**Chapter 6**

**1) Describe to a business person the multiplicity of a relationship between two classes.**

The *multiplicity of a relationship* represents how one instance of an object (such as an employee, department, or other entity) can be associated with another object. The main function of multiplicity is to document exactly how many instances of the first object can be associated with a certain number of the second object. An example is “an employee can have 0..\* (Zero to Many) children. These numbers denote the min and max instances that can be related. Pg. 217

**2) Why are assumptions important to a structural model?**

I am still looking for this one. I will repost when I find it. What it may be is….

One of the easiest ways to use CRC cards in developing a structural model is through anthropomorphism-pretending that the classes have human characteristics. This is done by having analyst and/or the user role-play and pretends that they are an instance of the class being considered.

**3) What is an association class?**

An association class has its own attributes and operations. It is shown as a rectangle attached by a dashed line to the association path, and the rectangle’s name matches the label of the association.

**4) Contrast the following sets of terms:**

Object – Objects represent the things, ideas, or concepts contained in the domain of the problem. They also allow the representation of the relationships between the things, ideas, or concepts.

Class – Is a general template that we use to create specific instances or objects in the problem domain. All objects of a given class are identical in structure and behavior but contain different data in their attributes. There are 2 different kinds of classes: *concrete* and *abstract.*

**Concrete classes** are used to create objects.

**Abstract classes** are not actually real classes; they are just useful abstracts

Instance – In object-oriented programming with classes, an instance variable is a variable defined in a class (i.e. a member variable), for which each object in the class has a separate copy

**5. Give three examples of derived attributes that may exist on a class diagram. How would they be denoted on the class diagram?**

Three examples of derived attributes that may exist on a class diagram are age (derived from today’s date minus date of birth), days employed (derived from today’s date minus day started), and discount price (derived from original price minus discount rate calculated). Derived attributes are denoted with a “/” in front of their name (i.e. - /age).

**6) What are the different types of visibility? How would they be denoted on a class diagram?**

*Visibility* relates to the level of information hiding to be enforced for a specific attribute in a class diagram.

Three types are:

1. **Public:** An attribute that is not hidden from any other object. Other objects can modify its value. It is represented by a “**+**” sign in a class diagram
2. **Protected:** An attribute that is hidden from all other classes except its immediate subclasses. It can be represented by a “**#**” sign in a class diagram.
3. **Private:** This is an attribute that is hidden from all other subclasses, and is dentoed by a “**-**“ sign in a class diagram.

**Pg. 213**

**7) Draw the relationships that are described by the following business rules. Include the multiplicities for each relationship.**

-A patient must be assigned to only one doctor and a doctor can have one or many patients.

doctor

patient

1..\*

1

-An employee has one phone extension, and a unique phone extension is assigned to an employee.

Phone extension

employee

1

1

-A movie theater show at least one movie, and a movie can be shown at up to four other movie theatres around town.

movie

Movie theatre

1..4

1..\*

-A movie either has one star, two costars, or more than ten people starring together. A star must be n at least one movie.

Costars

Movie

1,2,11..\*

1..\*

**8) How do you designate the reading direction of a relationship on a class diagram?**

The reading direction of a relationship in a class diagram is denoted with an arrow pointing into receiving end of the relationship.

**9) For what is an association class used in a class diagram?**

These are depictions on the diagram by drawing line between classes that are labeled using a verb phrase or role name that represent a relationship between multiple classes or a class and itself.

**10. Give two examples of aggregation, generalization, and association relationships. How is each type of association depicted on a class diagram?**

Two examples of **aggregation** would be:

1. Desk is part of Office
2. Wheel is part of Vehicle

**Aggregation** is depicted by placing a diamond nearest the class representing the aggregation (i.e. - Office and Vehicle), and lines are drawn from the arrow to connect the classes that serve as its’ parts (i.e. – Desk and Wheel).

Two examples of **generalization** would be:

1. Cardinal is a kind of Bird, which is a kind of Animal
2. General Practitioner is a kind of Physician, which is a kind of Person

**Generalization** is depicted with a solid line from the subclass to the superclass and a hollow arrow pointing at the superclass.

Two examples of **association** would be:

1. Invoice is associated with Purchase Order (and vice versa)
2. A Pilot flies an Aircraft

An **Association** is depicted by drawing lines between classes.

