Mid Term

Course Name: Object Oriented Software Engineering

Submitted To : Maam Sanaa Jehan

Submitted By : Qazi Shahzaib

ID: 13027

Object Oriented Software Engineering

MIDTERM EXAM

Question #1 (15 Marks)

Modelling is an important activity in everyday life. Explain how does it help to solve real life problems? Explain its importance in Software Engineering. Also explain that without modeling, what problems can be faced by a Software Developer during software development.

Question #2 (15 Marks)

Analysis Model is composed of 3 individual models. Name and explain all three. Mention how each one of them is represented.

Instructions for Assignment Submission

1. Write your names and Ids at the top of each paper of answer sheet.
2. Scan using simple scanner or the CamScanner of any android phone / Take Photo of each paper and save each photo with a number. E.g. photo of page 1 of answer sheet be saved with name 1.jpg, then 2.jpg and so on.
3. Make a PDF file of all the pictures and name it with your Roll Number, Name and Subject Name, e.g. "11512 - Sanaa Jeehan – OOSE".
4. Upload the file as it is or zipped.
5. Don't forget to check that the correct document is properly uploaded.

Question #1

Modelling is an important activity in everyday life. Explain how does it help to solve real life problems? Explain its importance in Software Engineering. Also explain that without modeling, what problems can be faced by a Software Developer during software development.

Ans:

We build models so that we can better understand the system we are developing.

Modelling plays the vital role in our daily life and it helps in overcoming our daily life problems as we plan and manage our problems we model it and we try to overcome it before it occurs. Every step is defined clearly in modeling so the chances of getting error comes low and we can easily overcome the real problems by jut doing it modelling correctly. For exampling when you model your daily routine so you know what to do on what time so the chances of doing error comes low and its much time saving as you modeled what to do on which time.

Importnace Of Modeling In Software Engineering:

Models are forms of description often adopted in software development. They are preoccupation used to constitute and communicate what is important, devoid of unnecessary detail, and to help the developers to deal with the complication of the problem being look over or the solution being developed.

Modelling is used in other forms of design and engineering. For instance, architects develop different models of buildings – some superscribe structures, others materials, others ergonomics, and so on. The same happens with modelling in software development: some models are used to capture properties of the problem of domain, like as key elements of the business or how it will proceeding the work, while other models are used to consider different feature of the software, such as how the code is divided up and organised, or how various elements of the software communicate and work together. Each model is an abstract representation of some view of the system, and such views may change as the development process unfolds.

in software development, we build models from different perspectives. In particular, we can distinguish between the following modelling types.

Domain modelling is concerned with understanding and modelling context information for a specific problem, independently of a decision to use a software system to deal with that problem. A domain model is a representation of the main concepts in the real-world problem context – for instance, a business under consideration. A domain model does not necessarily assume a software solution.

Specification modelling assumes that a software system will deal with the need in context. A specification model represents software elements used in the software solution to a problem, and is mainly concerned with the definition, at a high level of abstraction, of the services provided by the software.

Design modelling describes the software system itself, with the allocation of responsibilities to its various parts, and its behaviour and control flow.

Problems faced by a Software developer :

Problems which are faced by a software developer during the software development are as follow:

Rapid Technology Advancement

Every technology advancement is a blessing for the IT industry. But at the same time, technology evolving at a phenomenal rate leads to an added pressure for software development professionals to leverage these upcoming technology trends in software product development to gain a cutting edge over competitors and stand out in the market.

Increasing Customer Demands

Software projects are generally conceptual and are aimed at designing and developing software products that meet varied customer demands. To develop even the simplest application or product, developers must clearly understand the underlying business concept and bring in the required features to satisfy the growing customer demands.

Time Limitation

Software development is a time-game. Developers work under pressured environments and strive to complete project requirements within strict and scanty

timelines. This is especially a challenge when working with international clients on multiple time-zones. Time constraints often bring down efficiencies of development teams and lead to mediocre quality software products in the end.

