190030430 D.Sri Rama Krishna Reddy

SE Adv Practical-10

Pre-lab:

1) Explain Agile Unified process.

Ans:

Agile Unified Process (AUP) is a simplified version of the Rational Unified Process (RUP) developed by Scott Ambler. It describes a simple, easy to understand approach to developing business application software using agile techniques and concepts yet still remaining true to the RUP.

2)What is a lifeline?

Ans:

Lifelines – A lifeline is a named element which depicts an individual participant in a sequence diagram. Lifeline is a rectangle containing an identifier with a dashed line extending below the rectangle.

3)What does a message mean?

Ans:

All communications from one object to another are called messages and are represented by message arrows in sequence diagrams.

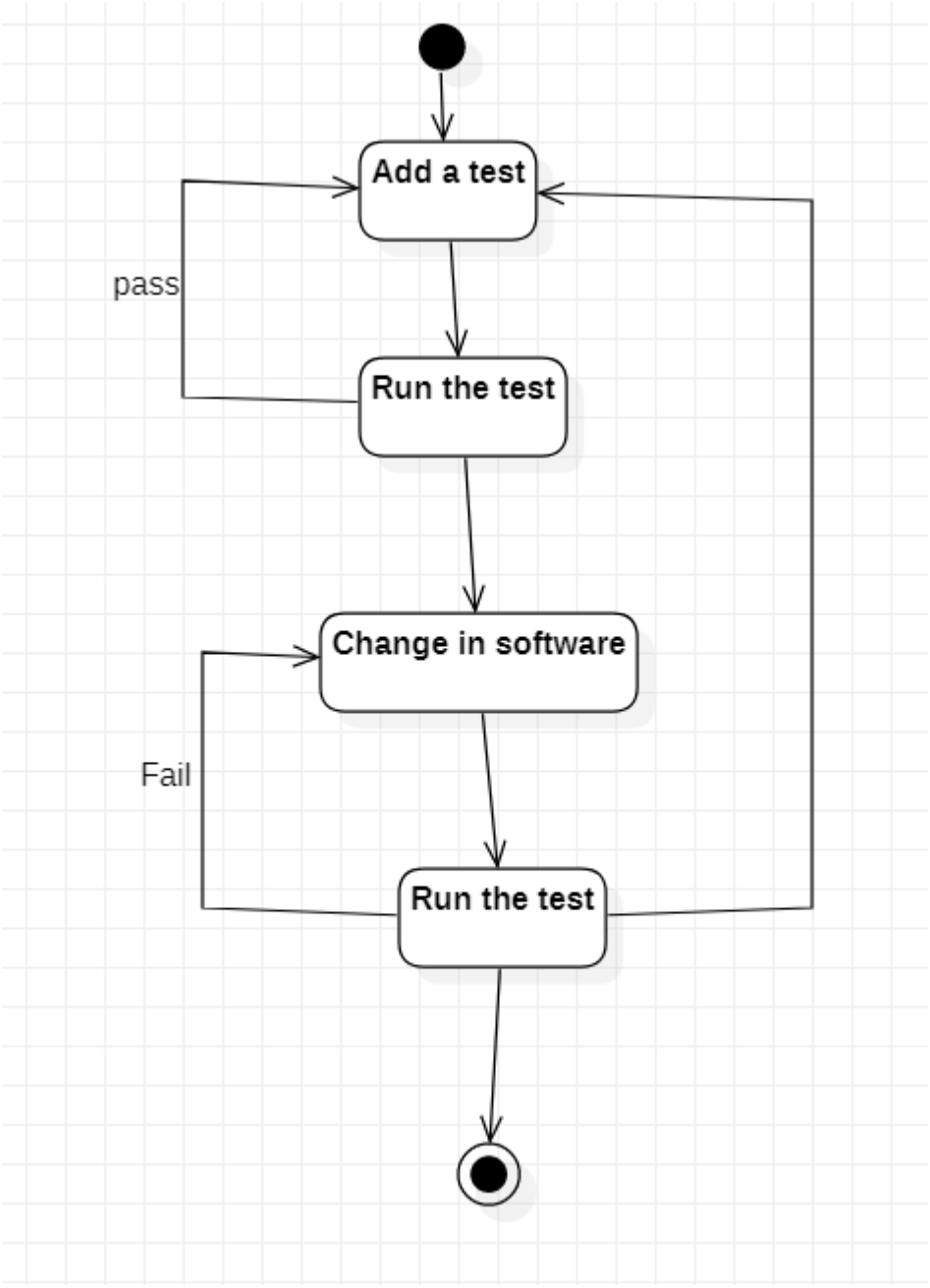
4)What are the three different types of message arrows?

