CMR
INSTIT UTE OF
TECHNLOGY

USN | | | | | | | | | | |

First Internal Test-October 2019

Sub:
Software Engineering

Sub Code:

18CS35

Branch:

ISE

Date:
12/10/2019

Duration:

90 min's

Max Marks:

50

Sem/Sec:

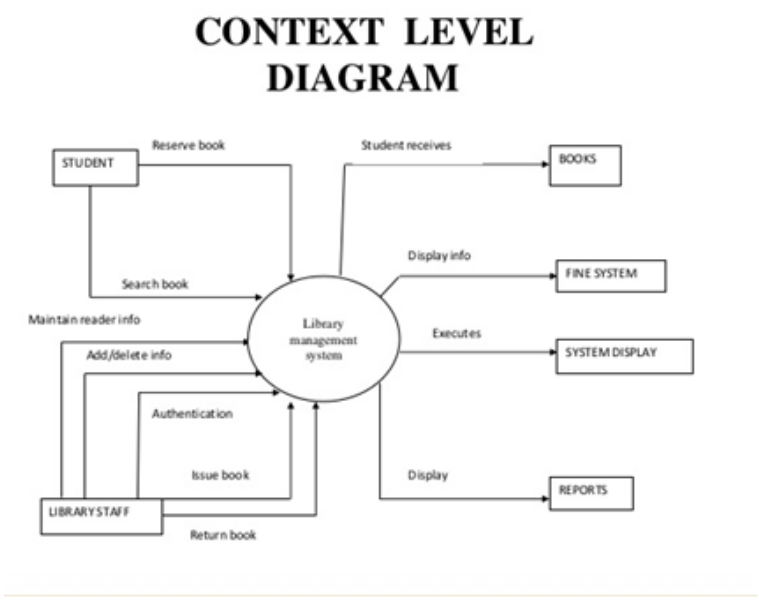
III A ,B,C

OBE

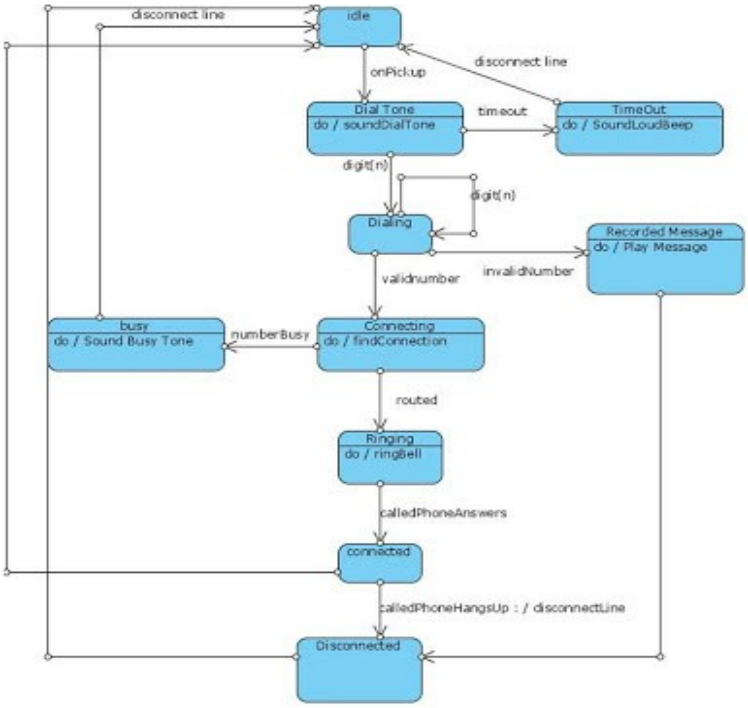
Scheme and Solution

MARKS

1 (a) Draw a context model for Library Management System. How the interactions are modeled? [6M].