Limited resources

Another challenge faced by majority of software development companies is a lack of resources or IT infrastructure to execute projects effectively. This could mean a lack of high performance software development tools, powerful computing platforms, inefficient data storage architectures or improper networks and connectivity. Such hindrances bring down productivity and performance of software development teams and impact the overall result.

Conflict With team

In a classic software development project, interpersonal conflicts occur inevitably between software development and testing teams. Several factors contribute to such conflicts like working under high performance pressure, different mindsets, difference in job roles and the very opposite nature of development and testing. If not controlled and managed effectively, these conflicts could hamper the overall project adversely.

To succeed in a dynamic software industry that is driven by changing technology trends and challenged by multiple internal and external factors, your development teams must have a clear understanding of the problems that lie ahead of them and a roadmap to overcome them.

Implementing software development best practices could help reduce these problems to a large extent. Introducing a DevOps team to handle the testing-development crisis, adopting Cloud for seamless network and infrastructure and constantly enhancing the technical knowhow of your software development teams could help you fight some of these major software development challenges and establish a good position in the software market.

Question #2

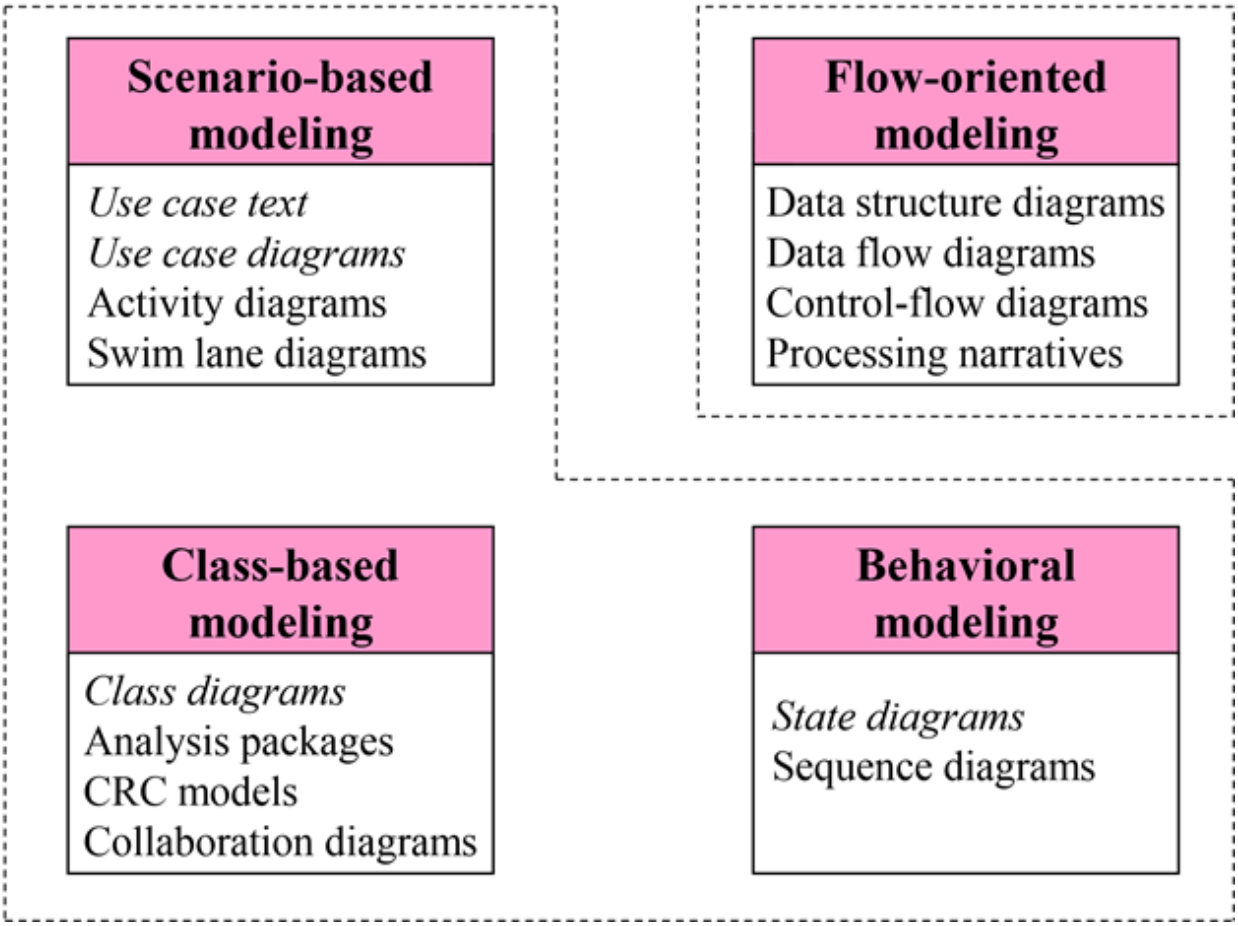
Analysis Model is composed of 3 individual models. Name and explain all three. Mention how each one of them is represented.

