CHAPTER THREE

Unified Modeling Language (UML)

UML VIEWS

❑ There is no sharp line between the various concepts and constructs in UML, but for convenience, we divide them into several views.

❑ A view is simply a subset of UML modeling constructs that represents one aspect of a system.

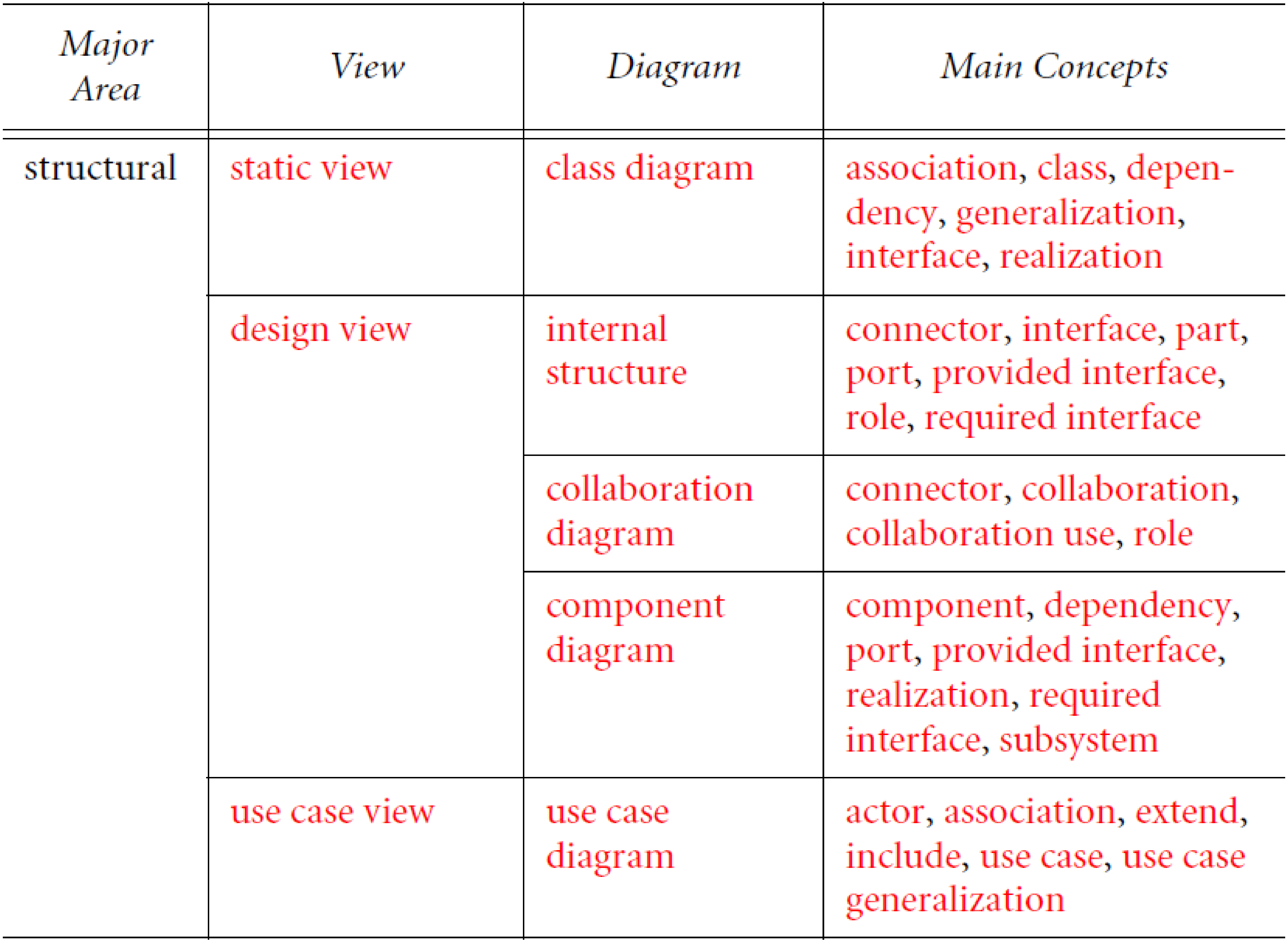
❑ One or two kinds of diagrams provide a visual notation for the concepts in each views.