1 (b) With the help of neat state diagram, illustrate the working of a telephone calling system.[4M]



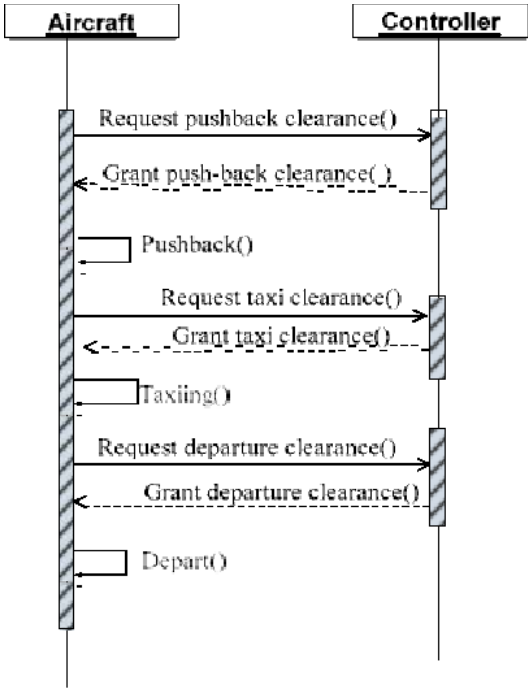
2(a) What is Model Driven Engineering? State the three types of abstract Design models recommended by MDA. Distinguish between MDA and MDE.[8M]

- Model-driven engineering (MDE) is an approach to software development where models rather than programs are the principal outputs of the development process.
- The programs that execute on a hardware/software platform are then generated automatically from the models.
- Proponents of MDE argue that this raises the level of abstraction in software engineering so that engineers no longer have to be concerned with programming language details or the specifics of execution platforms.

Types of models

- A computation independent model (CIM)
  - These model the important domain abstractions used in a system. CIMs are sometimes called domain models.
- A platform independent model (PIM)
  - These model the operation of the system without reference to its implementation. The PIM is usually described using UML models that show the static system structure and how it responds to external and internal events.
- Platform specific models (PSM)
  - These are transformations of the platform-independent model with a separate PSM for each application platform. In principle, there may be layers of PSM, with each layer adding some platform-specific detail.

2 (b) Draw a sequence diagram describing data collection of Air Traffic Control system [2M]

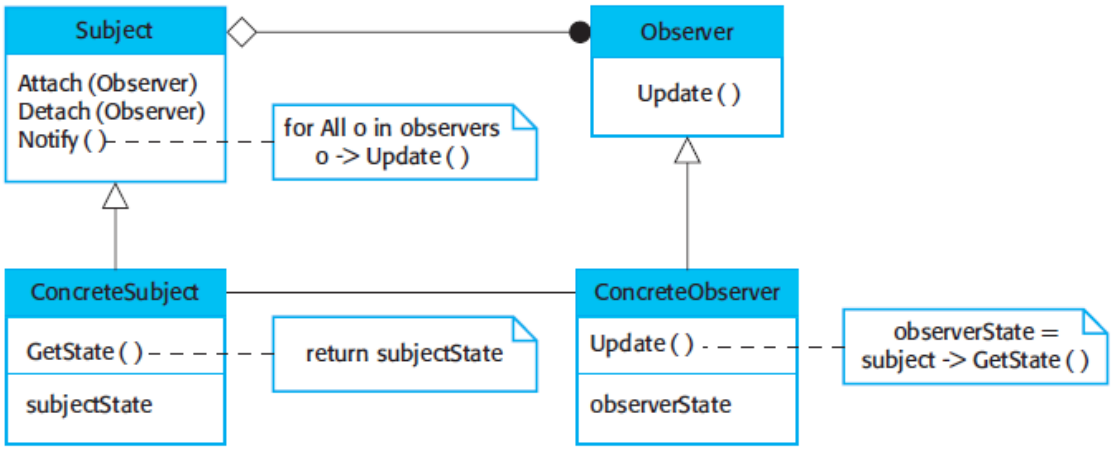


3 (a) What is Design pattern? Explain four elements of design pattern.[5M]

- A design pattern is a way of reusing abstract knowledge about a problem and its solution.
- A pattern is a description of the problem and the essence of its solution.
- It should be sufficiently abstract to be reused in different settings.
- Pattern descriptions usually make use of object-oriented characteristics such as inheritance and polymorphism.
- *Patterns and Pattern Languages are ways to describe best practices, good designs, and capture experience in a way that it is possible for others to reuse this experience.*
- Name
  - A meaningful pattern identifier.
- Problem description.
- Solution description.
  - Not a concrete design but a template for a design solution that can be instantiated in different ways.
- Consequences
  - The results and trade-offs of applying the pattern.