**11) Identify the following operations as constructor, query or update. Which operations would not need to be shown in the class rectangle?**

A *constructor* operation creates a new instance of a class. They are not shown on class diagrams.

A *query* operation makes information about the state of an object available to other objects, but does not alter the object in any way. If the query just requests information, they are not shown on a class diagram.

An *update* operation changes the value of some or all of the object’s attributes, which may result in a change in the object’s state.

* 1. Calculate employee raise (raise percent) Query-Do not show
  2. Calculate sick days () Query-Do not show
  3. Increment number of employee vacation days () Update
  4. Locate employee name () Query-Do not show
  5. Place request for vacation (vacation day) Update
  6. Find employee address () Query-Do not show
  7. Insert employee () Constructor-Do not show
  8. Change employee address () Update-Do not show
  9. Insert spouse () Constructor-Do not show

**Pg. 215-216**

**Chapter 7**

**1)How is behavioral modeling related to structural modeling?**

*Behavior Models* depict the internal view of the business process that a use case is describing. They are used to show how objects will work together to form a collaboration to support each of the use cases that are created.

*Structural Models* represent objects and the relationships between them.

The structural models sort of lay out the foundation for the behavior models. The behavior models (the business process) cannot be accurately represented if the underlying relationships between objects are not set up properly.**Pg. 239**

**2. How does a use case relate to a sequence diagram? A communication Diagram?**

A use case relates to a sequence diagram in that the sequence diagram illustrates the objects that participate in a scenario of a use case and the messages that pass between them over time. Similarly a communications diagram typically depicts the communications which occur inside of a use case scenario.

3) **Contrast the following sets of terms:**

State: behavior

State – state of an object is defined by the value of its attributes and its relationships with other objects at a particular point in time

Behavior – operations; an action that an object can perform

Class: object

Class – a generic framework that defines an object

Object - an instance of a class

Action: activity

Action – an atomic, nondecomposable process that cannot be interrupted

Activity – a nonatomic, decomposable process that can be interrupted

Use case: scenario

Use case – a comprehensive set of scenarios in which an activity can take place

Scenario – a single way in which a use case can unfold

Method: Message

Method – an action that an object can perform

Message – information sent to objects to tell an object to execute one of its behaviors

**4) Why is iteration important when creating a behavioral model?**

As the models are created, it is not unusual to make mistakes or changes to the functional and structural models.

**5. What are the main building blocks for the sequence diagram? How are they represented on the model?**

The main building blocks for the sequence diagram are:

1. ***Actor*** – a person or system that derives benefit from system. Depicted either as a stick figure or, if a nonhuman actor is involved, as a rectangle with <actor> in it.
2. ***Object*** – participates in a sequence by sending/receiving messages. Depicted with a rectangle across the to of the diagram and inside is “anObject : aClass”.
3. ***Lifeline*** – denotes the life of an object during a sequence. Depicted with a dashed line and contains an X at the point at which the class no longer interacts.
4. ***Execution Occurrence***– denotes when an object is sending or receiving messages. Depicted with a long narrow rectangle placed atop a lifeline.
5. ***Message*** – conveys information from one object to another. An operation call is labeled with the message being sent and a solid arrow, whereas a return is labeled with the value being returned and shown as a dashed arrow.
6. ***Object destruction*** – shows end of objects lifeline. An X is placed at the end of an object’s lifeline to show that it is going out of existence.

**6. How do you show that a temporary object is to go out of existence on a sequence diagram?**

A *temporary object* is one that is created by a “permanent” object within a sequence diagram. A *sequence diagram* illustrates the objects that participate in a use case and the messages that pass between them over time for ONE use case. It is a dynamic model that shows explicit sequence of messages that are passed between objects in a defined interaction.

\*An X is placed at the end of the lifeline of the *temporary object* at the point it is destroyed and no longer used.

**Pg. 240-241**

**7. Do lifelines always continue down the entire page of a sequence diagram?**

Lifelines do not always continue to the bottom of a sequence diagram. If an object involved in a sequence does not exist for the entire duration of the sequence diagram it does not extend to the bottom of the diagram.