Ans:

There are three different types of message arrows are- Synchronous, Asynchronous, Synchronous with instance creation.

In-lab:

1) PSCP Company wants to use a developments method which helps to create better modularized, extensible and flexible code and the process approach drives the Agile team to plan, develop and test the small units to be integrated at advanced stage. Under this approach, the concerned member delivers and performs better because of being more focused on smaller unit. Explain about this development process and also draw a UML activity diagram on developer TDD.



Post-lab:

1.What is the Difference between a lifeline and an actor

Ans:

Actors – An actor in a UML diagram represents a type of role where it interacts with the system and its objects. It is important to note here that an actor is always outside the scope of the system we aim to model using the UML diagram.We use actors to depict various roles including human users and other external subjects. We represent an actor in a UML diagram using a stick person notation. We can have multiple actors in a sequence diagram.

Lifelines – A lifeline is a named element which depicts an individual participant in a sequence diagram. So basically each instance in a sequence diagram is represented by a lifeline. Lifeline elements are located at the top in a sequence diagram. The standard in UML for naming a lifeline follows the following format – Instance Name : Class Name.We display a lifeline in a rectangle called head with its name and type. The head is located on top of a vertical dashed line.

Difference between a lifeline and an actor – A lifeline always portrays an object internal to the system whereas actors are used to depict objects external to the system.

2.What is Messages in sequence diagram

Ans:

Sequence diagrams depict workflow or activity over time using Messages passed from element to element. In the software model. These Messages correspond to Class operations and behavior.

3.What is the purpose of Synchronous messages

Ans:

Synchronous Messages

Messages are able to flow in both directions, to and from. Essentially it means that synchronous messaging is a two way communication. i.e. Sender sends a message to receiver and receiver receives this message and gives reply to the sender.

4.Describes Asynchronous Messages

Ans:

Asynchronous Messages

Asynchronous messaging is a communication method wherein the system puts a message in a message queue and does not require an immediate response to continue processing. Examples include a request for information, explanation or data needed but not needed immediately.

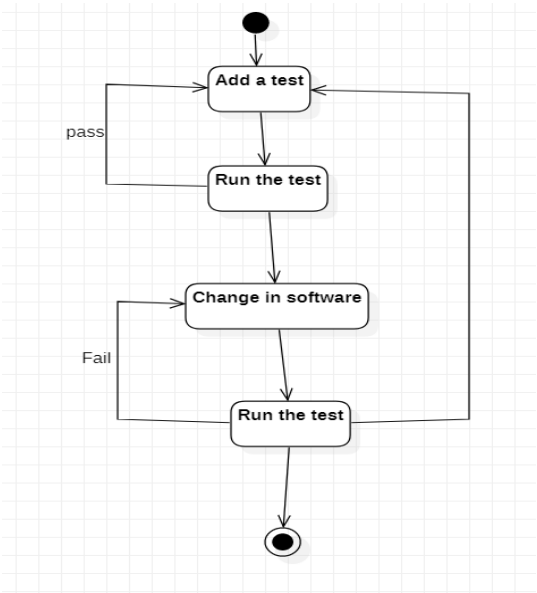
190030430 D.Sri Rama Krishna Reddy

SE Adv Practical-11

Pre-lab:

1) "TDD, test-driven development, or development through testing, is a software development methodology that is mainly based on the repetition of short cycles of development. While traditional testing requires a lot of time and money, using TDD results in faster and cleaner code."
Contrast the above statement with the help of a flowchart, which should in detail explain about the TDD process.