The Observer pattern

- Name
  - Observer.
- Description
  - Separates the display of object state from the object itself.
- Problem description
  - Used when multiple displays of state are needed.
- Solution description
  - See slide with UML description.
- Consequences
  - Optimisations to enhance display performance are impractical.



Pattern name	**Observer**
Description	Separates the display of the state of an object from the object itself and allows alternative displays to be provided. When the object state changes, all displays are automatically notified and updated to reflect the change.
Problem description	In many situations, you have to provide multiple displays of state information, such as a graphical display and a tabular display. Not all of these may be known when the information is specified. All alternative presentations should support interaction and, when the state is changed, all displays must be updated. This pattern may be used in all situations where more than one display format for state information is required and where it is not necessary for the object that maintains the state information to know about the specific display formats used.
Pattern name	**Ob7server**
Solution description	This involves two abstract objects, Subject and Observer, and two concrete objects, ConcreteSubject and ConcreteObject, which inherit the attributes of the related abstract objects. The abstract objects include general operations that are applicable in all situations. The state to be displayed is maintained in ConcreteSubject, which inherits operations from Subject allowing it to add and remove Observers (each observer corresponds to a display) and to issue a notification when the state has changed. The ConcreteObserver maintains a copy of the state of ConcreteSubject and implements the Update() interface of Observer that allows these copies to be kept in step. The ConcreteObserver automatically displays the state and reflects changes whenever the state is updated.
Consequences	The subject only knows the abstract Observer and does not know details of the concrete class. Therefore there is minimal coupling between these objects. Because of this lack of knowledge, optimizations that enhance display performance are impractical. Changes to the subject may cause a set of linked updates to observers to be generated, some of which may not be necessary.

3 (b) Explain terms i)OO Design using UML ii) context model iii) Dynamic Model [5M]

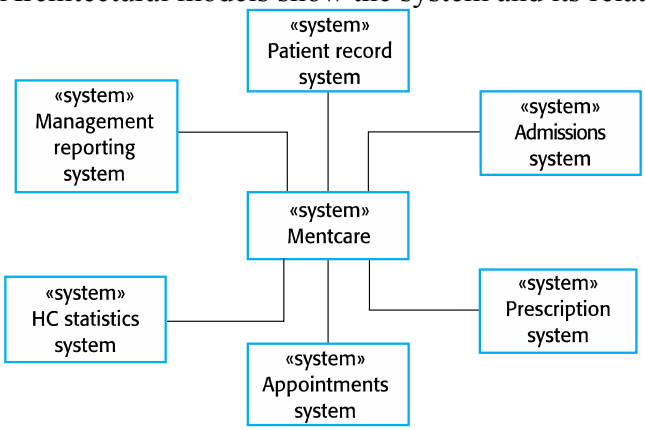
i.UML

- Activity diagrams, which show the activities involved in a process or in data processing .
- Use case diagrams, which show the interactions between a system and its environment.
- Sequence diagrams, which show interactions between actors and the system and between system components.
- Class diagrams, which show the object classes in the system and the associations between these classes.
- State diagrams, which show how the system reacts to internal and external events.

ii. Context model

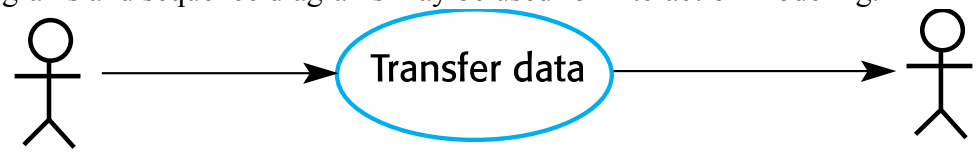
- Context models are used to illustrate the operational context of a system - they show what lies outside the system boundaries.
- Social and organisational concerns may affect the decision on where to position system boundaries.