**8. Describe the steps used to create a sequence diagram.**

1. Set the context of the sequence diagram. The context of the diagram can be a system, a use
2. Identify which objects will participate – the objects that interact with each other during the use-case scenario.
3. Set the lifeline for each object. You need to draw a vertical dotted line below each class to represent the class’s existence during the sequence. An X should be placed below the object at the point on the lifeline where the object goes out of existence.
4. Layout the messages from the top to the bottom of the diagram based on the order in which they are sent. This is done by drawing arrows to represent the messages being passed from object to object, with an arrow pointing in the message’s transmission direct.
5. Add the execution occurrence to each object’s life line by drawing a narrow rectabgle box over top of the lifelines to represent when the classes are sending and receiving messages.
6. Validate the sequence diagram. This step is used to guarantee that the sequence diagram completely represents the underlying process.

**9) Describe the main building blocks for the communication diagram and how they are represented on the model.**

Communication diagrams essentially provide a view of the dynamic aspects of an object-oriented system. Such as, they can show the members of a set of objects collaborate to implement a use case or a use case scenario. Furthermore, they can be used to model all the interactions among a set of collaborating objects.

**10. How do you show the sequence of messages on a communication diagram?**

The sequence of messages is depicted with a sequence number. The messages are shown as labels on the associations. Included with the labels are lines with arrows showing the direction of the message being sent.

**11) How do you show the direction of a message on a communication diagram?**

A *communication diagram* is an object diagram that shows message-passing relationships instead of associations between objects.

\*We can show the direction of a message by using an arrowhead pointed towards the destination.**Pg. 246-247**

**12. Describe the steps used to create a communication diagram.**

1. Set the context.

2. Identify which objects (actors) and the associations between the objects participate in the collaboration.

3. Layout the communication diagram.

4. Add messages.

5. Validate the communication diagram.

13. **Are states always depicted using rounded rectangles on a behavioral state machine? Explain.**

No. The initial state is shown by a solid filled in small circle while the final state is shown by a small filled in circle surrounded by an empty circle. All other states are depicted by rounded rectangles.

**14) What kinds of events can lead to state transitions on a behavioral state machine?**

Releasing an admitted patient (going from admitted to released) is an example

**15. What are the steps in building a behavioral state machine?**

* 1. Set the context.
  2. Identify the initial, final, and stable states of the object.
  3. Determine the order in which the object will pass through the stable states.
  4. Identify the events, actions, and guard conditions associated with the transitions.
  5. Validate the behavioral state machine.

**16. How are guard conditions shown on a behavioral state machine?**

A *behavioral state machine* is a dynamic model that shows the different states through which a single object passes during its life in response to events, along with its responses and actions.

A *guard condition* is a Boolean expression that includes attribute values, which allows a transition to occur only if the condition is true. This can be shown by including the Boolean expression within “[]” on the transition (object movement from one state to another) in the diagram.

**Pg. 250-251**

**17. Describe the type of class that is best represented by a behavioral state machine. Give two examples of classes that would be good candidates for a behavioral state machine.**

The type of class which is best depicted by a behavioral class diagram is a class which is very dynamic and complex, requiring a good understanding of their states over time and events triggering changes. Two examples of classes which would be good candidates for behavioral state machines are a patient in a hospital and an apartment in an apartment complex.

**18. What is CRUD analysis and what is it used for?**

The crud analysis is an approach to identify potential collaborations amoung classes and verifying and validating a system. It allows the analyst to see what type of interaction that the different types of objects have in the system in a very concise format. It also supports the identification of the more complex objects that could benefit from state modeling using a behavioral state machine.

**19) Identify the models that contain each of the following components:**

1) Actor – Communication Diagram, Sequence Diagram

2) Association – Communication Diagram

3) Class – Sequence Diagram

4) Association – Communication Diagram

5) Final State – Behavioral State Machine

6) Guard condition - Behavioral State Machine, sequence diagram, communication diagram

7) Initial state - Behavioral State Machine

8) Links - All

9) Message – Communication Diagram, Sequence Diagram,

10) Transition - Behavioral State Machine

11) Update Operation - ?

**Chapter 8**

**1) What is the primary difference between an analysis model and a design model?**

The primary difference lies in the purpose of each model.

An *analysis model* represents underlying business problem domain as a set of collaborating objects. In other words, it defines the functional requirements.