❑ The views used are not part of the UML specification.

❑ At the top level views can be divided into these areas:

- ✓ Structural classification
- ✓ Dynamic behavior ,
- ✓ Physical layout and
- ✓ Model management.

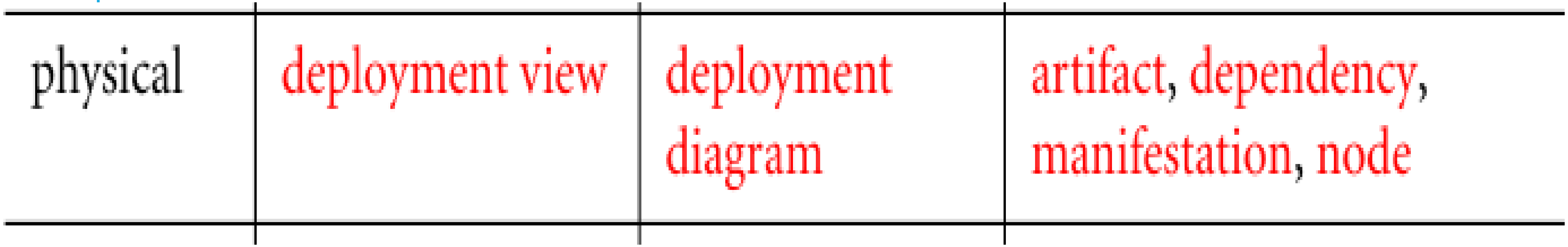
- Structural classification describes the things in the system and their relationships to other things.
- The classifier concept models things in a system.
- Classifiers include class, use case, actor, node, collaboration, and component.
- Classifiers provide the basis on top of which dynamic behavior is built.



Major Area	*View*	*Diagram*	*Main Concepts*
structural	static view	class diagram	association, class, dependency, generalization, interface, realization
	design view	internal structure	connector, interface, part, port, provided interface, role, required interface
		collaboration diagram	connector, collaboration, collaboration use, role
		component diagram	component, dependency, port, provided interface, realization, required interface, subsystem
	use case view	use case diagram	actor, association, extend, include, use case, use case generalization

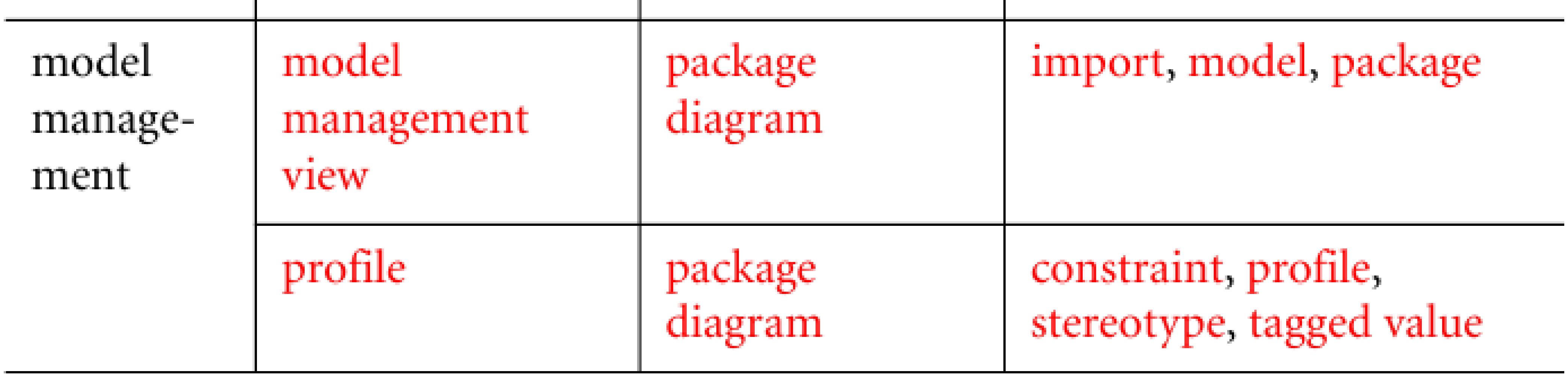
- Dynamic behavior describes the behavior of a system or other classifier over time.
- Behavior can be described as a series of changes to snapshots of the system drawn from the static view.