Ans:



2) What are the levels of TDD? Describe in brief.

Ans:

There are two levels of TDD:

1. **Acceptance TDD (ATDD):**With ATDD you write a single acceptance test, or behavioral specification depending on your preferred terminology, and then just enough production functionality/code to fulfill that test. The goal of ATDD is to specify detailed, executable requirements for your solution on a just in time (JIT) basis. ATDD is also called Behavior Driven Development (BDD).
2. **Developer TDD:**With developer TDD you write a single developer test, sometimes inaccurately referred to as a unit test, and then just enough production code to fulfill that test. The goal of developer TDD is to specify a detailed, executable design for your solution on a JIT basis. Developer TDD is often simply called TDD.

In-lab:

1) Draw sequence diagram for withdrawing an amount from ATM.

It should consist of 4 Classes and sequence of data Flow between the classes is showed in the diagram below.

The Four Classes are

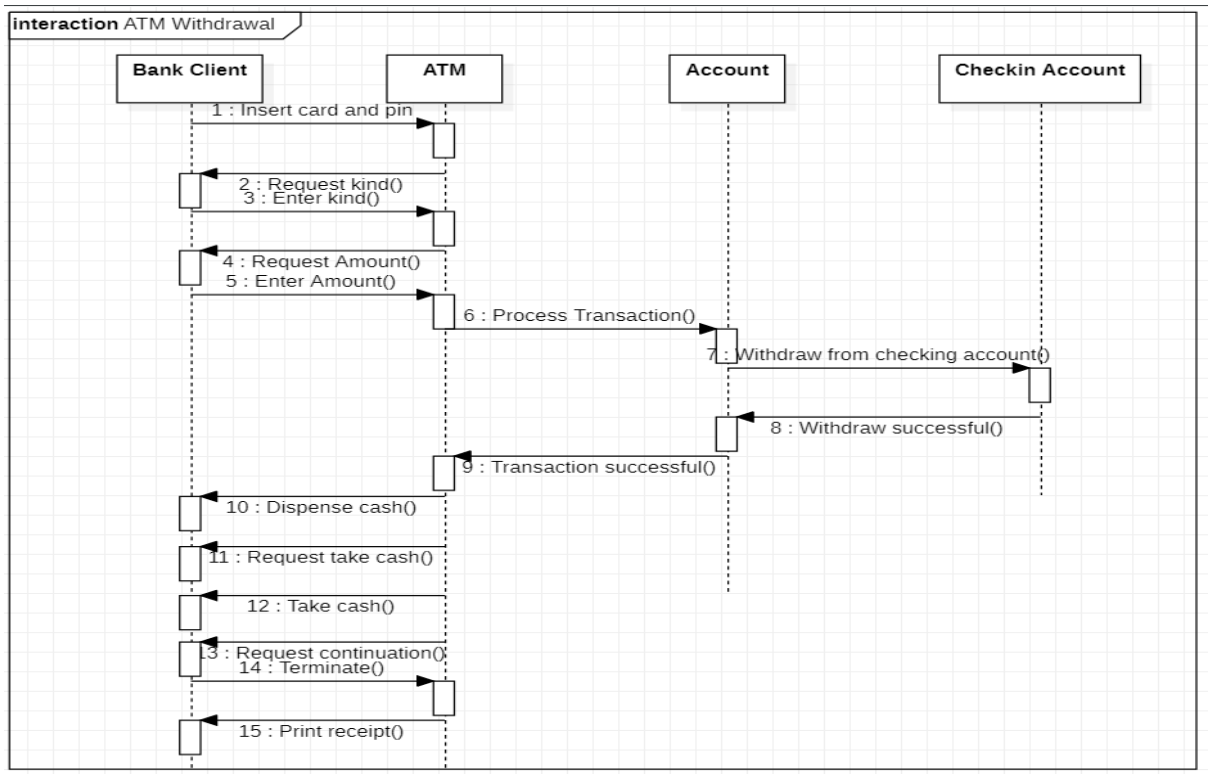
- Bank Client
- ATM Machine
- Account
- Checking Account

