1. ***What is System?***

Ans. System is an abstraction of a complex interacting set of elements, for which it is possible to identify a boundary, and environment, inputs and outputs, a control mechanism and some process or transformation that the system achieves.

***2.What are the element of a system?***

Ans. Elements of a system are-

1. Boundary and environment
2. Input, output and interface
3. Sub-systems
4. Control in systems
5. Feedback
6. Feed-forward
7. Emergent properties

***What is the difference between information and data?***

Ans. Information is the facts that have been selected as relevant to a purpose and then organized or processed in such a way that they have meaning for that purpose. Information is conveyed by message and has meaning. On the other hand, data is raw facts that not yet identified as relevant to any particular purpose.

***What is the purpose of MIS?***

Ans. The purpose of MIS is to

-Extract data from existing operational system.

-and analysis or combine it to give managing in information about the part of the organization for which they were responsible.

CHAPTER 2

1. **Define quality.**

Ans. Quality means fitness for purpose. Since it can be hard to identify the purpose.

1. **What is Stakeholder?**

Ans. A stakeholder has an interest in a project because they are(or will be) affected by its programs of by its results.

1. **What are the main underline causes of problem in information system?**

Ans. The main underline causes of problem in IS are discus in the following perspective

1. And end users perspective-
2. A software product that is much talked about but never released to its intended users.
3. System may fail to meet the criterion of usability in a numbers of ways such as poor interface design, inappropriate or illogical sequence of data entry, incomprehensive error messages, unhelpful help, poor response times and unreliability in operation
4. System is very pretty but it does not do anything useful.
5. A clients perspective-
6. Some clients also have the power to stop a project once it is underway.
7. A developers perspective.
8. **Difference between Quality problem and productivity problem.**

Ans.

|  |  |
| --- | --- |
| **Quality problem** | **Productivity problem** |
| It means fitness for purpose. | It relates to rate of progress of a project and the resources that is consumes along the way. |
| For quality assurance it is necessary to know  -the purpose of system and  -how to measure its fitness. | For productivity it is necessary to know  -if the product delivered.  -it it is delivered in time.  -if it affordable |

CHAPTER 3:

**Phases of waterfall life cycle.**

1. System engineering
2. Requirements analysis.
3. Design.
4. Construction
5. Testing
6. Installation
7. Maintenance
8. **Advantages & disadvantages of traditional waterfall life cycle.**

Ans. Advantages:

1. Teams with specialized skill can be assigned to tasks in particular phases.
2. Progress can be evaluated at the end of each phase.
3. Attendant risk can be controlled and managed.

Disadvantages:

1. Real project rarely follow a simple sequential life cycle include.
2. Interactions are almost inevitable.
3. The elapsed time between inception and delivery is frequently too long.
4. It is unresponsive to changes in the technology or requirements.
5. **Prototyping**

In software development a prototype is a system or a partially complete system that is build quickly to explore some aspect of a system requirements and that is not intended as the final working system.

Main system require to prepare prototype

1. Perform an initial analysis.
2. Define prototype objectives.
3. Specify prototype.
4. Construct prototype.
5. Evaluate prototype and recommend change.
6. **USDP**

The Unified Software Development Process (USDP) (Jacobson et al, 1999) reflects the current emphasis on iterative and incremental life cycles.

A development cycle for the USDP comprises four phases-

1. Inception
2. Elaboration
3. Construction
4. Transition

**Incremental Development**

Incremental development involves some initial analysis to scope the problem and identify the major requirements. The requirements are that reviewed and those that deliver most benefit to the client become the focus of the first increment of development and delivery. The installation of the first increment provides valuable feedback to the development team and informs the development of the second increment and so on.

CHAPTER 5

1. **What is the difference between model and diagram?**

Ans. **Model**:Like any map, models represent something also. Models are usually both abstract and visible. They are useful in several different ways, precisely because they differ from the things that they represent-

1. A model is quicker and easier to build.
2. A model can be used in simulations to learn more about the thing it represents.
3. A model can evolve as we learn more about a task or problem.
4. We can choose which details to represent in a model and which to ignore. It is an abstraction.
5. A model can represent real or imaginary things form any domain.

**Diagram**: Diagrams are used to build models of system in the systems in the same way as architects use drawings and diagrams to model buildings. Diagrammatical models are used extensively by system analysts and designers in order to-

1. Communicate ideas
2. Generate new ideas and possibilities
3. Test ideas and make predictions
4. Understand structures and relationships

A model provides a complete view of a system at a particular stage and form a particular perspective.