Major Area	*View*	*Diagram*	*Main Concepts*
dynamic	state machine view	state machine diagram	completion transition, do activity, effect, event, region, state, transition, trigger
	activity view	activity diagram	action, activity, control flow, control node, data flow, exception, expansion region, fork, join, object node, pin
	interaction view	sequence diagram	occurrence specification, execution specification, interaction, interaction fragment, interaction operand, lifeline, message, signal
		communication diagram	collaboration, guard condition, message, role, sequence number



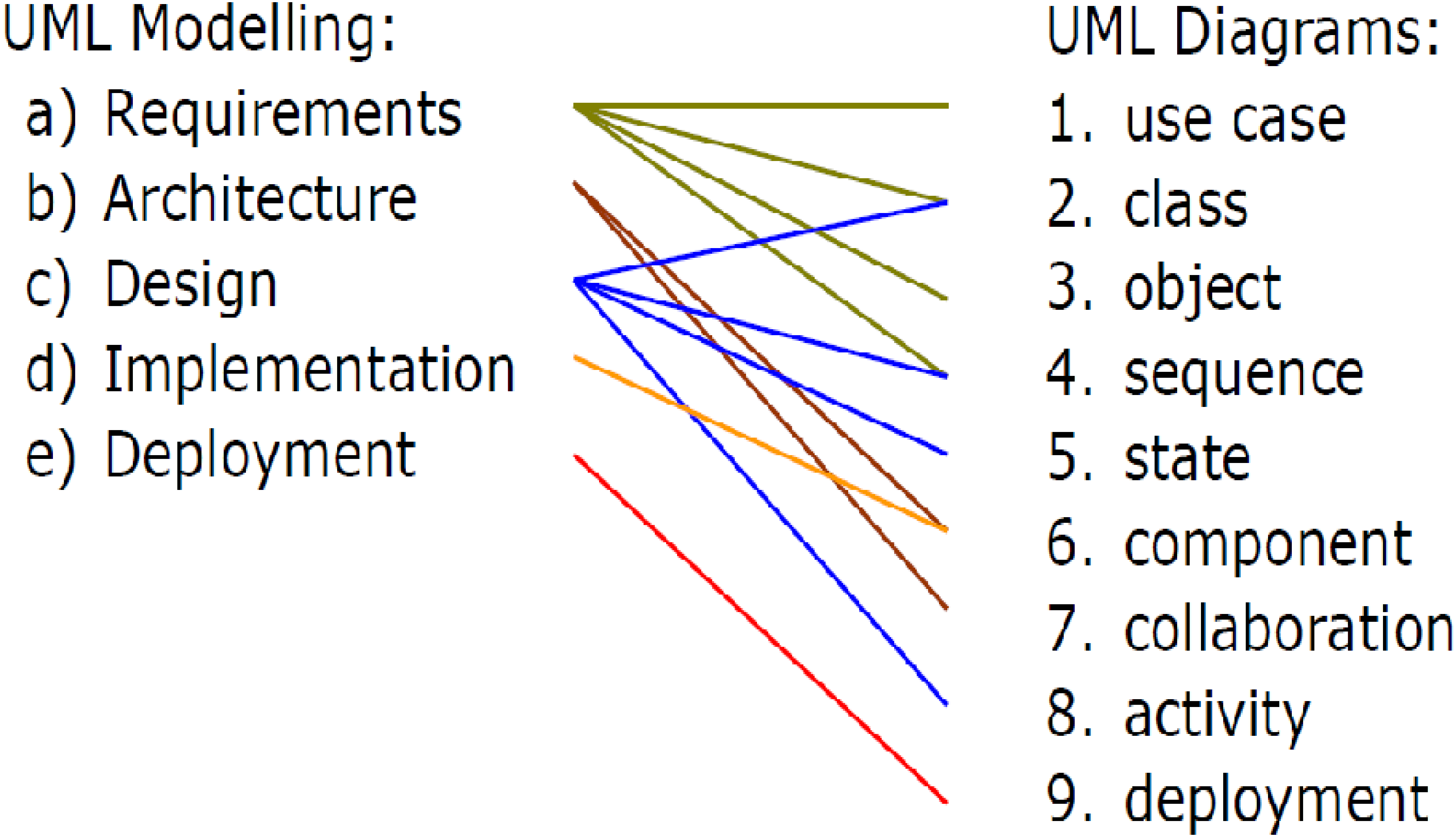
physical	deployment view	deployment diagram	artifact, dependency, manifestation, node