The different methods used between classes are as follows

The following is the order of the methods used in the program.

1. Request kind ()
2. Enter kind ()
3. Request Amount ()
4. Enter Amount()
5. Process Transaction ()
6. Withdraw from Checking Account()
7. Withdraw successful()
8. Transaction Successful()
9. Dispense Cash()
10. Request Take cash()
11. Take cash()
12. Request continuation()
13. Terminate()
14. Print Recipt()

Ans: Sequence diagram for withdrawing an amount from ATM



2.Which is an open source software with Regression Testing Framework used by developers to implement unit testing in Java and accelerate programming speed and increase the quality of code? Draw the Flow chart or process flow diagram of the above framework.

Ans:

Aim: To draw the process flow diagram of unit

Theory:JUnit is a unit testing framework for the Java Programming Language. It is written in Java and is an Open Source Software maintained by the JUnit.org community. JUnit is a simple, open source framework to write and run repeatable tests. It is an instance of the Unit architecture for unit testing frameworks. JUnit features include: Assertions for testing expected results.

Junit is used by both developers and testers JUnit promotes the idea of "first testing then coding", which emphasizes on setting up the test data for a piece of code that can be tested first and then implemented This approach is like "test a little, code a little, test a little, code a little " It increases the productivity of the programmer and the stability of program code, which in turn reduces the stress on the programmer and the time spent on debugging.

Procedure:

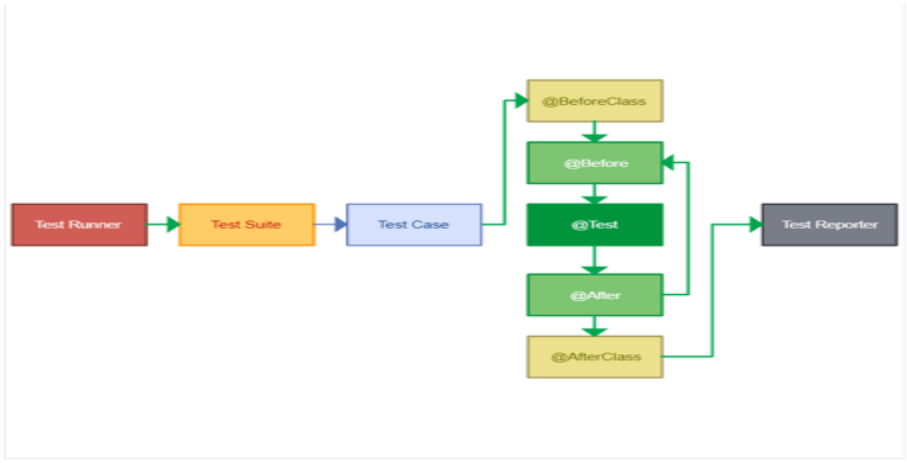
1) Place the terms test runner, test suite, test case and the annotations @BeforeClass ,@Before,@Test,@After @AfterClass Test Report in rectangular boxes

2) Connect the terms Test runner test suite, test case ,@BeforeClass with forward arrows

3) Now connect the annotations @BeforeClass.@Before, @ Test, @After @AfterClass and Test Reporter with forward arrows

4) Now connect @After and @Before with a forward arrow, because that is to another test case or another new method

Diagram :



Post-lab:

1) What are the benefits of TDD?