Architectural models show the system and its relationship with other systems



iii. Dynamic model (Interaction models)

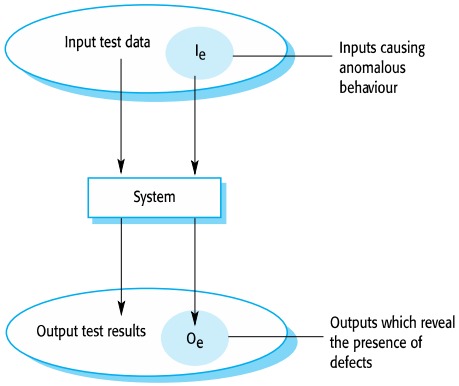
- Modeling user interaction is important as it helps to identify user requirements.
- Modeling system-to-system interaction highlights the communication problems that may arise.
- Modeling component interaction helps us understand if a proposed system structure is likely to deliver the required system performance and dependability.
- Use case diagrams and sequence diagrams may be used for interaction modeling.

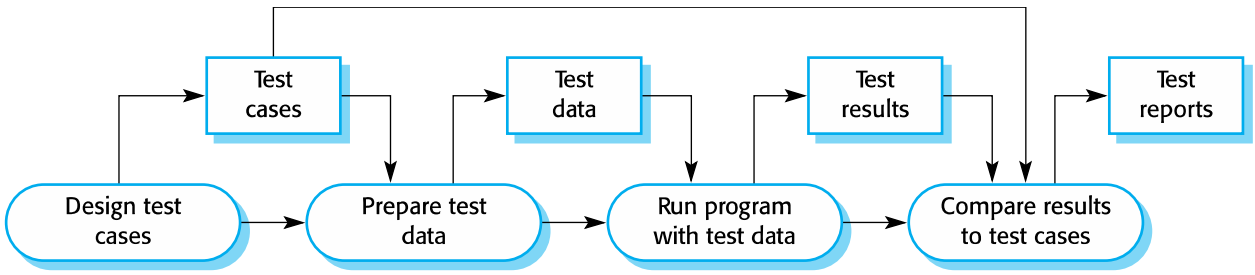


Example: Medical receptionist — Patient record system

4 (a) Discuss the types of testing at various stages of SDLC. [5M]

- Development testing
- Test-driven development
- Release testing
- User testing





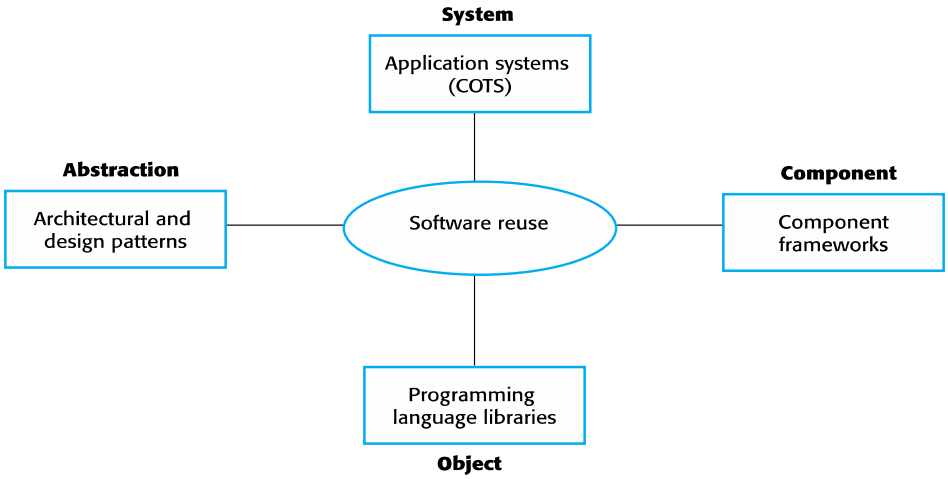
Stages of testing

- Development testing, where the system is tested during development to discover bugs and defects.

- Release testing, where a separate testing team test a complete version of the system before it is released to users.

User testing, where users or potential users of a system test the system in their own environment

4 (b) What is software reuse? State the general models of open source licenses. [5M]



Software Reuse : Most modern software is constructed by reusing existing components or systems. When you are developing software, you should make as much use as possible of existing code.

