Generated by Foxit PDF Creator © Foxit Software
http://www.foxitsoftware.com For evaluation only.

1. What is object technology? What do you perceive as object technology's strength? It's weakness?

Solution:

(一) Object technology:

A set of principles (abstraction, encapsulation, polymorphism) guiding software construction, together with languages, databases, and other tools that support those principles.

(二) Strength:

- Reflects a single paradigm
- Facilitates architectural and code reuse
- Reflects real world models more closely
- Encourages stability
- Is adaptive to change

(三) Weakness:

The object technology emphasized the cooperation actually displays the order of complexity which another violent characteristic - coupling degree is too high .

2. What is UML? List at least three benefits of developing with UML.

(一) UML:

The UML is a language for Visualizing、Specifying、Constructing、Documenting the artifacts of a software-intensive system.

(二) benefits of developing with UML：

- There are things about a software system you can't understand unless you build UML models.
- An explicit model facilitates communication.
- The UML builds models that are precise, unambiguous, and complete.
- UML models can be directly connected to a variety of programming languages such as Java, C++, Visual Basic.
- The UML addresses documentation of system architecture,

requirements, tests, project planning, and release management.

3. What process characteristic best fit the UML? Describe each characteristic.

The UML is largely process independent. A process fully benefits from the UML when the process is: Use-case driven、Architecture-centric、Iterative and incremental.

(一) Use-case driven ：

- Use cases defined for a system are the basis for the entire development process
- Benefits of use cases:

Concise, simple, and understandable by a wide range of stakeholders. Help synchronize the content of different models.

(二) Architecture-centric ：

Generated by Foxit PDF Creator © Foxit Software
http://www.foxitsoftware.com For evaluation only.

- A system's architecture is used as a primary artifact for conceptualizing, constructing, managing, and evolving the system under development.
- Benefits:

- Intellectual control over a project to manage its complexity and to maintain system integrity.
- Effective basis for large-scale reuse.
- A basis for project management.
- Assistance in component-based development.

(三) Iterative and incremental:

- Critical risks are resolved before making large investments.
- Initial iterations enable early user feedback.
- Testing and integration are continuous.
- Objective milestones focus on the short term.
- Progress is measured by assessing implementations.

Partial implementations can be deployed.

4. What is a use-case driven process? What is use-case? What are the benefits of use case?

(一) Use-Case Driven Process:
Use cases defined for a system are the basis for the entire development process.
(二) use-case:
Defines a set of use-case instances, where each instance is a sequence of actions a system performs that yields an observable
result of value to a particular actor.
(四) ƒ A use case models a dialogue between one or more actors and the system
(五) ƒ A use case describes the actions the system takes to deliver something of value to the actor
(三) Benefits of use cases:
(六) ƒConcise, simple, and understandable by a wide range of stakeholders.
(七) Help synchronize the content of different models.

5. What is system's architecture? What is an architecture-centric Process?

A system's architecture is used as a primary artifact for conceptualizing, constructing, managing, and evolving the system under development.

6. What is iteration? What are the benefits of Iterative Development?

(一) Iteration:
a single execution of a set of instructions that are to be repeated until a specified result is obtained
(二) benefits of Iterative Development:

- Critical risks are resolved before making large investments.
- Initial iterations enable early user feedback.

Generated by Foxit PDF Creator © Foxit Software
http://www.foxitsoftware.com For evaluation only.

- Testing and integration are continuous.
- Objective milestones focus on the short term.
- Progress is measured by assessing implementations.
- Partial implementations can be deployed.
- Each iteration produces an executable release, an additional increment of the system.
- Each iteration includes integration and test.

7. What are the basic principles of OO technology? Describe each in detail.

(一) OO technology:
Abstraction Encapsulation Modularity Hierarchy
(二) Detail:

- Abstraction:
  - The essential characteristics of an entity that distinguishes it from all other kinds of entities.
  - Defines a boundary relative to the perspective of the viewer.
  - Is not a concrete manifestation, denotes the ideal essence of something.
- Encapsulation:
  - Hides implementation from clients.
  - Clients depend on interface.
- Modularity:
  - Breaks up something complex into manageable pieces.
  - Helps people understand complex systems.
- Hierarchy

Inherits is refers to an object to use another object directly the attribute and the method.

8. What is use case model? Which artifacts can be included in a use case model?

(一) use case model:

- Š A model that describes a system's functional requirements in terms of use cases.
- Š A model of the system's intended functions (use cases) and its environment (actors).

(二) Artifacts:

- An actor represents anything that interacts with the system.

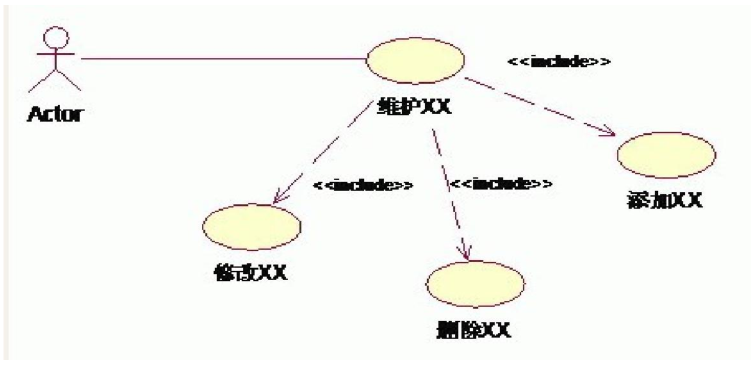
A use case describes a sequence of events, performed by the system, that yields an observable result of value to a particular actor.