1. **What are the basic elements of UML model diagram?**

Ans. UML diagrams are made up of four elements-

1. Icons
2. Two dimensional symbols
3. Paths
4. Strings

UML diagrams are graphs composed of various kinds of shapes, known as nodes, joined together by lines, known as paths.

* Different models present different views of the system. Booch et al (1999) suggest five views to be used with UML

1. The use case view
2. The design view
3. The process view
4. The implementation view
5. The development view

**What is the UML notation for each of the following package, sub-system and model?**

Ans.

Use case

Model

Campaign

Management

Use Cases

Package

Model

Sub-system

Fig: UML notation for packages, sub-systems and models.

1. **Draw a simple Activity diagram.**

**Activity Diagram:**

Fig: Activity diagram for the Activity write chapter.

(not Satisfied)

Add Re-exercise

To Bibliography

Add Exercise

Revise Draft

Produce First Draft

Plan Chapter

Activity diagram can be used to model different aspects of a system. At a high level, they can be used to model business activates in an existing or potential system.

Activity diagram can be used for the following purpose:

1. To model a task
2. To describe a system function that is represented by a use case.
3. In operation specifications, to describe the logic of an operation.
4. In USDP to model the activities that make up the life cycle.
5. **Guard Condition**

Ans. Guard condition is a Boolean expression associated with a transition that is evaluated at the time the event fires. The transition only takes place if the condition is true. A guard condition is a function that may involve parameters of the triggering event and also attributes and links of the object that owns the state chart.

CHAPTER 6

**List the name of the fact finding techniques.**

Ans. There are 5 main fact finding techniques that are used by analyst to investigate requirements-

1. Background reading
2. Interviewing
3. Observation
4. Document Sampling
5. Questionnaires

\*\*\*\*What is use case? What is the purpose of use case?

Ans. Use cases are descriptions of the functionality of the system from the users’ perspective.

Use case diagrams are used to show the functionality that the system will provide and to show which users will communicate with the system in some way to use that functionality.

\*\*\*\*\*\*What is Stereotypes? Describe include & extend.

Ans. Stereotype:

A stereotype is a special use of a model element that is constrained to behave in a particular way Stereotypes can be show by using a keyword. Such as ‘extend’ or ‘include’ in matched quillements like <<extend>>. Stereotype can also be represented using special icon. The actor symbol in use case diagrams is a stereotyped icon- an actor is a stereotyped class and could also be shown as a class rectangle with the stereotype <<actor>> above the name of the actor.

**<<extend>>** is used when we wish to show that a use case provides additional functionality that way be required in another use case.

**<<include>>** applies when there is a sequence of behavior that is used frequently in a number of use cases and we want to avoid copying the same description of it into each use case in which it is used.

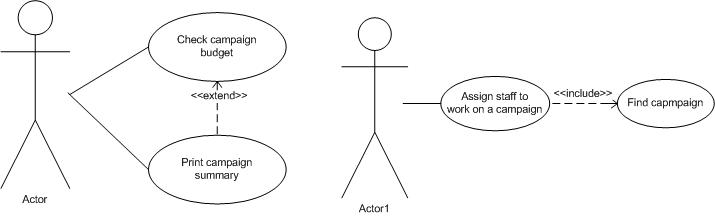


Fig: use case diagram showing <<extend>> and <<include>>

1. What is a collaboration diagram?

Ans. A collaboration diagram shows an interaction between objects and the context of the interaction in terms of the link between the objects.

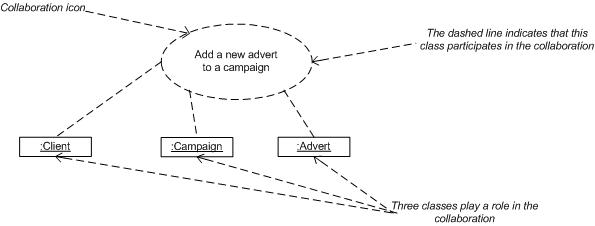


Fig: Collaboration for add a new advert to a campaign.

1. Define boundary class, entity class and control class?

Ans. **Boundary class**:

Boundary class is a stereotyped class that provides an interface to users or other system.



Fig: Alternative notations for boundary class stereotype.

**Entity class:**

Entity class is a stereotyped class that represents objects in the business domain model.