Ans: Advantages of TDD approach

1. Better program design and higher code quality
2. Detailed project documentation
3. TDD reduces the time required for project development
4. Code flexibility and easier maintenance
5. With TDD you will get a reliable solution
6. Save project costs in the long run

2) What is white box testing?

Ans: White box testing is a testing technique, that examines the program structure and derives test data from the program logic/code. The other names of glass box testing are clear box testing, open box testing, logic driven testing or path driven testing or structural testing.

3) What is meant by DSDM?

Ans:DSDM is an Agile method that focuses on the full project lifecycle, DSDM formally known as Dynamic System Development Method.

190030430 D.Sri Rama Krishna Reddy

SE Adv Practical-12

Pre-lab:

1.What are the different types of models in CMMI?

Ans:

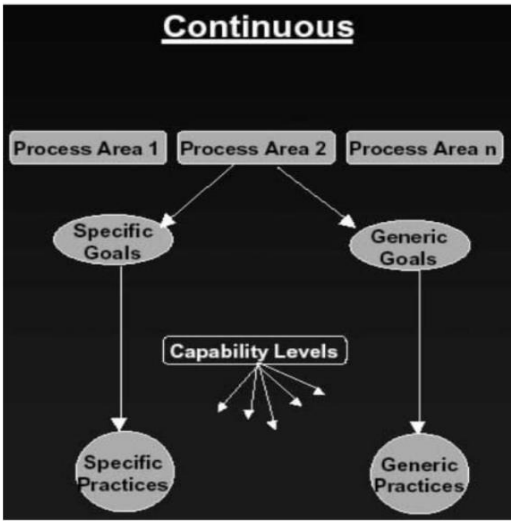
CMMI Models:

There are three CMMI Models available that addresses the following:

- The development of Software Products and Services i.e. CMMI–DEV
- The acquisition of Products and Services i.e. CMMI–ACQ
- The establishment, management, and delivery of services i.e. CMMI–SVC

2.Explain the continuous CMMI model representation by using a diagram.

Ans:



Continuous Representation:

The continuous representation is the approach used in the SECM and the IPD-CMM. This approach allows an organization to select a specific process area and improve relative to it. The continuous representation uses Capability Levels to characterize improvement relative to an individual process area.

CMMI Continuous Representation:

Allows you to select the order of improvement that best meets your organization's business objectives and mitigates your organization's areas of risk

Enables comparisons across and among organizations on a process-area-by-process-area basis.

Thus Continuous Representation provides flexibility for organizations to choose which processes to emphasize for improvement, as well as how much to improve each process.

3.What are the capabilities/maturity levels of CMMI?

Ans:

Maturity Levels of CMMI

Maturity levels represent a staged path for an organization's performance and process improvement efforts based on predefined sets of practice areas. Within each maturity level, the predefined set of PA's also provide a path to performance improvement. Each maturity level builds on the previous maturity levels by adding new functionality or rigor.

Maturity Level 0: Incomplete
Ad hoc and unknown. Work may or may not get completed.

Maturity Level 1: Initial
Unpredictable and reactive. Work gets completed but is often delayed and over budget.

Maturity Level 2: Managed
Managed on the project level. Projects are planned, performed, measured, and controlled.

Maturity Level 3: Defined
Proactive, rather than reactive. Organization-wide standards provide guidance across projects, programs, and portfolios.

Maturity Level 4: Quantitatively Managed
Measured and controlled. Organization is data-driven with quantitative performance improvement objectives that are predictable and align to meet the needs of internal and external stakeholders.

Maturity Level 5: Optimizing
Stable and flexible. Organization is focused on continuous improvement and is built to pivot and respond to opportunity and change. The organization's stability provides a platform for agility and innovation.

4.What is staged CMMI?