A *design model*’s purpose, however, is to increase the likelihood of successfully delivering a system that implements the functional requirements in a manner that is affordable and easily maintainable. In other words, it defines and addresses the nonfunctional requirements. **Pg. 287**

**2. What is a walkthrough? How does it relate to verification and validation?**

A walkthrough is essentially a peer review of a product. In the case of the analysis models, a walkthrough is a review of the different models and diagrams created during analysis. The reviews are typically completed by team of individuals that comes from analysis team, design team, and the client. The purpose of walkthrough is to thoroughly test the fidelity of the analysis models to the functional requirements and to ensure that the models are consistent. Basically, a walkthrough will uncover errors of faults in the evolving specification. It does not correct the errors, just identifies them. It relates to verification and validation by finding the errors and beginning the process for correcting them so that the models will be valid.

**3. What are the different roles played during a walkthrough? What are their purposes?**

a. Presenter role: this should be played by the individual who is primarily responsible for the specific representation being reviewed.

b. Recorder role: they should be a member of the analysis team. This person takes the minutes of the meeting by recording all significant events that occur during the walkthrough.

c. Maintenance oracle: this person raises issues regarding maintenance of the representation.

d. The last role is someone who is responsible for calling, setting up, and running the walkthrough meeting.

**4) What are the relationships within the functional models, structured models, and the behavioral models that need to be tested?**

The activity diagrams, use case descriptions, and use case diagrams must agree with the sequence diagrams, communication diagrams, behavioral state machines, and CRUD matrix.

**5. What is meant by balancing the models?**

It ensures consistency between models. It makes sure that models agree with one another. It also tells us whether the models are trustworthy.

**6) What are the interrelationships between the functional, structural, and the behavioral models that need to be tested?**

**Functional:Structural**

The activity diagrams, use-case descriptions, and use-case diagrams must agree with the CRC cards and class diagrams that represent the evolving model of the problem domain.

**Functional:Behavioral**

The activity diagrams, use-case descriptions, and use-case diagrams must agree with the sequence diagrams, communication diagrams, behavioral state machines, and CRUD matrix.

**Structural:Behavioral**

Objects in a CRUD matrix must be associated with classes that are represented by CRC cards and appear on the class diagram.

Behavioral state machines must be associated with instances (objects) of classes on a class diagram and with a CRC card that represents the class of the instance.

Communication and sequence diagrams contain objects that must be an instantiation of a class that is represented by a CRC card and is located on a class diagram.

Messages contained on the sequence and communication diagrams, transitions on behavioral state machines, and cell entries on a CRUD matrix must be associated with responsibilities and associations on CRC cards and operations in classes and associations connected to the classes on class diagrams.

The states in a behavioral state machine must be associated with different values of an attribute or a set of attributes that describe an object. **Pg. 280-287**

**7. What does factoring mean? How is it related to abstraction and refinement?**

Factoring is the process of separating out a module into a standalone module in and of itself. Abstraction deals with the creation of a higher level idea from a set of ideas [taking low level classes and making a super class. Refine takes a superclass and gets the derived classes from it.

**8. What is a partition? How does a partition relate to collaboration?**A partition is the object oriented equivalent of a subsystem, where a subsystem is decomposition of a larger system into it component systems. (Like accounting information system could be functionally decomposed into an accounts payable system, an account receivable system, a payroll system, and so on)

When looking for partitions, a good place to look is in merged collaborations. Merged collaborations can be decomposed into multiple partitions. Then the rule of thumb is that more messages sent between objects, the more likely the objects belong in the same partition.

9) What is a layer? Name the different layers

A layer represents an element of the software architecture of the evolving system. The layers are: Foundation, Problem domain, Data access and management, Human-Computer interaction, and Physical Architecture.

**10. What is a package? How are packages related to partitions and layers?**

A **package** is a logical grouping of UML elements. It is used to simplify UML diagrams by grouping related elements into a single higher-level element. Partitions and layers are modeled as packages in UML. Furthermore, collaborations are normally factored into a set of partitions, which are typically placed on a layer. In addition, partitions can be composed of other partitions. Also, it’s possible to have classes in partitions, which are contained in another partition, which is placed on a layer. All these groupings are represented using packages in UML.

**11) What is a dependency relationship? How do you identify them?**

A *dependency* relationship is a type of package diagram that represents the fact that a modification dependency exists between two packages. In other words, it tells us that a change in one package could possibly cause a change to be required in another package (the dependent).