Fig: Alternative notation for an entity class.

**Control class:**

Control class is a stereotyped class that controls the interaction between boundary classes and entity classes.

Fig: Alternative notation for a control class.

**A link between instances.**

Association:- Association is a logical connection. usually between different classes although in some circumstances a class can have an association with itself. An association describes possible links between objects and may correspond either to logical relationships in the application domain or to message paths in software.

\*\*\*How does a collaboration diagram differ from class diagram?

Ans. A collaboration diagram shows only those classes that collaborate to provide the functionality of a particular use cases (or operation); the links that are shown are those that are required for that purpose.

But a class diagram typically shows all the classes in a particular package and all the associations between them.

CHAPTER 8

Distinguish between composition from aggregation.

Ans.

|  |  |
| --- | --- |
| Composition | Aggregation |
| 1. Composition is a type of abstraction that encapsulates groups of classes that collectively have the capacity to be a reusable sub-assembly. Represent the whole and the other part of the whole. 2. Symbol 3. A part can belong only one composition. | 1. Aggregation represent a whole part association between two or more objects. 2. Symbol 3. A part can belong more than one aggregation. |

1. What are the advantages of components?

Ans. The use of component saves time and work.

1. Pattern:

A pattern is an abstract solution to a commonly occurring problem in a given content

CHAPTER 9

1. What is interaction sequence diagram?

Ans. Sequence diagram or interaction sequence diagram shows an interaction between objects arranged in a time sequence. Sequence diagram can be drawn at different levels of detail and also to meet different purpose ant several stages in the development life cycle.

* Interaction diagrams are two types-

1. Sequence diagram
2. Collaboration diagram
3. Difference between sequence diagram and collaboration diagram.

|  |  |
| --- | --- |
| Sequence diagram | Collaboration diagram |
| 1. Sequence diagram shows an interaction between objects arranged in a time sequence. 2. Sequence diagrams have a time dimension. 3. It does no show the link between object. | 1. Collaboration diagram shows an interaction between object and the content of the interaction in terms of the links between the objects. 2. Don not have time dimension. 3. It shows the link between objects. |

1. What is an object lifeline and focus of control?

**Object lifeline**: An object lifeline represents the existence of an object during an interaction represented in a sequence diagram.

**Focus of control**: Focus of control indicates which operation is executing at a particular stage in an interaction represented in a sequence diagram.

1. ***Difference between Synchronous and Asynchronous message.***

|  |  |
| --- | --- |
| Synchronous message | Asynchronous message |
| 1. Synchronous message or procedural call is shown with a full arrowhead. 2. It causes the invoking operation to suspended execution until the focus of control has been returned. | 1. Asynchronous message is shown with an open arrowhead. 2. It does not cause the invoking operation to Holt execution while it awaits a return. |

1. ***Callback:***It may be necessary for an operation that has been invoked asynchronously to notify the object that invoked it when it has terminated. This is done by explicitly sending a message (known as callback) to the originating object.

***CHAPTER 10***

***Difference between algorithmic and non-algorithmic technique to operation specification.***

Ans.

|  |  |
| --- | --- |
| **Algorithmic technique** | **Non-Algorithmic technique** |
| 1. An algorithm defines the step-by-step behavior of an operation. 2. An algorithm also specifies the sequence in which the steps are performed. 3. Generally do not prefer in object-oriented development. 4. Describe the internal logic.   eg. Activity Diagram. | 1. A non-algorithmic approach defines only inputs and results. 2. If does not specifies the sequence. 3. Generally preferred in object-oriented because Non-algorithmic methods of operation specification emphasize encapsulation. 4. Do not describe.   eg. Decision table. |

***What are the main component of OCL operation?***

Ans. The UML has also a formal language known as Object Constraint Language(OCL). which is intended mainly for specifying general constraints on a model.

Most OCL statement consists of the following three components-

1. A context within which the expression is valid (for example, a specified class).
2. A property within the context to which the expression applies(for example and attribute of the specified class).
3. An operation that is applied to the property (for example a mathematical expression that tests the value of the attribute)
4. ***Define Activity Diagram.***

Activity Diagram: It is a version of state chart diagram that focuses on a flow of activity driven by internal processing within an object rather than by events that are external to it. In an activity diagram most(or all) states are action states(also called activities) each of which represent the execution of an operation. Activity diagrams can be used to specify the logic or procedurally complex operations.