9. List three types of relationships existed between different use cases and give examples.

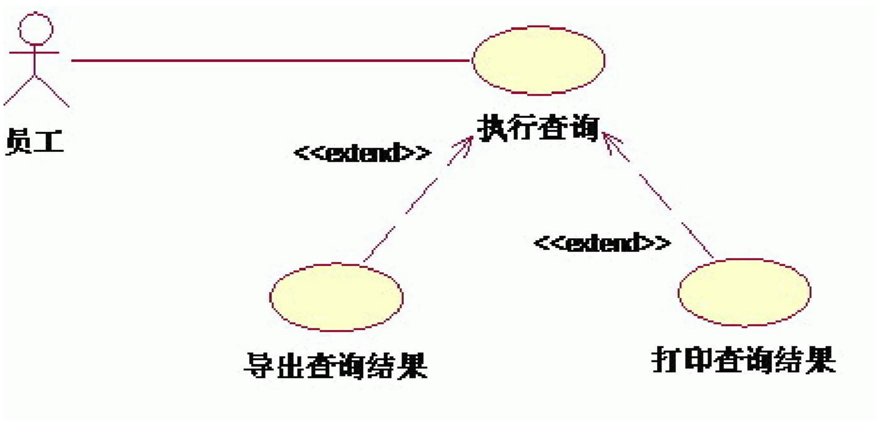
three types of relationships ：
Include、extend、generalization

Generated by Foxit PDF Creator © Foxit Software
http://www.foxitsoftware.com For evaluation only.

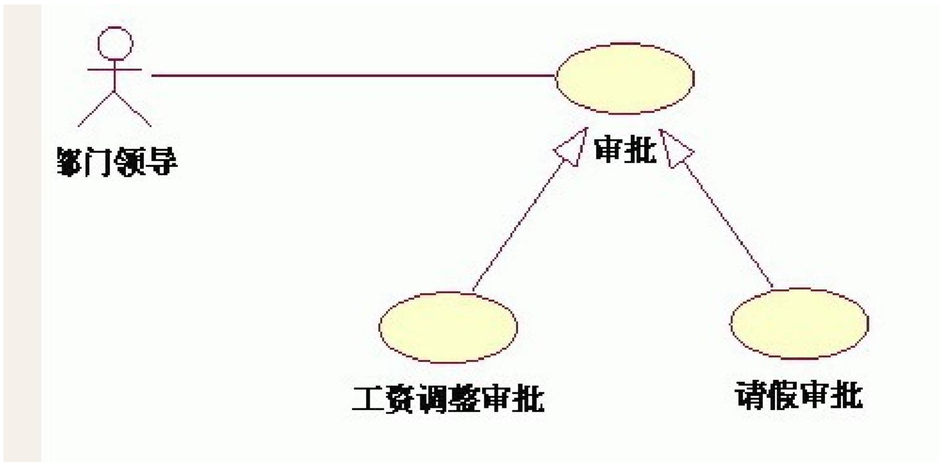
Include:



Extend:



Generalization:



10. Explain the following diagram and their elements with examples.
1) Use case diagram 2) Activity diagram 3) Sequence diagram 4) Collaboration diagram
5) Class diagram 6) state chart diagram 7) Deployment diagram

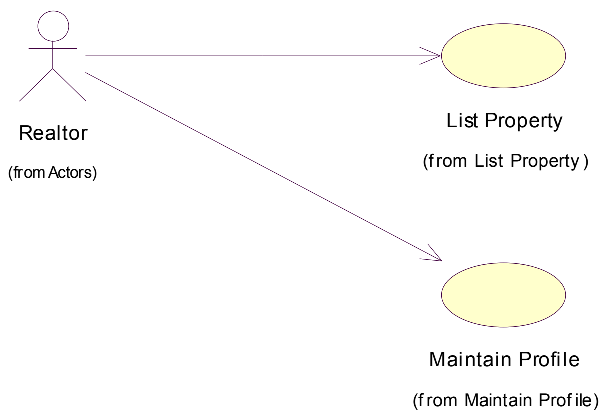
1)Use case diagram 2) Activity diagram 3) Sequence diagram 4) Collaboration diagram
5) Class diagram 6) state chart diagram 7) Deployment diagram

1) Use case diagram:

- ƒ A use case diagram models a dialogue between one or more actors and the

Generated by Foxit PDF Creator © Foxit Software
http://www.foxitsoftware.com For evaluation only.

system.

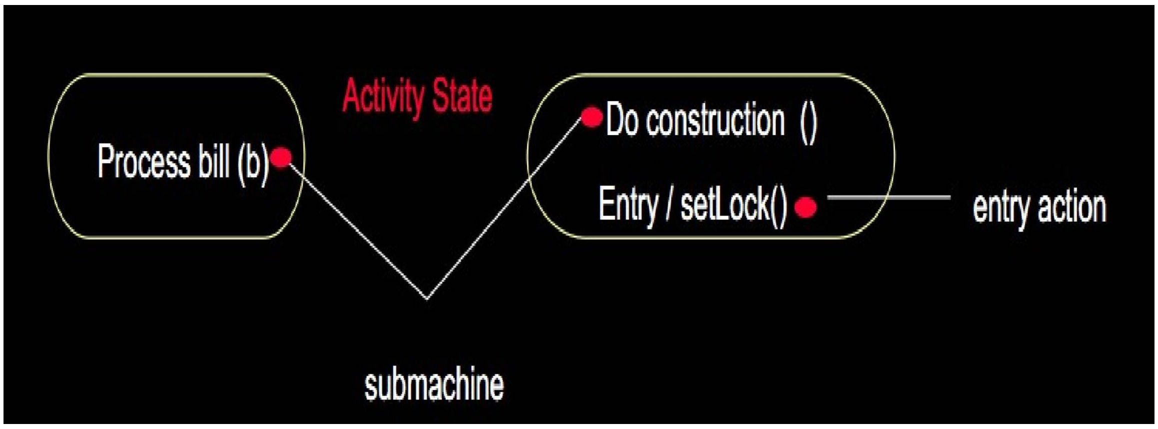


Its elements is

- actor: Realtor
- usecase: List Property、Maintain Profile

2) Activity diagram:

A Activity diagram is essentially a flow chart, showing flow of control from activity to activity.



Its elements is Activity state 、entry and exit actions、submachine specifications

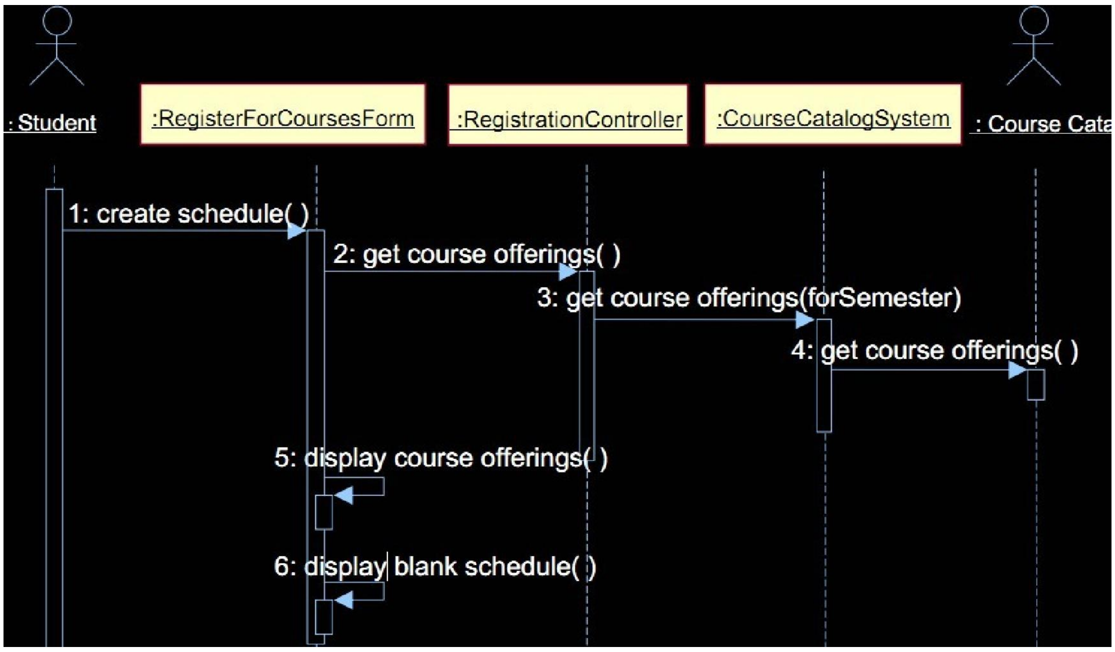
3) Sequence diagram

A sequence diagram is an interaction diagram that emphasizes the time ordering of messages.

The diagram shows:

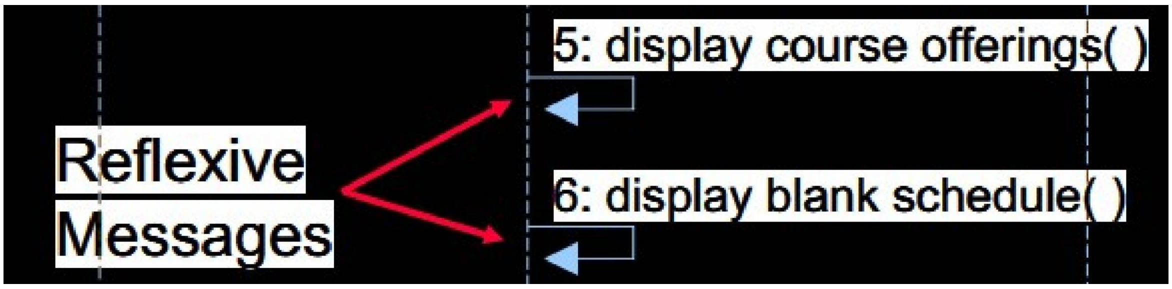
- ƒ The objects participating in the interaction.
- ƒ The sequence of messages exchanged.

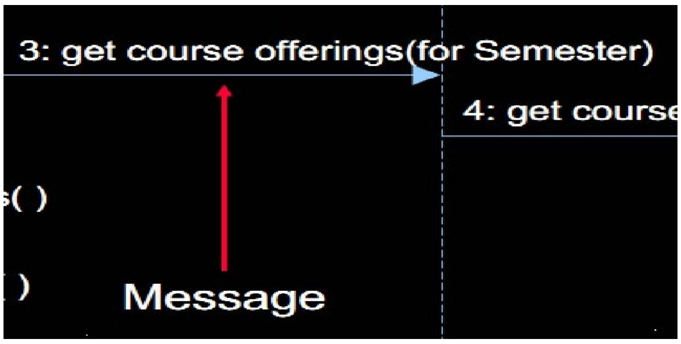
Generated by Foxit PDF Creator © Foxit Software
http://www.foxitsoftware.com For evaluation only.