Ans:

Goals of Analysis Model:

- Provides the first technical representation of a system
- Is easy to understand and maintain
- Deals with the problem of size by partitioning the system
- Uses graphics whenever possible
- Differentiates between essential information versus implementation information
- Helps in the tracking and evaluation of interfaces
- Provides tools other than narrative text to describe software logic and policy

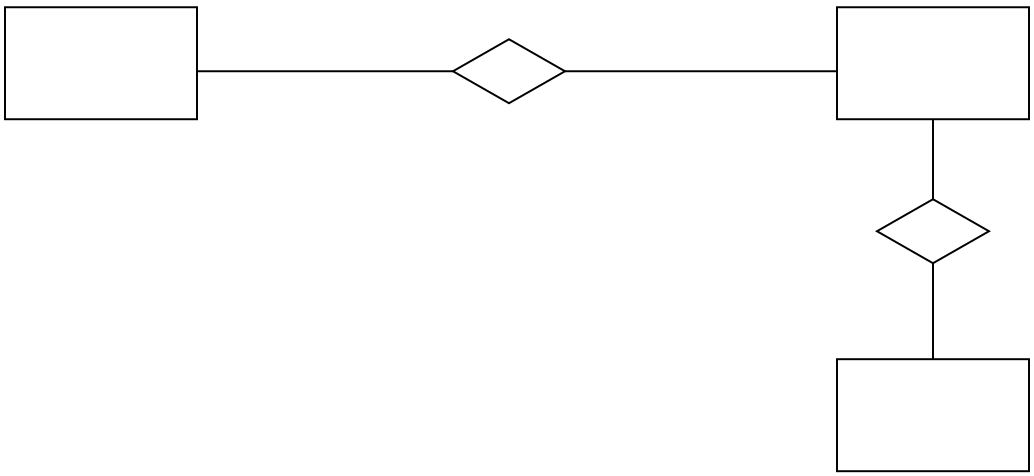
Elements of the Analysis Model:



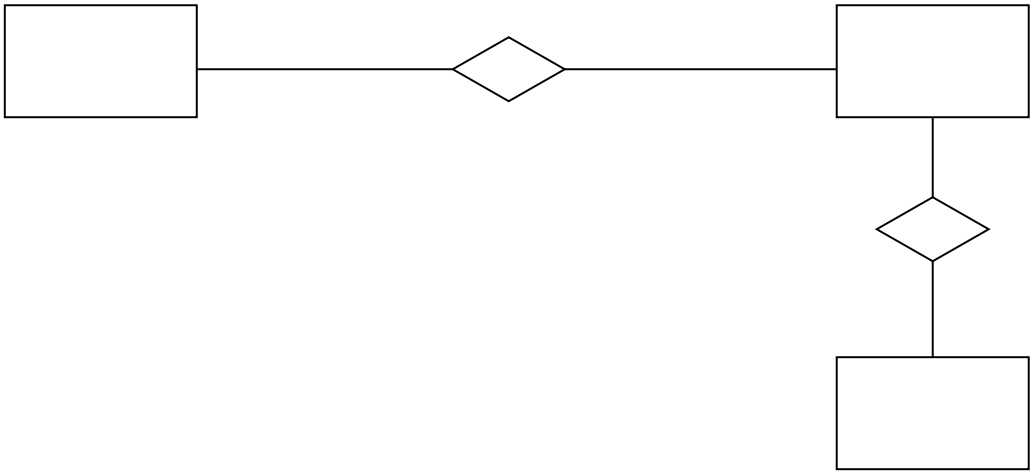
Flow Oriented Model:

Identify the following items

- Data objects (Entities)
- Data attributes
- Relationships
- Cardinality (number of occurrences)



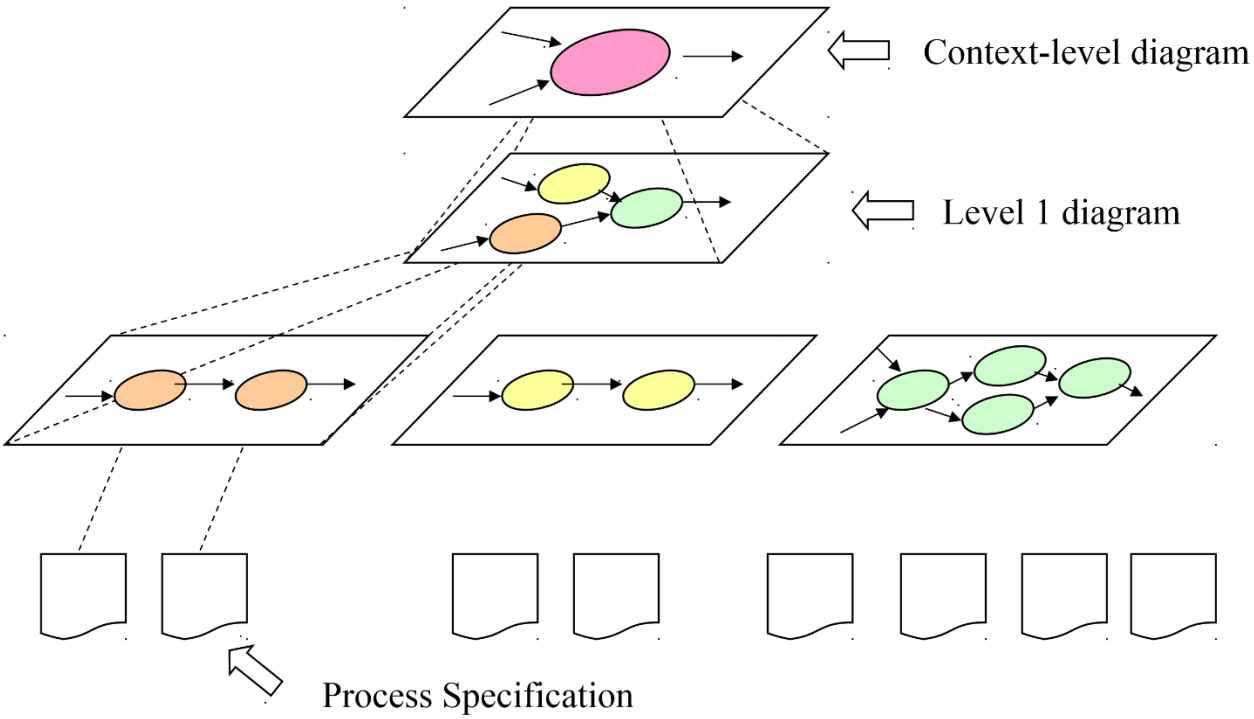
- Data objects (Entities)
- Data attributes
- Relationships
- Cardinality (number of occurrences)



