College Roll No. :- 171347
Level :- Bachelor
Programme :- BE Computer
Semester :- VI
Subject :- Object Oriented Software Engineering

Signature of the Examinee/student

Date of Examination
30 June, 2021

exam roll no: 18070416

171347

(1.) a)

=) soln:

The iterative model is a particular implementation of a software development life cycle (SDLC) that focuses on an initial, simplified implementation, which then progressively gains more complexity and a broader feature set until the final system is complete.

An iterative process flow is a process flow that repeads one or more of the activities before proceeding to the next.

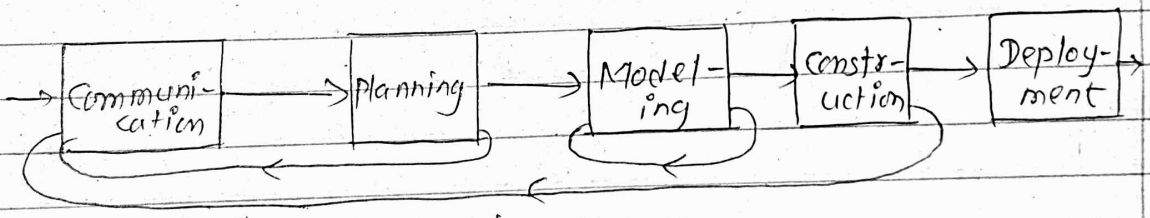


fig: Iterative process flow.

When discussing the iterative method, the concept of incremental development will also often be used liberally and interchangeably. It describes the incremental alterations made during the design and implementation of each new iteration. This is how, iterative modes of software development process help to mitigate different risks that may arise in linear mode of software development process.

Exam roll no. 180704116

171347

(1.) b)

=) soln;

"Unified process (UP) is an architecture-centric, use-case driven, iterative and incremental development process.

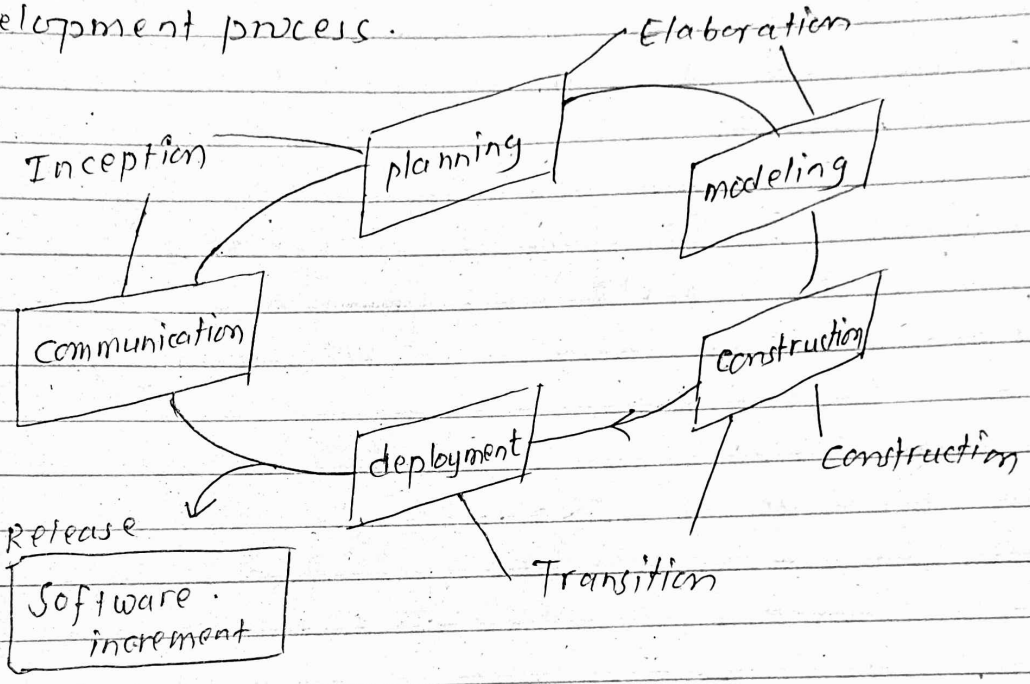


fig: The Unified process

Unified process is an architecture-centric, iterative and ~~ar~~ incremental development process framework that leverages the Object Management Group's (OMG) UML and is compliant with the OMG's SPEM. The UP is widely applicable to different types of software systems, including small-scale and large-scale projects having various degrees of managerial and technical complexity, across different application domains

and organizational cultures. Th
The UP is a process framework idea that provides an infrastructure for executing projects but not all of the details required for executing projects.
Essentially, UP is a software development process framework, a life cycle model involving context, collaborations and interactions.