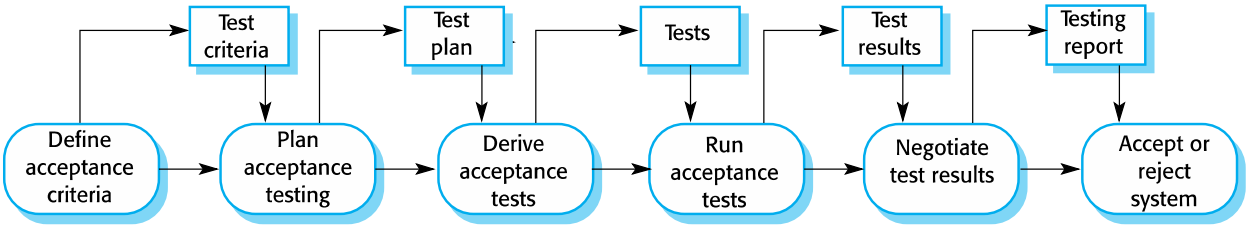
- An approach to development based around the reuse of existing software emerged and is now generally used for business and scientific software.

License models

- The GNU General Public License (GPL). This is a so-called 'reciprocal' license that means that if you use open source software that is licensed under the GPL license, then you must make that software open source.
- The GNU Lesser General Public License (LGPL) is a variant of the GPL license where you can write components that link to open source code without having to publish the source of these components.
- The Berkley Standard Distribution (BSD) License. This is a non-reciprocal license, which means you are not obliged to re-publish any changes or modifications made to open source code. You can include the code in proprietary systems that are sold.

5 (a) What is alpha, beta and acceptance testing.? Explain six stages of acceptance testing process. [6M]

- User or customer testing is a stage in the testing process in which users or customers provide input and advice on system testing.
- User testing is essential, even when comprehensive system and release testing have been carried out.
  - The reason for this is that influences from the user's working environment have a major effect on the reliability, performance, usability and robustness of a system. These cannot be replicated in a testing environment.
- Alpha testing
  - Users of the software work with the development team to test the software at the developer's site.
- Beta testing
  - A release of the software is made available to users to allow them to experiment and to raise problems that they discover with the system developers.
- Acceptance testing
  - Customers test a system to decide whether or not it is ready to be accepted from the system developers and deployed in the customer environment. Primarily for custom systems.

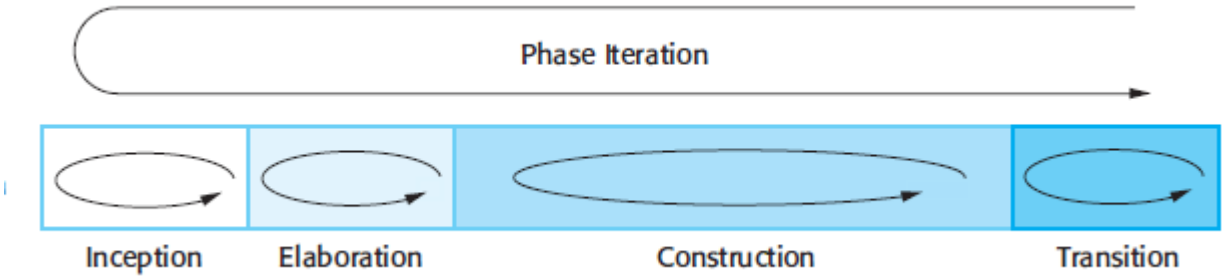


- Define acceptance criteria
- Plan acceptance testing
- Derive acceptance tests
- Run acceptance tests
- Negotiate test results
- Reject/accept system

5 (b) Explain development testing. Explain the three levels of granularity carried out in testing.[4M]

- Development testing includes all testing activities that are carried out by the team developing the system.
  - Unit testing, where individual program units or object classes are tested. Unit testing should focus on testing the functionality of objects or methods.
  - Component testing, where several individual units are integrated to create composite components. Component testing should focus on testing component interfaces.
  - System testing, where some or all of the components in a system are integrated and the system is tested as a whole. System testing should focus on testing component interactions.