: Student
:RegisterForCoursesForm
:RegistrationController
:CourseCatalogSystem
: Course Cata
1: create schedule( )
2: get course offerings( )
3: get course offerings(forSemester)
4: get course offerings( )
5: display course offerings( )
6: display blank schedule( )

Its elements:

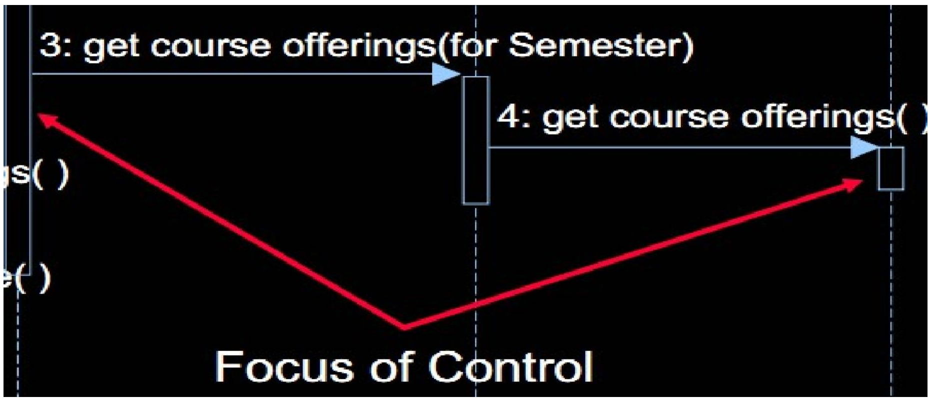
- Anonymous Objects
- :RegisterForCoursesForm
- :RegistrationController
- Named Object
- SWTSU Catalog:
CourseCatalogSystem
- Actor instances
- : Student
- : Course Catalog
- Messages





Generated by Foxit PDF Creator © Foxit Software
http://www.foxitsoftware.com For evaluation only.

- Focus of Control

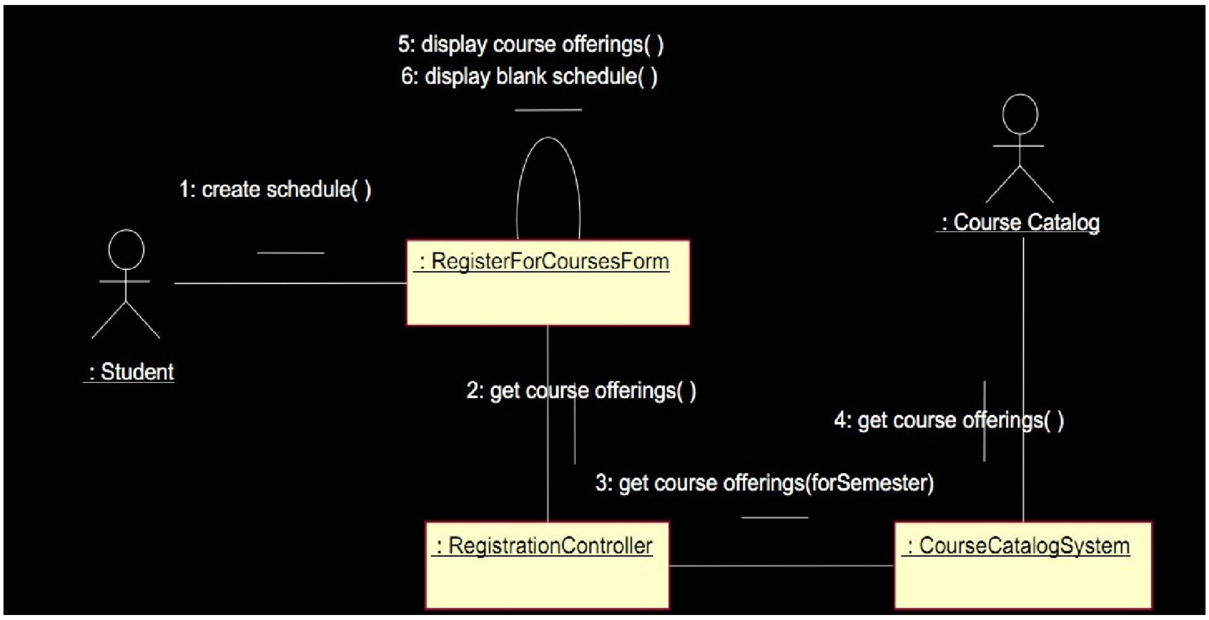

3: get course offerings(for Semester)
4: get course offerings( )
s( )
e( )
Focus of Control

4) Collaboration diagram

A collaboration diagram emphasizes the organization of the objects that participate in an interaction.

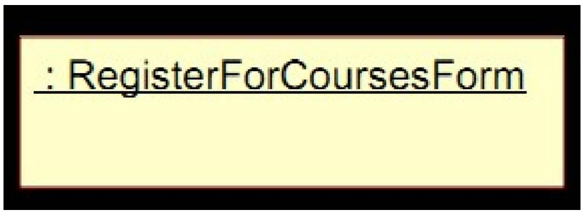
The collaboration diagram shows:

- The objects participating in the interaction.
- Links between the objects.
- Messages passed between the objects.