The UP insists that architecture sits at the heart of the project team's efforts to shape the system. Hence, it is architecture-centric.

The UP is use-case driven process as it includes various use-case diagrams as the requirements for the software development providing the overall view of the project.

The UP is an iterative and incremental development process. The Elaboration, Construction and Transition phases are divided into a series of iterations.
The Inception phase may also be given to priority for large projects.
Each iteration results in an increment, which is a release of the system that contains added or improved functionality compared with the previous release.

exam roll no. 18070416

171347

(2.) a)

⇒ soln:

Object-oriented metrics can be used to measure the software size. Lines of code and functional point metrics are not appropriate in the case of incremental software development as they donot provide adequate details for effort and schedule estimation. Thus, for object-oriented projects, different sets of metrics have been proposed. ~~These are~~ t.

Following are four different object oriented metrics:-

i) Number of scenario scripts:-

Scenario scripts are sequence of steps, which depict the interaction betn the user and the application.

ii) Number of key classes:-

Key classes are independent components defined in object-oriented analysis.

iii) Number of support classes:-

Classes, which are required to implement the system but are indirectly related to problem domain, are known as support classes.

Eg. user interface clauses and computation class are support classes.

exam roll no. 18070416

171347

(iv) Number of subsystems :-

A collection of classes that supports a function visible to the user is known as a subsystem. Identifying subsystems makes it easier to prepare a reasonable schedule in which work on subsystems is divided among project members.

(2.) b)

=) Soln:

Risk transfer can be defined as a mechanism of risk management that involves the transfer of future risks from one person to another. Most common example of risk management is purchasing insurance where the risk of an individual or a company is transferred to a third party (insurance company).

- Risk transfer helps to reduce the risks.
- Risk transfer is the transfer of the implications of risks from one party to another.
- Risk transfer helps in risk management.

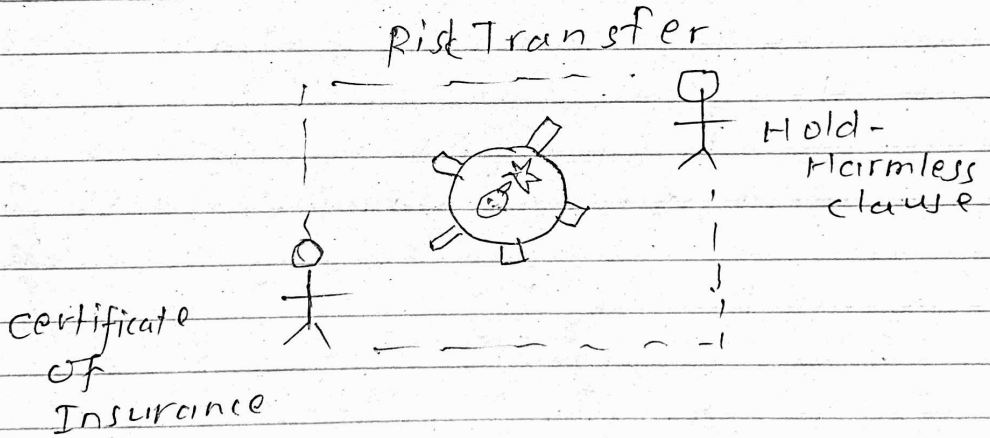
Hence, risk transfer is chosen in software development project.

exam roll no. 18070416

171347

An excimple of risk transfer activity can be as :-

Suppose 'A' buys car insurance for $5,000, which is valid only for the physical damage of the same, and this insurance is right up to 31st Dec 2020. 'A' had a car accident on 20th November 2020. His car suffers from severe physical damage, and the cost of repair of the same accounts $5,050. 'A' can claim a maximum of $5,000 from his insurance provider, and the cost will be solely borned by him.



Types
i) Insurance
ii) Derivatives
iii) Contracts with an Indemnification clause
iv) Outsourcing

exam roll no. 18070416

171347

(3.) a)

=> soln:

Cohesion:-

Cohesion is a measure of the degree to which the elements of the module are functionally related.