6 Draw a neat diagram and explain the four phases and nine work flows of Rational Unified Process (RUP).[10M]



- RUP is divided into four phases, named:
- Inception
- Elaboration
- Construction
- Transition

Inception

- Overriding goal is obtaining buy-in from all interested parties
- Initial requirements capture
- Cost Benefit Analysis
- Initial Risk Analysis
- Project scope definition
- Defining a candidate architecture
- Development of a disposable prototype
- Initial Use Case Model (10% - 20% complete)
- First pass at a Domain Model

Elaboration

Requirements Analysis and Capture

- Use Case Analysis
  - Use Case (80% written and reviewed by end of phase)
  - Use Case Model (80% done)
  - Scenarios
    - Sequence and Collaboration Diagrams
    - Class, Activity, Component, State Diagrams
- Glossary (so users and developers can speak common vocabulary)
- Domain Model
  - to understand the problem: the system's requirements as they exist within the context of the problem domain
- Risk Assessment Plan revised
- Architecture Document

Construction

Focus is on implementation of the design:

- cumulative increase in functionality
- greater depth of implementation (stubs fleshed out)
- greater stability begins to appear
- implement all details, not only those of central architectural value
- analysis continues, but design and coding predominate

Transition

- The transition phase consists of the transfer of the system to the user community
- It includes manufacturing, shipping, installation, training, technical support and maintenance
- Development team begins to shrink
- Control is moved to maintenance team
- Alpha, Beta, and final releases
- Software updates
- Integration with existing systems (legacy, existing versions, etc.)

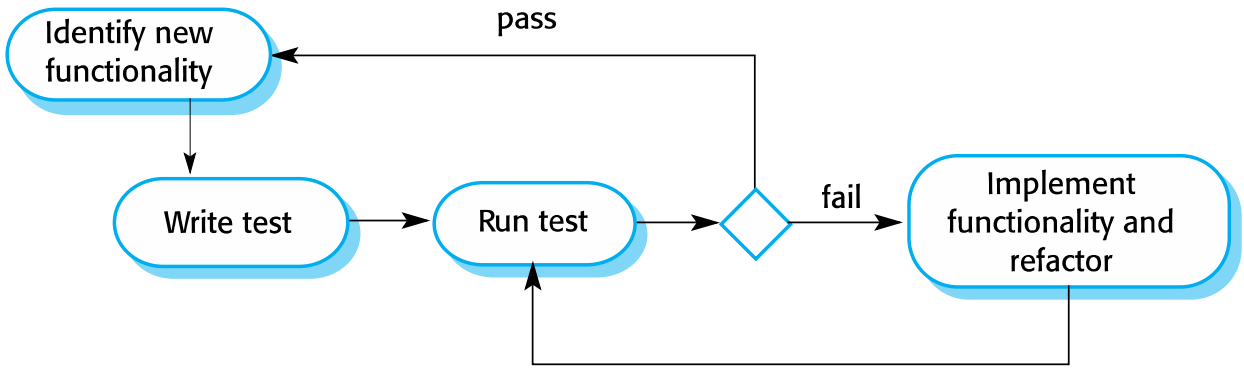
7 (a) List out all the guidelines for testing. [3M]

General testing guidelines

- Test software with sequences which have only a single value.
- Use sequences of different sizes in different tests.
- Derive tests so that the first, middle and last elements of the sequence are accessed.
- Test with sequences of zero length.
- Choose inputs that force the system to generate all error messages
- Design inputs that cause input buffers to overflow
- Repeat the same input or series of inputs numerous times
- Force invalid outputs to be generated
- Force computation results to be too large or too small.

7 (b) Explain Test-driven development (TDD), with a block diagram Explain TDD activities and benefits of TDD. [7M]

- Test-driven development (TDD) is an approach to program development in which you inter-leave testing and code development.
- Tests are written before code and 'passing' the tests is the critical driver of development.
- You develop code incrementally, along with a test for that increment. You don't move on to the next increment until the code that you have developed passes its test.
- TDD was introduced as part of agile methods such as Extreme Programming. However, it can also be used in plan-driven development processes.



TDD process activities

- Start by identifying the increment of functionality that is required. This should normally be small and implementable in a few lines of code.
- Write a test for this functionality and implement this as an automated test.
- Run the test, along with all other tests that have been implemented. Initially, you have not implemented the functionality so the new test will fail.
- Implement the functionality and re-run the test.
- Once all tests run successfully, you move on to implementing the next chunk of functionality.

Benefits of test-driven development:

Code coverage

- Every code segment that you write has at least one associated test so all code written has at least one test.

- Regression testing
  - A regression test suite is developed incrementally as a program is developed.
- Simplified debugging
  - When a test fails, it should be obvious where the problem lies. The newly written code needs to be checked and modified.
- System documentation

  The tests themselves are a form of documentation that describe what the code should be doing