Ans: The staged representation is concerned with selecting multiple process areas to improve within a maturity level; whether individual processes are performed or incomplete is not the primary focus.

In-lab:

1) Write a program which counts number of times each method with differentannotations will be executed in a program. Explain about annotations in Junit.
Ans:

Aim: To understand the functionality of annotations in Junit.

Requirements: Junit installed in Eclipse IDE, or in any other IDEs.

Theory:

Annotations:

Junit annotation is a special form of syntactic meta-data that can be added to Java source code for better code readability and structure While writing a program we write plenty of methods, each method is having an annotation The order of Execution of methods depends on annotations only.

Implementation:

```
import org.junit.jupiter.api.*;

import jdk nashorn internal.ir.annotations.Ignore;

class JunitDemo {

public static int Beforeall count, Beforeaach_count, Afterall_cownt, Afterea ch_count, Test _Count;

@BeforeAll

static void setUpBeforeClass()

{

System.out.prmtln("@Beforeall+Beforeall_count"@Afterall* + Afteral Count"and@BeforeEach"+Beforeeach count+"n@AfterEach "+aftereach com-a TestTest_count)

Beforeall_count++;

}

@AfterAll

static void tear Down AfterClass(){

Afterall_count++;

System out print (“@Beforeall:”-Before count+”\n@Aterall:”+Afteral_count+”\n@ Beforeach”+Beforeeach_count+”\n@AfterEach”+Aftereach_count+”\nTest”+Test_count”);

}

@BeforeEach
```

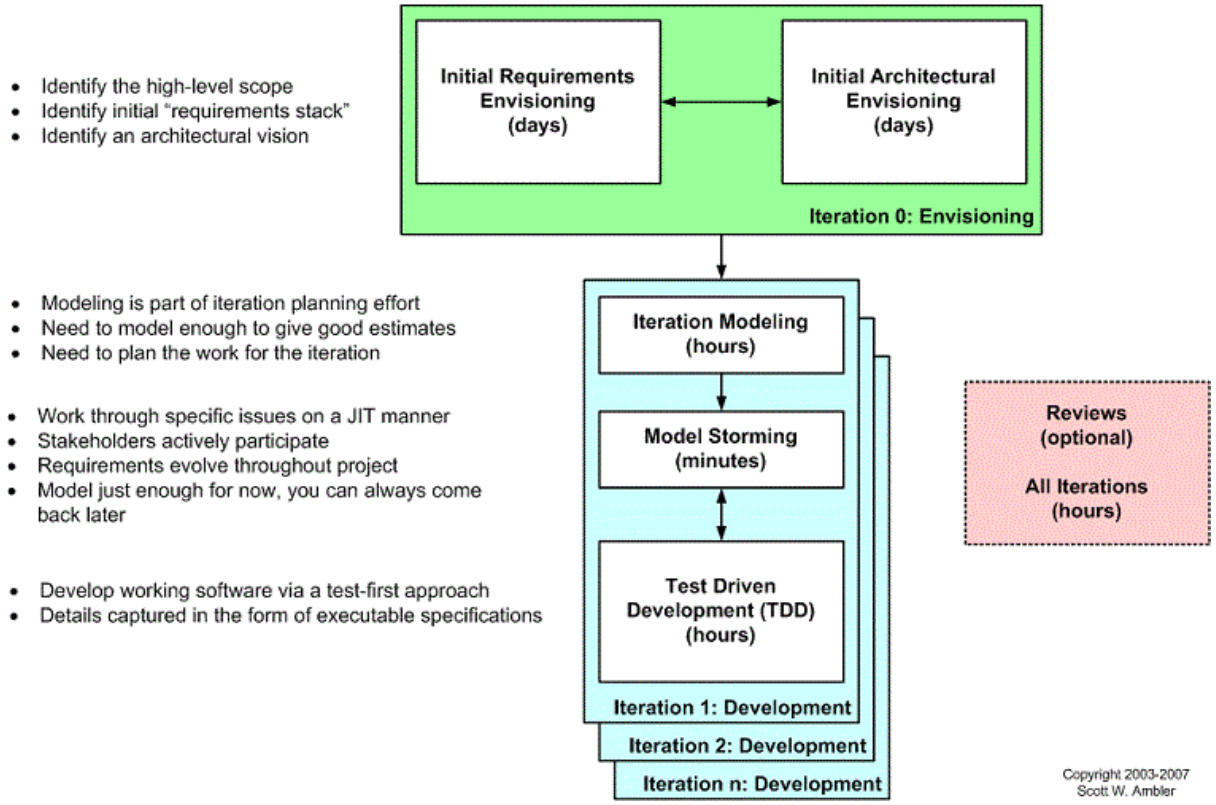
```
void SetUp(){
Beforeach_count++,
}
@AfterEach
void tear Down(){
Aftereach_count++,
}
@Test
void test1(){
System out println("Code for 1st Test Case"),
Test_count++;
}
@Test
public void test2(){
System out println(“Code for 2nd test case”);
Test_count++;
}
@Test
public void test3(){
System out print(“Code for 3rd test case”);
Test_count++;
}
@Ignore
public void test 4(){
System.out.println("code for 4th test case");
Test_count++;
}
}
```