Basically, it is the internal glue that keeps the module together.

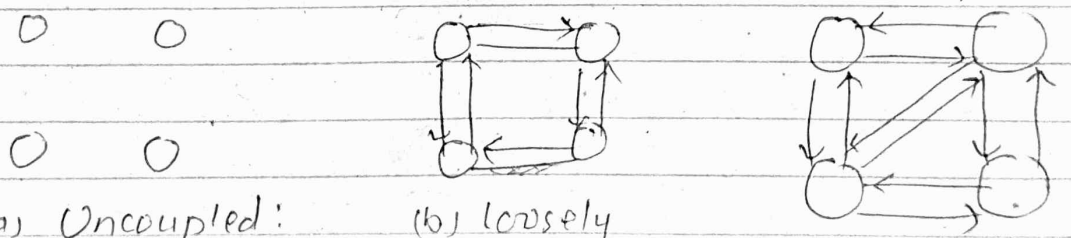
A good software design will have high cohesion.

Coupling:-

Coupling is the measure of the degree of interdependence between the modules.

Low coupling favors a good software.

In software engineering, two modules should be loosely coupled.



(a) Uncoupled: (no dependencies)

(b) loosely coupled: (Some dependencies)

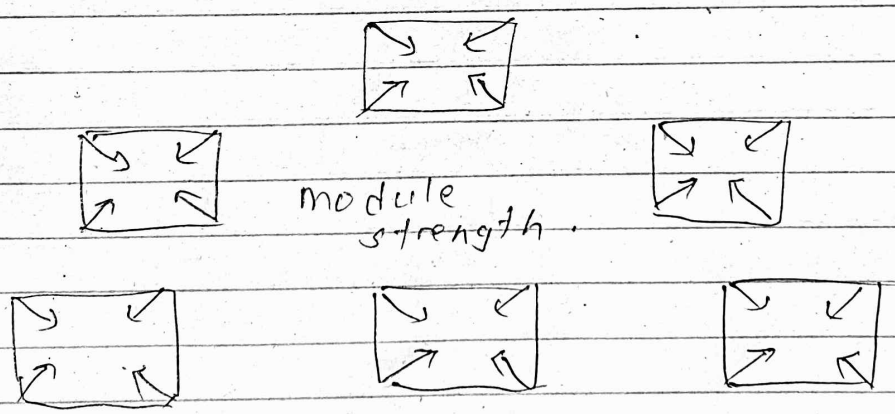
(c) Highly coupled. (many dependencies)

exam roll no. 18070416

171347

A good software design is the one that has low coupling. Coupling is measured by the number of relations between the modules.
That is, the coupling increases as the number of calls between modules increase or the amount of shared data is large. Hence, it is said that a design with high coupling will have more errors.

Next, we have cohesion, the internal glue that keeps the module together.
It measures the strength of functionality within a given module. For example, in highly cohesive systems, functionality is strongly related.



Cohesion = Strength of relations within modules

exam roll no. 18070416

171347

(3.)(b)

=> soln:

case study :-

Activity diagram for the case study :-

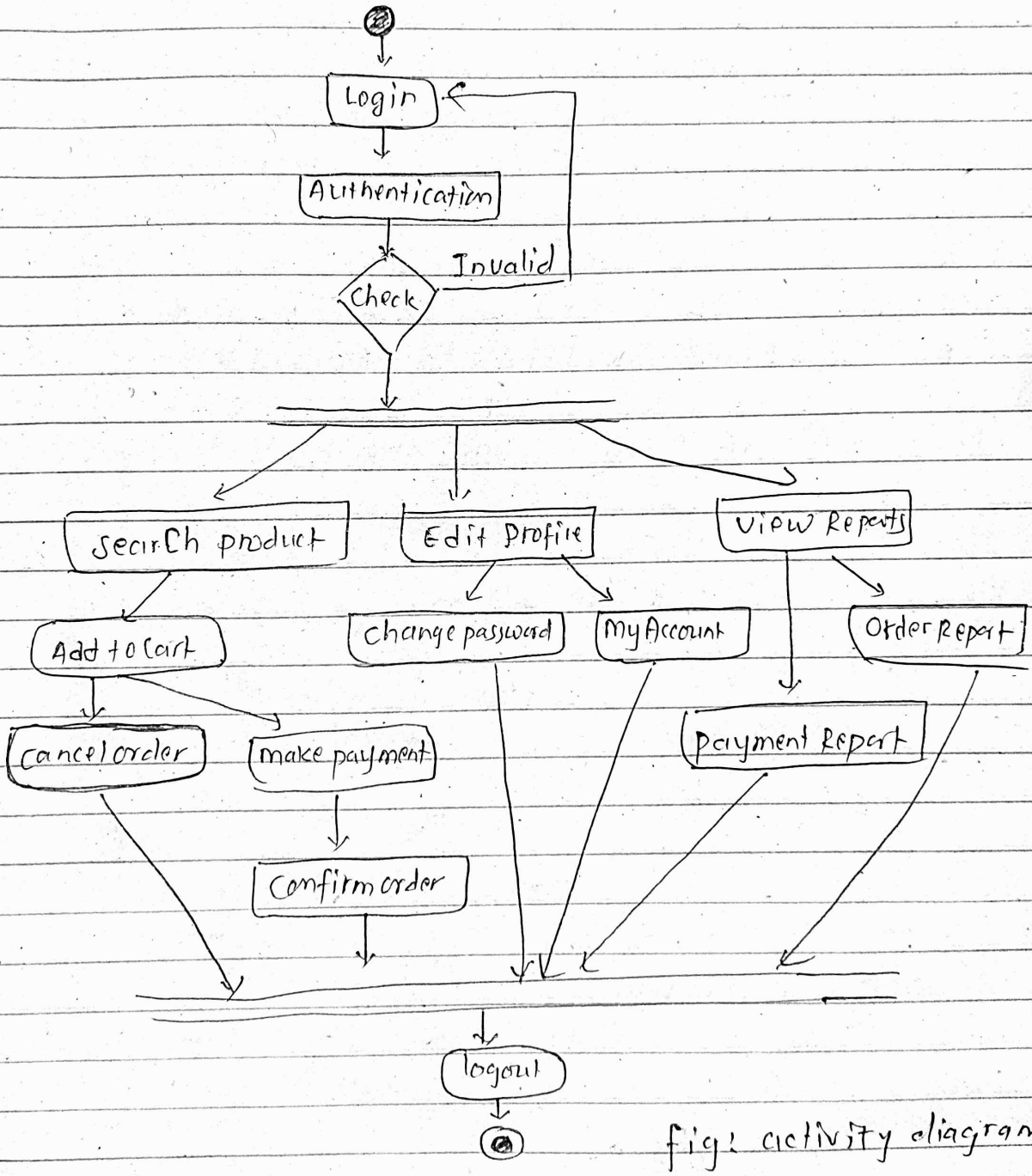


fig: activity diagram

exam roll no. 18070416

171347

(4.) a)

=) soln:

Composition :-

Composition is a form of aggregation with two additional constrains.

eg. Hospital has Department.
University has Department.

pseudocode :-

```
public class University {
    public string Name { get; set; }
    List <Department> _1stDepartments;
    ...
    public void GetDepartments()
    {
        Department _department1 = new Department {Name = "D1"};
        Department _department2 = new Department {Name = "D2"};

        _1stDepartments.Add(_department1);
        _1stDepartments.Add(_department1);
    }
}

public class ClientCode {
    public void CallingCode() {
        University _university1 = new University {Name = "U1"};
    }
}
```

exam roll no. 18070416

171347

Association:-

End1 End2

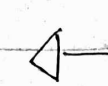
An association is a description of group of links with common structure and common semantic.
eg. A person works for a company.

```
public class student{
    public string Name{get; set;}
    List<Teacher> _lstTeachers{get; set;}

    public void setAssoTeachers(List<Teacher> lstTeachers)
    {_lstTeachers = lstTeachers;
    }
}
public class Teacher{
    public string Name{get; set;}
    List<Student> _lstStudents{get; set;}
}
public class ClientCode{
    public void CallingCode(){
    Student _student = new Student();
    _student.Name = "Std1";
    List<Teacher> _lstTeachers = new List<Teacher>();

    Teacher _teacher1 = new Teacher{Name = "T1"};
    Teacher _teacher2 = new Teacher{Name = "T2"};
    _lstTeachers.Add(_teacher1);
    _lstTeachers.Add(_teacher2);
    _student.setAssoTeachers(_lstTeachers);
}}
```

exam roll no. 18070416

171347

Generalization :-



⇒ A generalization is the relationship betⁿ a super class and sub class.