Data Flow and Control:

- Data Flow Diagram
  - Depicts how input is transformed into output as data objects move through a system
- Process Specification
  - Describes data flow processing at the lowest level of refinement in the data flow diagrams
- Control Flow Diagram
  - Illustrates how events affect the behavior of a system through the use of state diagrams

Diagram Layering and Process Refinement



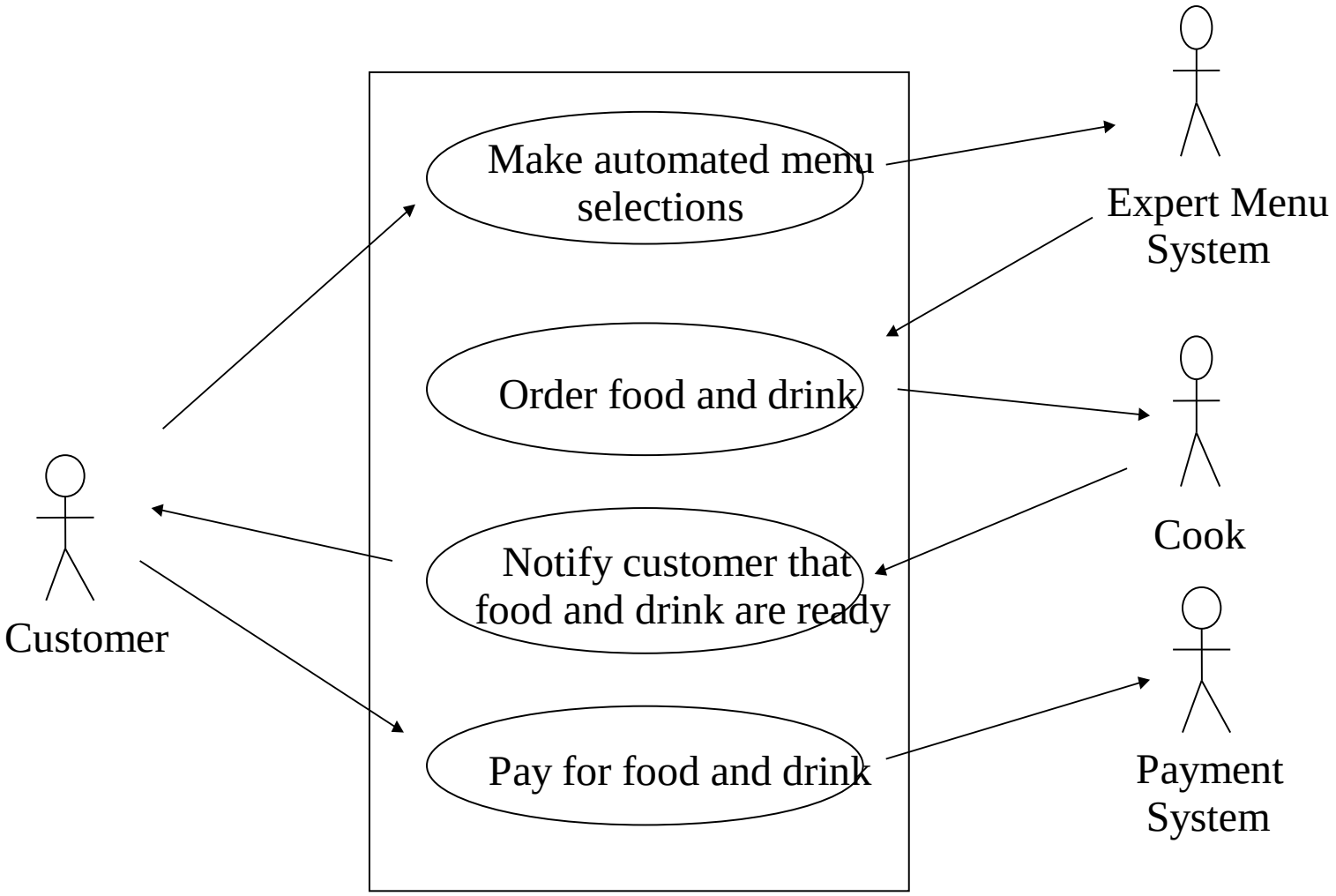
Scenerio Based Modeling

Writing Use Cases

- It is effective to use the first person “I” to describe how the actor interacts with the software