Conclusion:JUnit provides annotations and Assert methods which makes you to code faster with increased quality.

2) What is a critical strategy for scaling agile software development beyond the small, co-located team approach that we saw during the first stage of agile adoption? Do explain its life cycle (Diagrammatically).

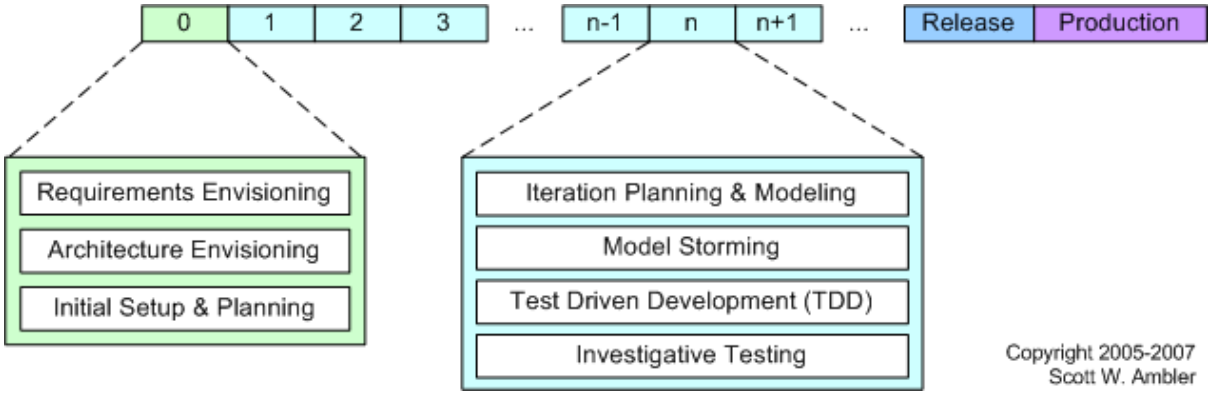
Ans:

AMDD is a critical strategy for scaling agile software development beyond the small, co-located team approach that we saw during the first stage of agile adoption.



The AMDD lifecycle: Modeling activities throughout the lifecycle of a project.

Figure depicts a high-level lifecycle for AMDD for the release of a system. First, let's start with how to read the diagram. Each box represents a development activity. The envisioning includes two main sub-activities, initial requirements envisioning and initial architecture envisioning. These are done during Inception, iteration being another term for cycle or sprint. "Iteration 0, or Inception", is a common term for the first iteration before you start into development iterations, which are iterations one and beyond (for that release). The other activities - iteration modeling, model storming, reviews, and implementation - potentially occur during any iteration, including Inception. The time indicated in each box represents the length of an average session: perhaps you'll model for a few minutes then code for several hours.