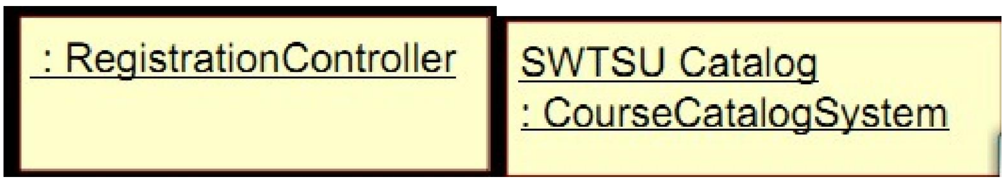

5: display course offerings( )
6: display blank schedule( )
1: create schedule( )
: Course Catalog
: RegisterForCoursesForm
: Student
2: get course offerings( )
4: get course offerings( )
3: get course offerings(forSemester)
: RegistrationController
: CourseCatalogSystem

Its element:

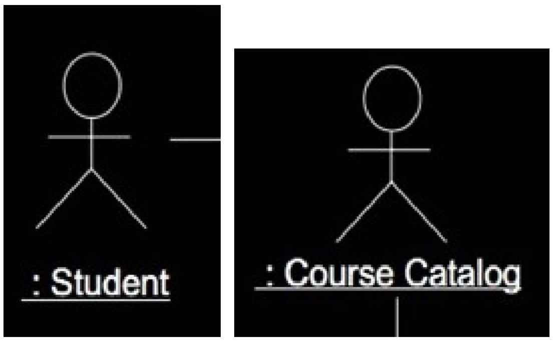
- Objects:



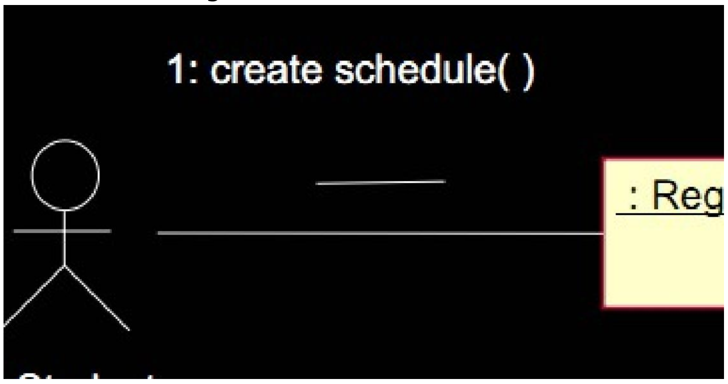
Generated by Foxit PDF Creator © Foxit Software
http://www.foxitsoftware.com For evaluation only.



- Actors:



- Links and Messages:

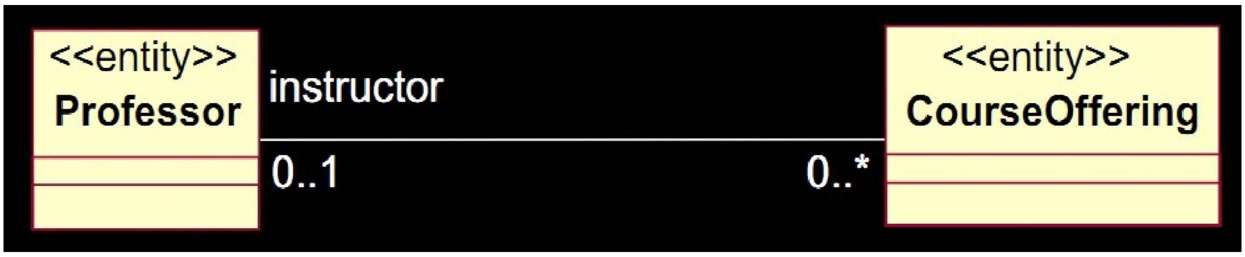


5) Class diagram

- Association:

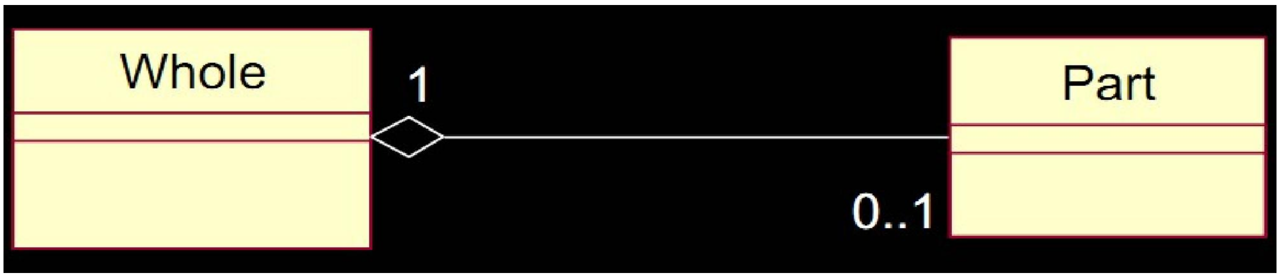
A structural relationship specifying that objects of one thing are connected to objects of another thing.

Multiplicity is the number of instances one class relates to ONE instance of another class.