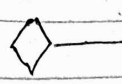
eg. car is a sub class of vehicle

```
public class Vehicle {    // Base class
    public String Color { get; set; }
    public Int32 price { get; set; }
    private string VehType { get; set; }
}
public class Car : Vehicle // Derived class
{
    public Int32 Discount { get; set; }
    public Car() {
        this.discount = 9000;
    }
}
public static class ClientCode {
    public void GetVehicleProperties()
    {
        Car _car = new Car();
        _car.Color = "Red";
        _car.price = 50000;
    }
}
```

exam roll no. 18070416

171347

Aggregation:-



⇒ An aggregation is a strong form of association in which an Aggregate Object is made up of constituent parts.

pseudocode:-

```
public class Dept{
    public String Name {get; set;}
    public List<Teacher> _lstTeacher {get; set;}
}
public class Teacher {
    public String Name {get; set;}
}
public class ClientCode
{
    public void CallingCode(){
    Dept _dept = new Dept();
    _dept.Name = "D1";

    Teacher _teacher1 = new Teacher();
    _teacher1.Name = "T1";
    Teacher _teacher2 = new Teacher();
    _teacher2.Name = "T2";

    _dept._lstTeacher.Add(_teacher1);
    _dept._lstTeacher.Add(_teacher2);
}
}
```

171347

(9.)

(b) => Soln:

Following are the differences bet sub-system design and object design :-

Sub-system Design	Object design
① The basic abstractions, which are given to the user are real world functions.	① The basic abstractions are not real world functions.
② It is a ~~bottom up~~ top down approach	② It is a bottom up approach.
③ We decompose in procedure level.	③ We decompose in class level.
④ Begins by considering the use case diagram and scenarios:	④ Begins by identifying objects and classes.

exam roll no. 18070416

171347

(5.) a)

⇒ Soln:

Scenario based testing for thread testing:-

Scenario based testing is software testing method in which actual scenarios are used for testing the software application instead of test cases.

The purpose of scenario testing is to test end to end scenarios for a specific complex problem of the software.

Thread testing is a software testing type which verify the key functional capabilities of a specific task (thread).

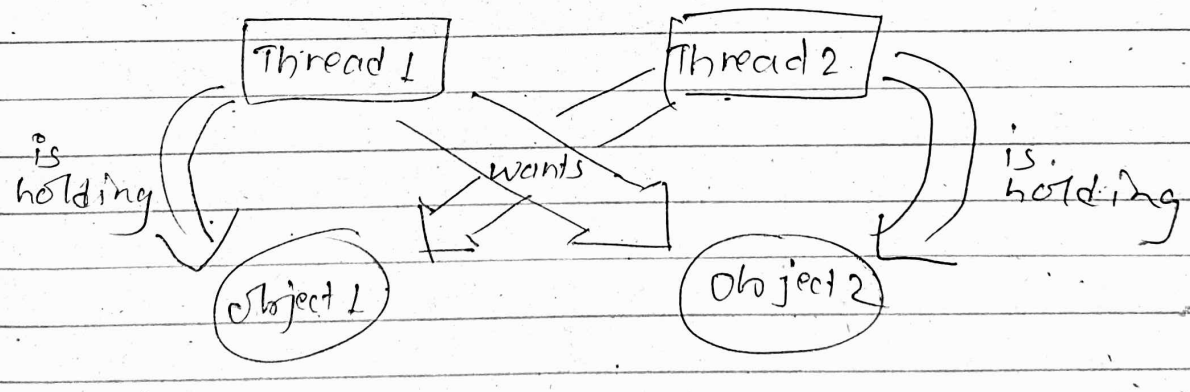


fig: Thread Testing

171347

exam roll no. 18070416

Example:

Test scenario for eCommerce Application.

Test scenario 1: Check the Login Functionality

Test scenario 2: Check the Search Functionality.

Test scenario 3: Check the product Description Page.

Test Scenario 4: Check the payments functionality.

Test Scenario 5: Check the Order History.

(5.) (b)

=) Soln:

FTR

FTR is an abbreviated form of formal Technical Review.

A FTR is a software quality control activity performed by software engineers (and others).