CHAPTER 11

***What do you mean by state chart?***

Ans. The state chart is a versatile technique and can be used within and Object-Oriented approach for other purpose than the modeling of object life cycles. A state chart must have at least one initial state.

***Mention the important link between state-chart and iteration diagram.***

Ans. There is an important link between state chart and iteration diagrams. A model of state behavior in a state chart captures all the possible responses of a single object to all the use cases in which it is involved. By contrast a sequence or a collaboration diagram captures the responses of the entire object that are involved in a single use case.

CHAPTER 12

***Difference between Cohesion and Coupling.***

* Cohesion: Cohesion is the degree to which the responsibilities of a single component form a meaningful unit.
* Coupling: Coupling describes the relationship between software components.
* goal- Reduce coupling increase cohesion.

***List 12 quality criteria for good design?***

Ans. Functional, efficient , economical, reliable, secure, flexible, general, buildable, manageable, maintainable, usable, reusable.

***What make good analysis?***

1. Correct scope
2. Completeness
3. Correct content and
4. Consistency.

CHAPTER 13

***What is software architecture?***

Ans. A software architecture is a description of the sub-systems and components of a software system and the relationship between them.

***What is Layering and partitioning?***

Ans. There are two general approaches to the division of a software system into subsystems. These are known as Layering and partitioning.

Layering- The different sub-systems usually represent different levels of abstraction.

Partitioning- Usually means that each subsystem focuses on different aspect to the functionality of the system as a whole.

Guidelines on the development of Layered architecture:

1. Define the criteria by which the application will be grouped into layers.
2. Determine the member of layers.
3. Name the layers and assign functionality to them.
4. Specify the services for each layer.
5. Refine the layering by iterating through steps i to l.
6. Specify interfaces for each layer.
7. Specify the structure of each layer.
8. Specify the communication between adjacent layer.
9. Reduce the coupling between adjacent layer..

* Open layered architectures are more difficult to maintain because each layer communicate with all lower layers hence increasing the degree of coupling in the architecture. A change to one layer amy ripple to many layers.
* A closed layer architecture may require more processing as messages have to be passed through interviewing layers.

CHAPTER 14

* What levels of visibility may be assigned to an attribute or an operation?

Ans. Public, Private and Protected.

* Attributes should be designated private to enforce encapsulation.

Attribute:

Attribute is an element of the data structure that together with operation, defines a class. Describes some property of instances of the class.

* An attributes data type is declared in UML using the following system:

name ‘:’ type-expression ‘=’ initial-value ‘{’ property-string ‘}’

Data type

attribute name

balance : Money =0.00

* Operation: Operation in an aspect of the behavior that defines a class an element of the services that are provided by a class; a specification of an element of system functionality that will be implemented as a method of an object.

The syntax used for an operation is-

operation name’(’ parameter-list ’)’ ‘:’ return-type expression

* Object visibility: Visibility is an UML modeling element (eg, attributes or operations) may be designated with different levels of accessibility or visibility.

|  |  |  |
| --- | --- | --- |
| **Visibility symbol** | **Visibility** | **Meaning** |
| + | public | The feature is directly accessible by an instance of any class |
| - | private | The feature may only be used by an instance of the class that include it. |
| # | protected | The feature may be used them by instances if the class that includes it or of a subclass or descendant of that class. |
| ~ | package | The feature is directly accessible only by instance of a class in the same package. |

**Interfaces**: An interface in UML is a group of externally visible (i.e. public) operations. An interface is equivalent to associations and only abstract operation.

**Further design guidelines:**

1. Design clarity: design should be mode as easy as possible.
2. Don’t over design
3. Control Inheritance Hierarchies.
4. Keep message and operation simple
5. Design volatility.
6. Evaluate by scenario
7. Design by Delegation
8. Keep classes Separate

**Association:** Association is a logical connection, usually between different classes.

1. One-to-one association
2. One-to-many
3. Many-to-many

Integrity constraint:

A constraint that has to be enforced to ensure that the information system holds data that is manually consistent and is manipulated correctly.

* **Referential integrity** ensures that an object identifier in one object actually refers to an object that exists.
* **Dependencies constraints** ensure that attribute dependencies, values are maintained consistently where the value of one attribute is calculated from other attributes are maintained consistently.
* **Domain integrity** ensures that attributes only hold permissible values.

**Normalization:** Normalization is a technique that group’s attributes based upon functional dependencies according to several rules to produce normalized data structures that are largely redundancy.