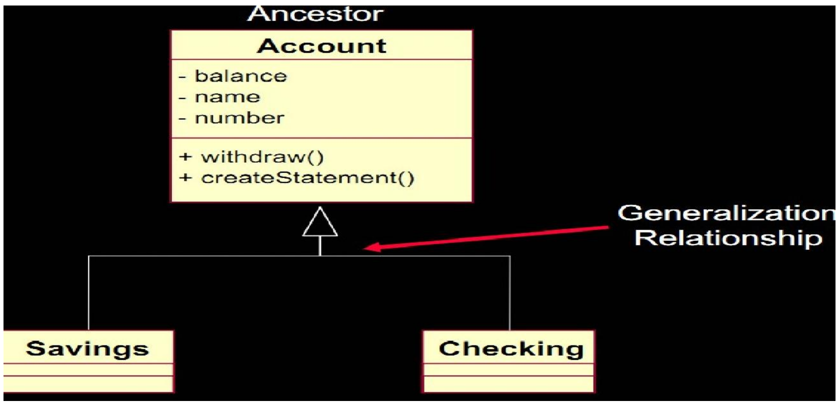


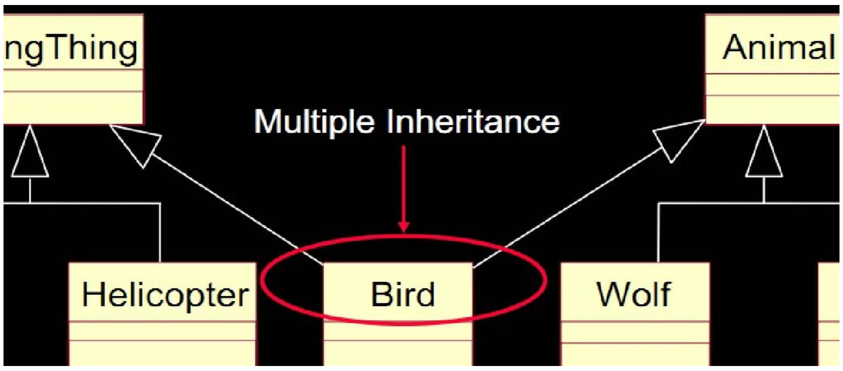
- Aggregation

Generated by Foxit PDF Creator © Foxit Software
http://www.foxitsoftware.com For evaluation only.



- Generalization

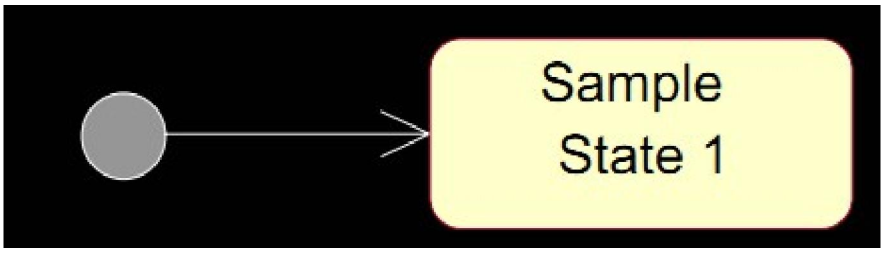




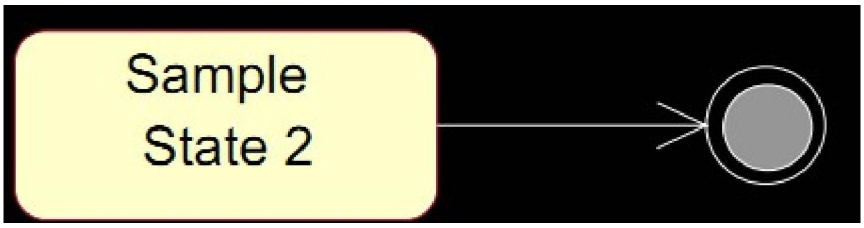
6) state chart diagram

The state of an object is one of the possible conditions in which an object may exist.

- The initial state:

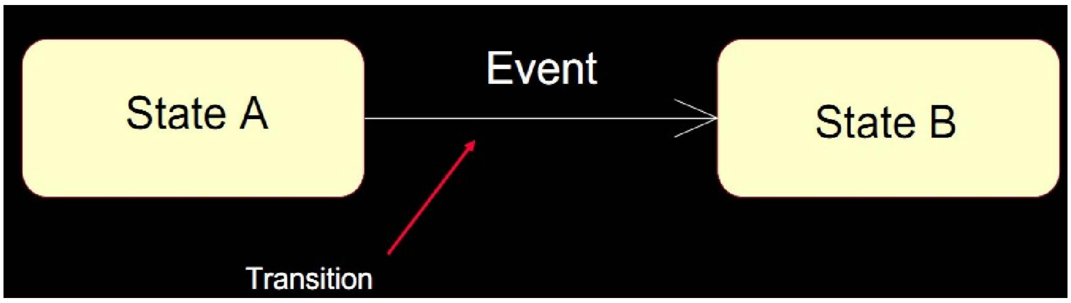


- A final state:



- Transition:

Generated by Foxit PDF Creator © Foxit Software
http://www.foxitsoftware.com For evaluation only.



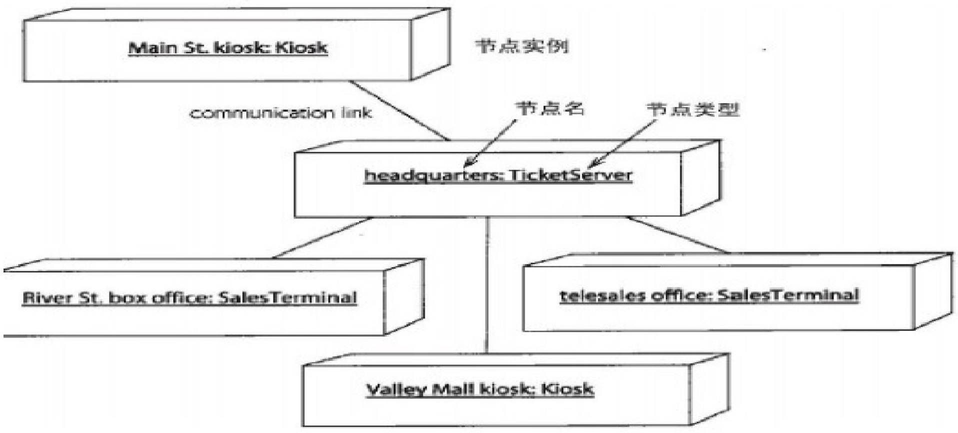
7) Deployment diagram

The deployment diagram shows:

ƒ Configuration of processing nodes at run-time

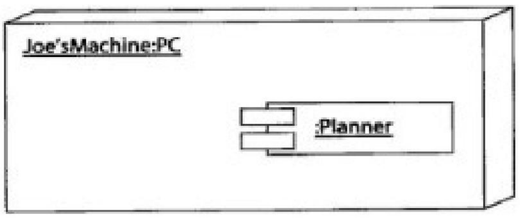
ƒ Communication links between these nodes

ƒ Component instances and objects that reside on them

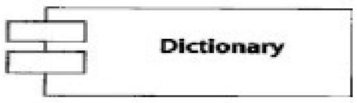

Main St. kiosk: Kiosk
节点实例
communication link
节点名
节点类型
headquarters: TicketServer
River St. box office: SalesTerminal
telesales office: SalesTerminal
Valley Mall kiosk: Kiosk

Its elements:

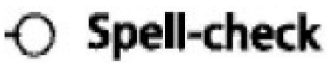
- Node



- Component



- interface



- Component intance

Generated by Foxit PDF Creator © Foxit Software
http://www.foxitsoftware.com For evaluation only.

11. Describe the similarities and differences between the sequence diagram and collaboration diagram.

Sequence diagram is a mode that describe interactions between the objects, It use an object "lifeline" and message passing between them to show how the objects take part in interaction.
Collaboration diagram is a kind of interaction Diagram, emphasize the receiving and sending messages between the objects of organizational structure.
They are all used to describe interactions between the objects. But Sequence Diagram emphasize the time, while collaboration diagram emphasize the space.

12. Define the different relationships in class diagram: dependency, association, aggregation, composition, generalization.

Dependency is a relationship among class where an object construct an instance of another object or rely on the service of another object.
Association is a structural relationship specifying that objects of one thing are connected to objects of another thing.
Aggregation is an "is a part-of" relationship, it is a special form of association that models a whole-part relationship between the aggregate (the whole) and its parts.
Composition is a relationship stronger than aggregation, a form of association.
Generalization is an "is a kind of" relationship, it is a relationship among classes where one class shares the structure and/or behavior of one or more classes.

13. What is a node in deployment diagram? List two different types of nodes.

The deployment diagram shows:configuration of processing nodes at run-time, communication links between these nodes, component instances and objects that reside on them
The node in deployment diagram is a physical element that exists at run-time and represents a computational resource.
Types:
Processor Node
Device Node

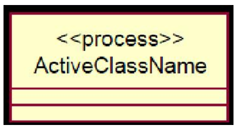
14. Describe the extensibility mechanisms of UML.

Extensibility Mechanisms include Stereotypes, Tagged values, Constraints.

Generated by Foxit PDF Creator © Foxit Software
http://www.foxitsoftware.com For evaluation only.

15. What is the function of Stereotypes? Give two examples of stereotypes.

A particular abstraction is marked as a "stereotype" and this stereotype is then used at other places in the model to denote the associated abstraction, and Stereotypes help create new building blocks.
Examples : <<process>> or <<thread>>



16. Explain the six best practices of software engineering.

Practice 1: Develop Iteratively

It contains Requirements analysis, Design, Code and Unit test, Subsystem integration, System test, And iterative development produces an executable.

Practice 2: Manage Requirements

The aspects of Requirements Management are these: Analyze the Problem, Understand User Needs, Define the system, Manage Scope, Refine the System Definition, Manage Changing Requirements.

Practice 3: Use Component Architectures

The points of Resilient:

Meets current and future requirements
Improves extensibility
Enables reuse
Encapsulates system dependencies

The point of Component-based:

Reuse or customize components
Select from commercially available components
Evolve existing software incrementally

17. What is RUP? How many phases are there in RUP? Describe each phase's purpose and milestone.

RUP is short for Rational Unified Process. It implements best practices.
There are four phases in RUP:

Inception - Define the scope of project. After it is the **Lifecycle Objective Milestone (LCO)**

Elaboration - Plan project, specify features and baseline architecture. After it is the **Lifecycle Architecture Milestone (LCA)**

Construction - Build the product. After it is the **Initial Operational Capability Milestone (IOC)**

Generated by Foxit PDF Creator © Foxit Software
http://www.foxitsoftware.com For evaluation only.

Transition - Transition the product into end-user community. After it is the **Product Release**

18. Name and briefly describe the "4+1"views of architecture.

Use-Case View: It is the view for End-user, shows Functionality;
Logical View: It is the view for Designers, shows Structure;
Process View: It is the view for System integrators, shows Performance, Scalability,Throughput;
Implementation View: It is the view for Programmers, shows Software management;
Deployment View: It is the view for System engineering, shows System topology, Delivery, installation and communication.