- Physical layout describes the computational resources in the system and the deployment of artifacts on them.



model management	model management view	package diagram	import, model, package
	profile	package diagram	constraint, profile, stereotype, tagged value

- Model management describes the organization of the models themselves into hierarchical units.
- The package is the generic organizational unit for models.
- A model is a package hierarchy that provides a semantically complete abstraction of a system from a particular viewpoint.
- The model management view crosses the other views and organizes them for development work and configuration control.



UML Modelling:

a) Requirements

b) Architecture

c) Design

d) Implementation

e) Deployment

UML Diagrams:

1. use case
2. class
3. object
4. sequence
5. state
6. component
7. collaboration
8. activity
9. deployment

STATIC VIEW

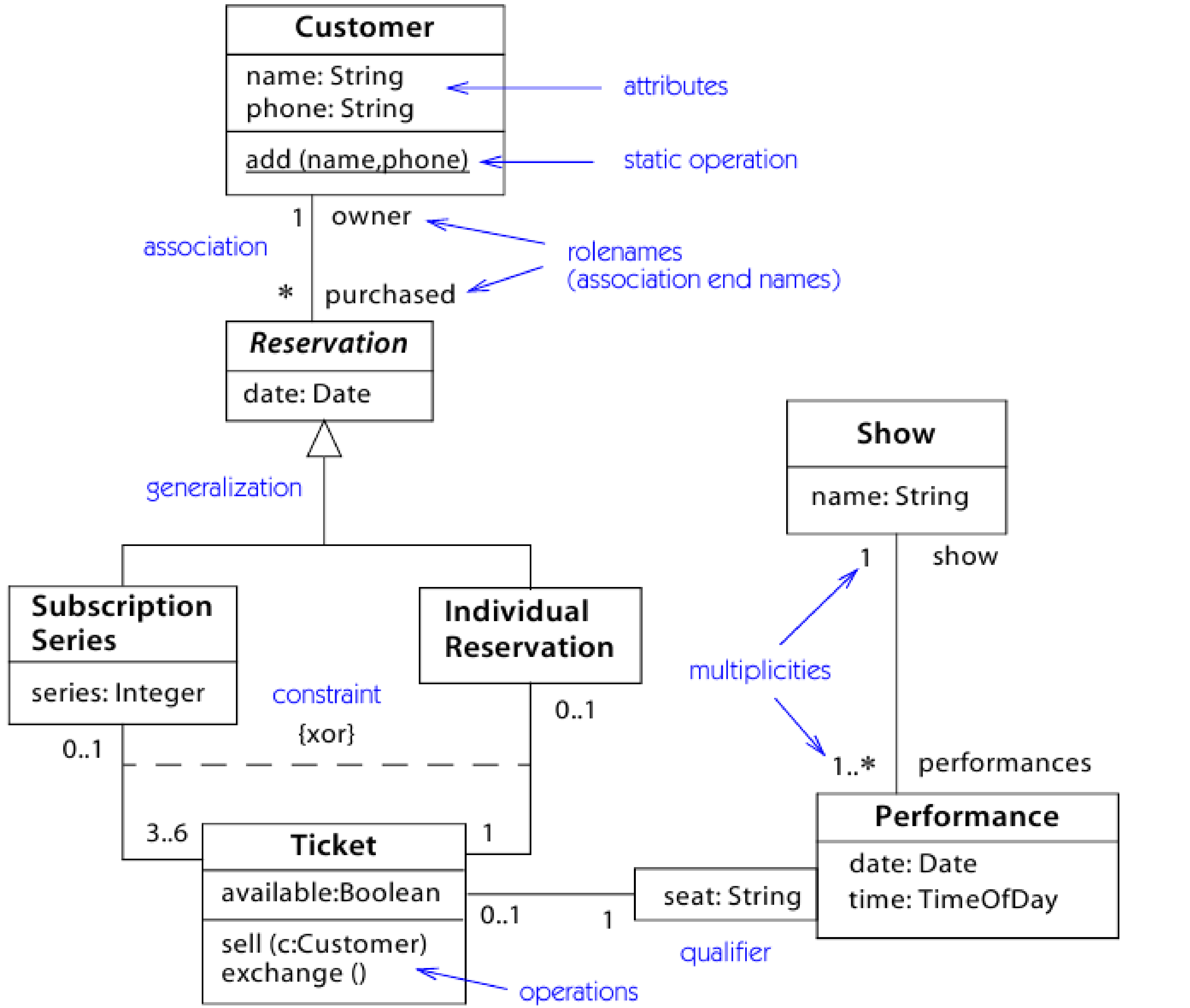
❑ The static view models concepts in the application domain as well as internal concepts invented as part of the implementation of an application.

❑ This view is static because it does not describe the time-dependent behavior of the system, which is described in other views.

❑ The static view is displayed in class diagrams, so called because their main focus is the description of classes.

❑ The main constituents of the static view are classes and their relationships: association, generalization, and various kinds of dependency, such as realization and usage.

❑ A class is the description of a concept from the application domain or the application solution.


Customer
name: String
phone: String
add (name,phone)
attributes
static operation
1
owner
association
rolenames
(association end names)
*
purchased
Reservation
date: Date
generalization
Show
name: String
1
show
Subscription Series
series: Integer
Individual Reservation
multiplicities
constraint
{xor}
0..1
0..1
1..*
performances
Performance
date: Date
time: TimeOfDay
3..6
Ticket
1
available:Boolean
seat: String
0..1
1
sell (c:Customer)
exchange ()
qualifier
operations