They can be identified by *dashed arrows* connecting the related packages. **Pg. 294-295**

**12. What are the five steps for identifying packages and creating package diagrams?**

1. Set the context (what area/part are you looking at)
2. Cluster classes together based on shared relationships
3. Model clustered classes as paskage
4. Identify dependency relations among packages
5. Place dependency relationships between packages

**13. What needs to be verified and validated in package diagrams?**

1. Identified packages must make sense from a problem domain point of view

2. All dependency relationships must be based on message-sending relationships on the communications diagram, associations on the class diagram, and cell entries in the CRUD matrix.

**14) What situations are most appropriate for a custom development design strategy?**

Custom development is considered the best way to create a system and is building a new system from scratch. Teams have complete control over the way the system looks and functions. The most appropriate situations are those for small and simple projects or the beginning of a small business.

**15. What are some problems using a packaged software approach to building a new system? How can these problems be addressed?**

One problem is that companies buying packaged systems must accept the functionality that is provided by the system, and rarely is there a perfect fit. If the packaged system is large in scope, its implementation could mean a substantial change in the way the company does business. This is a dangerous way to go. Another problem would be the difficulty of installing some packaged software’s. One way these problems can be addressed through customization, which allows for manipulation of system parameters to change the way certain features work. Another way to address a problem is through a workaround. A workaround is a custom-built add-on program that interfaces with the packaged application to handle special needs. Another way to address a problem is through systems integration. This refers to the process of building a new system by combining packaged software, existing legacy systems, and new software written to integrate these together.

**16) Why do companies invest in ERP systems?**

An *ERP*  (Enterprise Resource Planning) *System* is an all-encompassing application that is installed to automate an entire business. It is a type of packaged software that is bought from a third party vendor.

Companies invest in these systems because most of their needs can be met and they do not have to have their own IT department spending countless hours developing the various systems they need. It is likely that purchasing this software will be more efficient to install and can be installed in less time than if the software had to be developed from scratch. The ERP’s have gone through testing, and have been proved to operate as intended. The vendor’s also provide support to their customers. In essence, the hard work is done, the company has money to spend, so why not just buy already proven software that will work? **Pg. 300-301**

**17. What are the pros and cons of using a workaround?**

Pros

* Give you the ability to handle special need
* Nice way to create needed functionality that does not exist in the software package

Cons

* Workarounds aren’t supported by the vendor who supplies the package software, so when upgrades are made to the main system, they may make the workaround ineffective
* Also if problems arise, vendors have a tendency to blame the workaround as the culprit and refuse to provide support

**18. When is outsourcing considered a good design strategy? When is it not appropriate?**

There can be great benefit to having someone else develop your system. They may be more experienced in that technology or have more resources. So this outsourcing can be a good for a new system. The problem being solved is well defined.

When the cost of outsources out weight the benefits of keeping it in house. When the problem being solved is not understood.

**19) What are the differences between the time and arrangements, fixed-price, and value added contracts for outsourcing?**

**Time and Arrangements** – is very flexible because a company agrees to pay for what ever time and expenses are needed to get the job done. Of course this agreement could result in a large bill that exceeds the initial estimates.

**Fixed price** – A company will pay no more than expected with this contract. If the outsourcer exceeds the agreed upon price, it will have to absorb the costs.

**Value added** – The outsourcer reaps some percentage of the completed system’s benefits.

**20. How are the alternative matrix and feasibility analysis related?**

The alternative matrix and feasibility analysis are related because they are both created using the exact same steps. The only difference is that the alternative matrix combines several feasibility analyses into one matrix so that the alternatives can be easily compared.

**21) What is an RFP? How is this different from an RFI?**

An *RFP* (Request for Proposal) is a document written by a company that solicits proposals to provide alternative solutions from various vendors for a software need the company has.

An *RFI* (Request for Information) just list the company’s needs, and it requires general information from respondents that communicates the services they can provide.

An ***RFP*** is different in that the company explains the system they are trying to build and the criteria it will us to select a system. The exact needs (description of system, special technical needs, and evaluation criteria) are laid out in detail. It is usually much longer and vendors typically lay out detailed time, costs, and descriptions of how their solution will address the needs of the company.