- Format of the text part of a use case

Use-case title:
Actor:
Description: I …

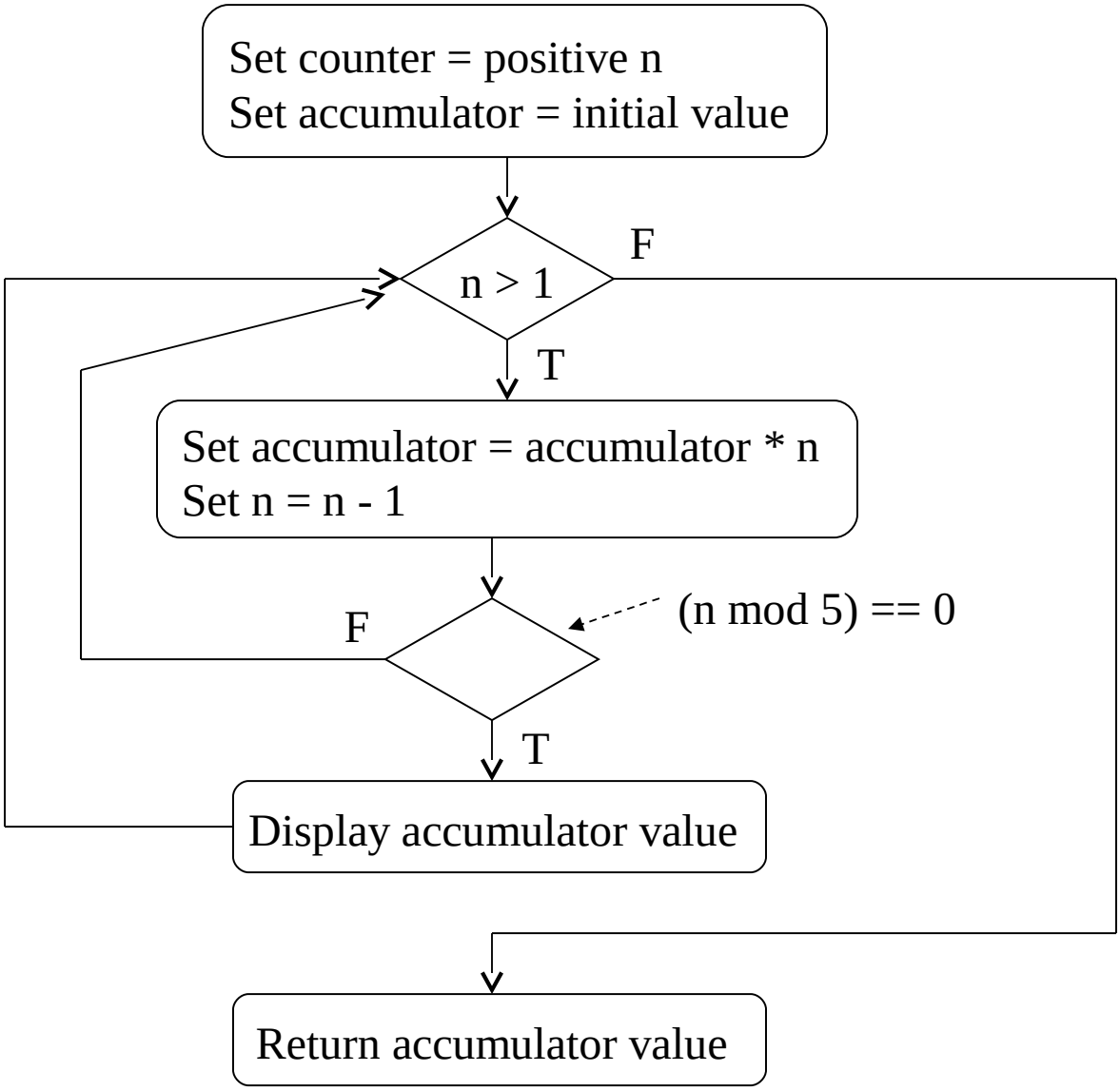


Activity Diagrams

- Supplements the use case by providing a graphical representation of the flow of interaction within a specific scenario
- Uses flowchart-like symbols

- Rounded rectangle - represent a specific system function/action
- Arrow - represents the flow of control from one function/action to another
- Diamond - represents a branching decision
- Solid bar – represents the fork and join of parallel activities

Example



Class Based Modeling:

Identifyng analysis of class

- Perform a grammatical parse of the problem statement or use cases

- Classes are determined by underlining each noun or noun clause
- A class required to implement a solution is part of the solution space
- A class necessary only to describe a solution is part of the problem space
- A class should NOT have an imperative procedural name (i.e., a verb)
- List the potential class names in a table and "classify" each class according to some taxonomy and class selection characteristics
- A potential class should satisfy nearly all (or all) of the selection characteristics to be considered a legitimate problem domain class

Potential classes General classification Selection Characteristics

General classifications for a potential class

- External entity (e.g., another system, a device, a person)
- Thing (e.g., report, screen display)
- Occurrence or event (e.g., movement, completion)
- Role (e.g., manager, engineer, salesperson)
- Organizational unit (e.g., division, group, team)
- Place (e.g., manufacturing floor, loading dock)

- Structure (e.g., sensor, vehicle, computer)

- Six class selection characteristics

1)Retained information

- Information must be remembered about the system over time

1)Needed services

- Set of operations that can change the attributes of a class

1)Multiple attributes

- Whereas, a single attribute may denote an atomic variable rather than a class

1)Common attributes

- A set of attributes apply to all instances of a class

