. Such a phase is often called

reverse engineering (or refactoring). In the other phase, the engineer proceeds forward

and actually designs and applies the necessary changes. Reverse engineering includes

program understanding, which means that the software engineer seeks to understand

the structure of the program, its main algorithms, and data structures and tries to isolate the main components that need to be restructured. identify the main system components and. their relationships and needs to create abstract representations of the system that aid in understanding it more precisely. This task is, of course, facilitated if the documentation is complete and is consistent with the actual implementation. Very often, however, this is not true, and the engineer needs to

go through the painful process of recovering the design from the code and reconstructing abstract representations from low-level implementations.

***CLO #3: Discuss key Software Engineering Principles***

**Q3:** Choose ONE correct/best suited option, unless specified otherwise. Write your answers on the provided answer sheets. [9 marks]

1) The reactive modification of software to repair problems discovered after the software has been delivered to a customer’s end user is called:

1. Corrective maintenance
2. Adaptive maintenance
3. Preventive maintenance
4. Perfective maintenance
5. None of the above

2) The reactive modification of software after delivery to keep the software usable in a changing end-user environment is called:

1. Corrective maintenance
2. Adaptive maintenance
3. Preventive maintenance
4. Perfective maintenance
5. None of the above

3) The proactive modification of the software after delivery to provide new user features, better program code structure, or improved documentation is called:

1. Corrective maintenance
2. Adaptive maintenance
3. Preventive maintenance
4. Perfective maintenance
5. None of the above

4) *Separation of concerns* rule is also called:

1. Consistency
2. Abstraction
3. Divide and Conquer
4. Transfer of information
5. None of the above

5) Software engineers use \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ as a means of cataloging and reusing solutions to problems they have encountered in the past.

1. Modules
2. Databases
3. Patterns
4. Color schemes
5. User interfaces

6) The software engineering principles state that when possible, examine the \_\_\_\_\_\_\_\_\_\_\_\_ and \_\_\_\_\_\_\_\_\_\_\_\_from several different perspectives:

1. Software, hardware
2. Module, phases
3. Design, code
4. Code, testcases
5. Problem, solution

7) The principle of consistency applies to which of the following?

1. Requirements specification
2. Software design
3. Source code
4. Test cases
5. All of the above

8) Which of the following is true?

1. A large problem is easier to solve if it is subdivided into a collection of elements
2. A large problem becomes difficult to solve if it is subdivided into a collection of elements
3. In *divide and conquer* tasks are broken down and then kept separate to conquer them
4. The development cost should be divided among all the stakeholders to conquer the process
5. None of the above

9) Which of the following is/are the type(s) of maintainability (you may choose more than one option in this question):

1. Interoperability
2. Portability
3. Repairability
4. Evolvability
5. Usability

***CLO #3: Discuss key Software Engineering Principles***

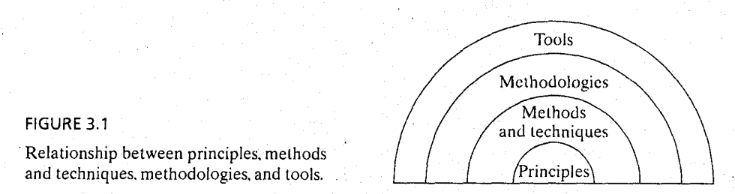
**Q4: a)** What is the difference between principles, methods/techniques, methodologies and tools**.** Draw a diagram to show their relationships. [4+2 marks]

Software engineering principles are the fundamental beliefs or guidelines guiding behavior, decision-making, and actions.

Methods are high-level approaches or strategies used to accomplish a task or goal.

Sometimes, methods and techniques are packaged together to form a methodology.

Tools, in turn, are developed to support the application of techniques, methods, and methodologies



**b)** Which of the following is an example each of principle, method, methodology and tool: [4 marks]

“Jira” – tool

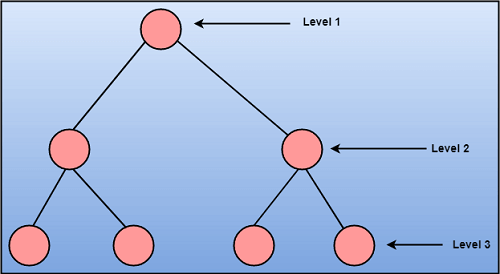
“Kanban” - methodology,

“Abstraction” - principle,

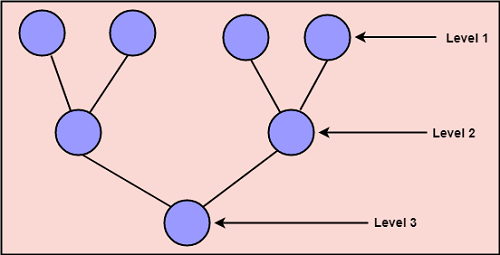
“Agile” - method

***CLO #3: Discuss key Software Engineering Principles***

**Q5:** Draw the hierarchical diagram to explain the top-down and bottom-up approaches in modularity. [4 marks]



*Figure 1 Top down*



*Figure 2 Bottom up*

***CLO #4: Describe different phases of software development***

**Q6:** What is the difference between functional and nonfunctional requirements? Give an example of each. [4 marks]

Functional requirements specify/document:

* What the system will do
* The functionality that the system will provide
* The behaviour of the system
* What the user requires

A nonfunctional requirement (NFR) can be described as a quality attribute, a performance attribute, a security attribute, or a general constraint on a system.

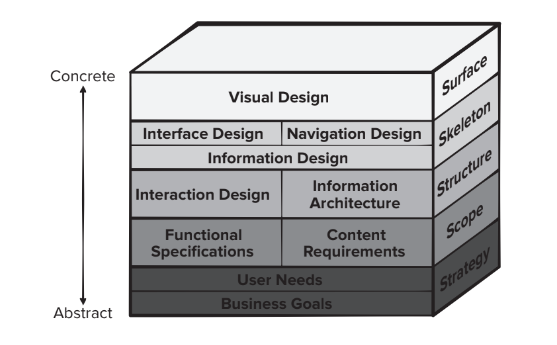
***CLO #4: Describe different phases of software development***

**Q7:** While making a UX design, a software engineer faces the challenge/dilemma of balancing two aspects/concerns. What are they? Which of those aspects should be given preference? Why? [2+1+1 marks]

A software engineer has to make a choice between easier to implement solutions or satisfying user demands. He/she should prefer satisfying user demands because only user satisfaction will eventually determine the success of the project.

***CLO #4: Describe different phases of software development***

**Q8:** List down the breakdown of *user experience design elements* from abstract to concrete levels, used for a more manageable positive user experience. [10 marks]



***CLO #4: Describe different phases of software development***

**Q9:** Which of the following is an internal quality and which is an external quality? [5 marks]

1. The quality of the theme being customizable by the user External
2. The understandability of the system’s class diagram Internal
3. The understandability of the system’s code Internal
4. Security of the system by ensuring that system shall protect data from unauthorized access External
5. The quickness in responding to the user’s commands External