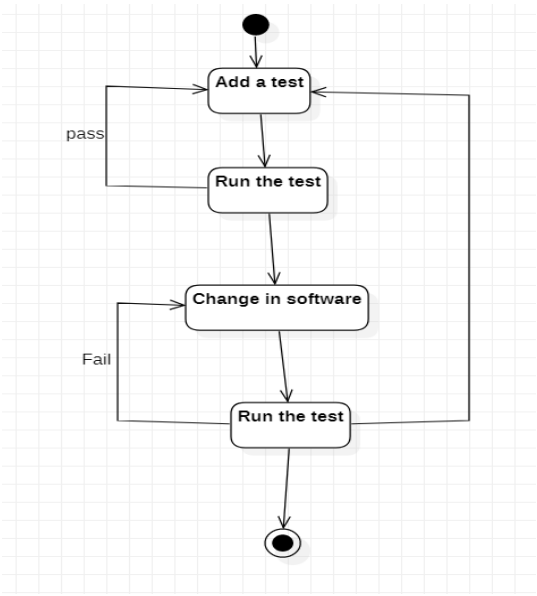
AMDD Through the Agile Development Lifecycle

Figure depicts how the AMDD activities fit into the various iterations of the agile software development lifecycle. It's simply another way to show that an agile project begins with some initial modeling and that modeling still occurs in each construction iteration.

Post Lab:

1) Draw the UML diagrams to show how the levels in TDD work.

Ans:



2) Give examples of Black Box Testing techniques.

Ans: Black Box Testing Techniques

1.Equivalence Partitioning

Testers can divide possible inputs into groups or "partitions", and test only one example input from each group. For example, if a system requires a user's birth date and provides the same response for all users under the age of 18, and a different response for users over 18, it is sufficient for testers to check one birth date in the "under 18" group and one date in the "over 18" group.

2.Boundary Value Analysis

Testers can identify that a system has a special response around a specific boundary value. For example, a specific field may accept only values between 0 and 99. Testers can focus on the boundary values (-1, 0, 99 and 100), to see if the system is accepting and rejecting inputs correctly.

3.Decision Table Testing

Many systems provide outputs based on a set of conditions. Testers can then identify "rules" which are a combination of conditions, identify the outcome of each rule, and design a test case for each rule.

For example, a health insurance company may provide different premium based on the age of the insured person (under 40 or over 40) and whether they are a smoker or not. This generates a decision table with four rules and up to four outcomes.

Conditions	Rule 1	Rule 2	Rule 3	Rule 4
Under 40	False	False	True	True
Smoker	False	True	False	True
OUTCOMES				
1: High premium				
2: Medium premium				
3: Low premium				

In this case four use cases (one for each rule) would be sufficient to fully test the system.

4.State Transition Testing

In some systems, significant responses are generated when the system transitions from one state to another. A common example is a login mechanism which allows users to authenticate, but after a specific number of login attempts, transition to a different state, locking the account.

If testers identify a state transition mechanism, they can design test cases that probe the system when it transitions states. For example, for a system that locks the account after five failed login attempts, a test case can check what happens at the sixth login attempt.

5.Error Guessing

This technique involves testing for common mistakes developers make when building similar systems. For example, testers can check if the developer handled null values in a field, text in a numeric field or

numbers in a text-only field, and sanitization of inputs—whether it is possible to submit user inputs that contain executable code, which has security significance.

A specific type of error guessing is testing for known software vulnerabilities that can affect the system under test.

3) What is the six-sigma model?

Ans:

Six Sigma is a method that provides organizations tools to improve the capability of their business processes. This increase in performance and decrease in process variation helps lead to defect reduction and improvement in profits, employee morale, and quality of products or services.