Common operations

- A set of operations apply to all instances of a class

1)Essential requirements

- Entities that produce or consume information

Defining Operations Of a class:

- Operations define the behavior of an object
- Four categories of operations
  - Operations that manipulate data in some way to change the state of an object (e.g., add, delete, modify)

- Operations that perform a computation
- Operations that inquire about the state of an object
- Operations that monitor an object for the occurrence of a controlling event

- An operation has knowledge about the state of a class and the nature of its associations
- The action performed by an operation is based on the current values of the attributes of a class

Using a grammatical parse again, circle the verbs; then select the verbs that relate to the problem domain classes that were previously identified

Behaviour Modeling:
Identify the events found within th use cases and implied by the attributes in the class diagrams.
Build a state diagram for each class and if useful for the whole software system

Identifying the events:

- An event occurs whenever an actor and the system exchange information
- An event is NOT the information that is exchanged, but rather the fact that information has been exchanged
- Some events have an explicit impact on the flow of control, while others do not
  – An example is the reading of a data item from the user versus comparing the data item to some possible value

Building a state diagram

- A state is represented by a rounded rectangle
- A transition (i.e., event) is represented by a labeled arrow leading from one state to another

- Syntax: trigger-signature [guard]/activity

- The active state of an object indicates the current overall status of the object as is goes through transformation or processing
  - A state name represents one of the possible active states of an object
- The passive state of an object is the current value of all of an object's attributes
  - A guard in a transition may contain the checking of the passive state of an object

Example of state diagram.