19. What is the difference between analysis and design?

Analysis include these points:
- Focus on understanding the problem
- Idealized design
- Behavior
- System structure
- Functional requirements
- A small model

While Design include the following points:
Focus on understanding the solution
Operations and attributes
Performance
Close to real code
Object lifecycles
Nonfunctional requirements
A large model

20. Please describe the whole process of OO analysis and design with UML.

In the workflow.
Analysis define a candidate architecture, perform architectural synthesis, and analyze behavior.
Design refine the architecture, define components, and design the database.
That's the whole process.

21. What is a layered architecture? Give examples of typical layers.

Generated by Foxit PDF Creator © Foxit Software
http://www.foxitsoftware.com For evaluation only.

layered architecture is a way of constructuring a system in which each component is implemented as a layer, which draws upon services made available by a lower layer. Such as the OSI layered architecture, which consist of Physical Layer . Data Link Layer . Network Layer . Transport Layer . Session Layer . Presentation Layer . Application Layer .

22. What are analysis mechanisms? What are design mechanisms? Give examples.

An analysis mechanisms is a conceptual representation of an architectural mechanism. Over time, analysis mechanisms are refined into design mechanisms.
A design mechanism is a concrete representation of an architectural mechanism. It is refined from an analysis mechanism.
For example, we may have to design a device to store some data, in the analysis mechanisms we have analysis the function of the device is to store some data, and then we have to design this thing in more detail based on the size of the data and so on.

23. What is an analysis class? Name and describe the three analysis stereotypes. Give examples.

Analysis classes represent an early conceptual model for 'things in the system which have responsibilities and behavior'. They eventually evolve into classes and subsystems in the Design Model.
The three analysis stereotypes of the analysis class are boundary classes control classes and entity classes.
A boundary class is a class used to model interaction between the system's surroundings and its inner workings. Such interaction involves transforming and translating events and noting changes in the system presentation (such as the interface). Common boundary classes include windows, communication protocols, printer interfaces, sensors, and terminals.
A control class is a class used to model control behavior specific to one or a few use cases. Control objects (instances of control classes) often control other objects, so their behavior is of the coordinating type. Control classes encapsulate use-case specific behavior.
An entity class is a class used to model information and associated behavior that must be stored. Entity objects (instances of entity classes) are used to hold and update information about some phenomenon, such as an event, a person, or some real-life object. They are usually persistent, having attributes and relationships needed for a long period, sometimes for the life of the system. Typical examples of entity classes in a banking system are Account and Customer. In a network-handling system, examples are Node and Link.

24. What is Use-case realization? What‘s your understandings about the benefit of the use-case realization structure.

Generated by Foxit PDF Creator © Foxit Software
http://www.foxitsoftware.com For evaluation only.

A use-case realization describes how a particular use case is realized within the design model, in terms of collaborating objects.
The use-case realization structure make it easier for us to organize the model element needed to realize the use cases in the design model.

25. Describe the steps occurred in the use-case analysis.

For each use case in an iteration perform the following steps:
Create the Use-Case Realization
Supplement the Use-Case Description
Find Analysis Classes from Use-Case Behavior
Distribute Behavior to Analysis Classes
Describe Responsibilities
Describe Attributes and Associations
After getting the result class perform the next four steps:
Reconcile the Use-Case Realizations
Qualify Analysis Mechanisms
Establish Traceability
Review the Results

26. What's the package, and why we need package?

A package is something used to group elements, and to provide a namespace for the grouped elements.
package make all the classes, objects, use cases, components, nodes and node instances organized in it, thus making it possible for us to manage all the stuffs of the project.

27. What is a subsystem? What is an interface? How does a subsystem differ from a package?

A subsystem is a coherent and somewhat independent component of a larger system
A interface is a common means for unrelated objects to communicate with each other. These are definitions of methods and values which the objects agree upon in order to cooperate.
The difference is that the subsystem should fulfill the request by itself rather, while for a package, it just need to provide the elements inside it, and the client will finish the task.

28. What is the purpose of describing the run-time architecture? How to model the process view?

The purpose is to analyze concurrency requirements, to identify processes, identify inter-process communication mechanisms, allocate inter-process

Generated by Foxit PDF Creator © Foxit Software
http://www.foxitsoftware.com For evaluation only.

coordination resources, identify process lifecycles, and distribute model elements among processes.
we can model the process view by the following steps:
Analyze Concurrency Requirements
Identify Processes and Threads
Identify Process Lifecycles
Identify Inter-Process Communication Mechanisms
Allocate Inter-Process Coordination Resources
Map Processes onto the Implementation Environment
Map Design Elements To Threads of Control Analyze

29. What is the purpose of describing the distribution? How to model the deployment view?

The purpose is to describe how the functionality of the system is distributed across physical nodes. This activity applies only to distributed systems.
we can model the deployment view by the following steps:
Analyze Distribution Requirements
Define the Network Configuration
Allocate System Elements to Nodes

30. Describe the 3 typical distribution patterns, C/S , B/Sand P2P.

C/S is short for Client/Sever, which describes the relationship of cooperating programs in an application. The server component provides a function or service to one or many clients, which initiate requests for such services.
B/S is short for Browser/Server. which also describes the relationship of cooperating programs in an application. However the client does not need to install the program. The user can access the function or service and interact with the Web server and database through browser.
P2P is short for peer to peer, it relies on the computing and networking of the participator rather than that of few servers.

31. What is the difference between the process models of agile and RUP?

The RUP is based on documentation, while agile focus on working code. In Rup, developer communicate with user via documentation, while Agile focus on direct communication. After all, RUP is really a project-management centered process. But a project based on Agile can be very difficult to manage.