It is important in SQA activities because of the following reasons:-

i) To uncover errors in function and logic.

ii) It helps to verify that the software under review meets its requirements.

iii) It helps to ensure that the software has been represented according to predefined standards.

iv) It aids to achieve software that is developed in a uniform manner.

v) It aids to make projects more manageable.

exam roll no. 18070416

171347

FTR is conducted as a meeting and will be successfull only if it is properly planned, controlled, and attended.

(a) Review meeting:-
Every review meeting should abide by the following constraints:-
- Between three or five people should be involved.
- Advance preparation should occur without exceeding more than 2 hours of work for each person.
- The duration of the meeting must be less than 2 hours.

(b) Review reporting and record keeping:-
Record keeping and reporting are other quality assurance activities.
A review summary report answers the following questions.
- What was reviewed?
- Who reviewed it?
- What were the findings & conclusions.

(c) Review Guidelines.
The following represents a minimum set of guidelines for FTR:-
① Review the product, not the producer
② Set an agenda and maintain it
③ Limit debate and rebuttal
④ Enunciate problem areas but don't attempt every noted problem

exam roll no. 18070416

171347

(5) Take written notes
(6) Limit the number of participants and insist upon advance preparation.
(7) Develop a check list each work product that is likely to be reviewed.
(8) Allocate resources and time schedule for FTRs.
(9) Conduct meaningful training for all reviewers
(10) Review your earlier reviews.

(7.)

a) Functional Vs Non-Functional Requirements:-

Software requirement is a functional or non-functional need to be implemented in the system.

Functional requirement means providing particular service to the user.
Example, in context to banking application the functional requirement will be when customer selects "View Balance" they must be able to look at their latest account balance.

Non-functional requirements are another software requirement.
It can be a performance requirement.
Eg, a non-functional requirement is where every page of the system should be visible to the users within 5 seconds.

exam roll no. 18070416

171347

7.

(b)

⇒ Alpha & Beta Testing

Alpha Testing is a type of software testing performed to identify bugs before releasing the production to real users or to the public. Alpha testing is one of the user acceptance testing.

- Alpha testing involves both white box and black box testing
- It is performed by testers who are usually internal employees of the organization.
- It is performed at developer's site.

Beta Testing:-

Beta testing is performed by real users of the software application in a real environment. Beta testing is one of the type of User Acceptance testing.

- Beta testing is commonly black box testing.
- Beta testing is performed by the clients who are not part of the organization.
- It is performed at the end-user of the product.

exam roll no. 18070416

171347

(6.) (a)

=> Soln:

Relationship of mean-time-between-failure (MTBF), mean-time-to-failure (MTTF), mean-time-to-repair with service availability

$$MTBF = MTTF + MTTR.$$

where, MTBF = mean-time-between-failure.
MTTF = mean-time-to-failure
MTTR = mean-time-to-repair.

Software availability is the probability that a program is operating according to requirements at a given point in time & is

$$\text{Availability} = \frac{MTTF}{MTTF + MTTR} \times 100\%.$$

Example:- A mechanical mixer designed to operate for 10 hours a day. Suppose mixer breaks down after normally operating for 5 days. The MTBF for this case is 50 hours as calculated below:

$$MTBF = \frac{(10 \text{ hours per day} * 5 \text{ days})}{1 \text{ breakdown}}$$

$$\boxed{MTBF = 50 \text{ hour}}$$ Ans

171347

exam roll no. 18070416

(6.)

(b)

=> soln:

SPI & SPI framework :-

Software process improvement encompasses a set of activities that will lead to a better software process, and, as a consequence, higher-quality software delivered in timely manner.

SPI framework:

SPI framework defines:

(i) A set of characteristics that must be present if an effective software process is to be achieved.

(ii) A method for assessing whether those characteristics are present.

(iii) A mechanism for summarizing the results of any assessment, and

(iv) A strategy for assisting a software organization in implementing those process characteristics that have been found to be weak or missing.

171347

exam roll no. 18070416

CMMI

The original CMM was developed and upgraded by the Software Engineering Institute throughout the 1990s as a complete SPI framework.

Today, it has evolved into the Capability Maturity Model Integration (CMMI), a comprehensive process meta-model that is predicated on a set of system and software engineering capabilities that should be present as organizations reach different levels of process capability and maturity.