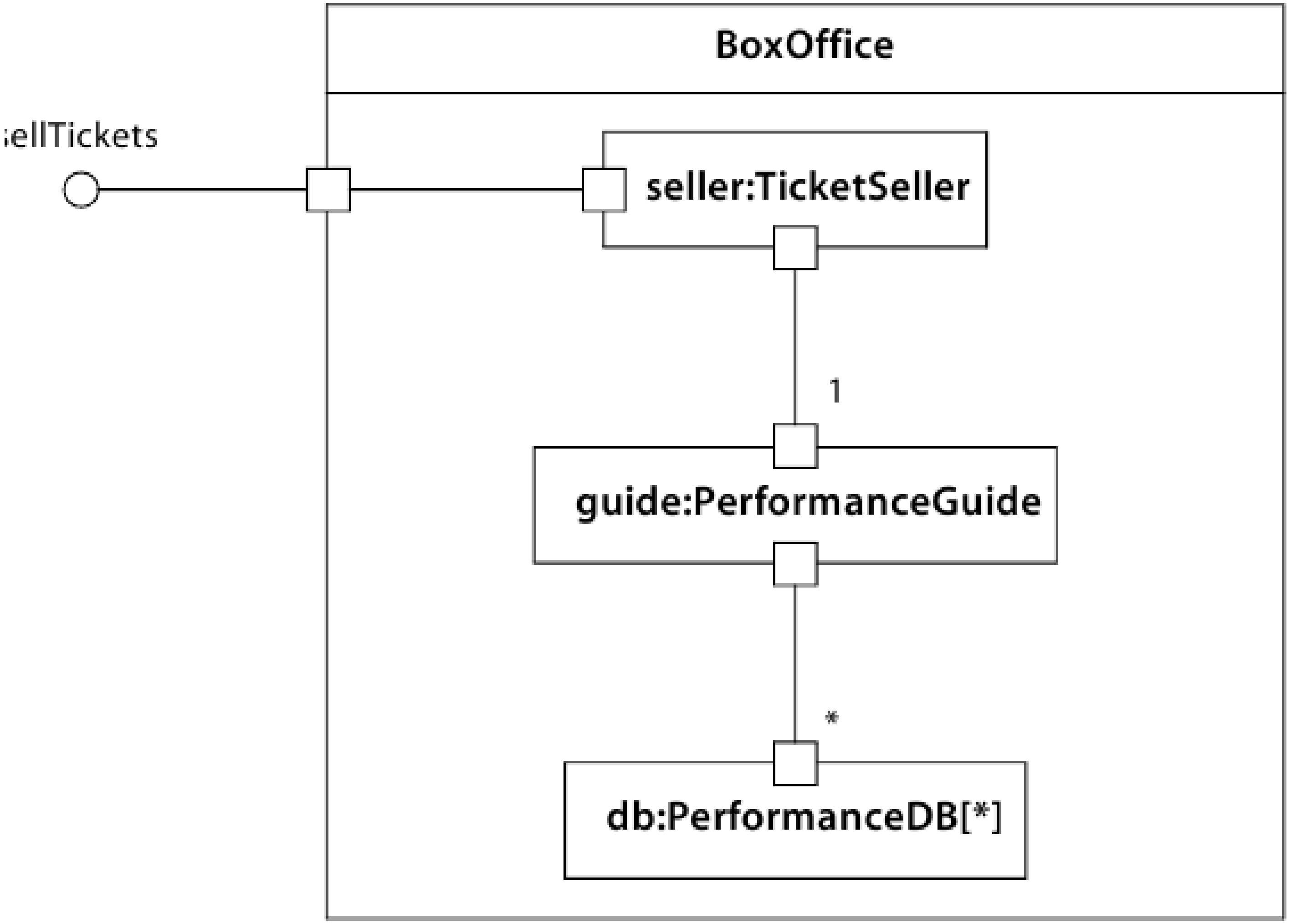
- ❑Classes are drawn as rectangles. Lists of attributes and operations are shown in separate compartments.
  - ❑The compartments can be suppressed when full detail is not needed.
- ❑A class may appear on several diagrams.
- ❑The attributes and operations are often shown on one diagram (the "home" diagram) and suppressed on other diagrams.
- ❑Relationships among classes are drawn as paths connecting class rectangles.
- ❑The different kinds of relationships are distinguished by line texture and by adornments on the paths or their ends.

DESIGN VIEWS

- The previous views model the concepts in the application from a logical view point.
- The design views model the design structure of the application itself.
  - These views provide an opportunity to map classes onto implementation components and expand high-level classes into supporting structure.
- Implementation diagrams include the internal structure diagram, the collaboration diagram, and the component diagram.

INTERNAL STRUCTURE DIAGRAM

- An internal structure diagram shows the decomposition of a class.



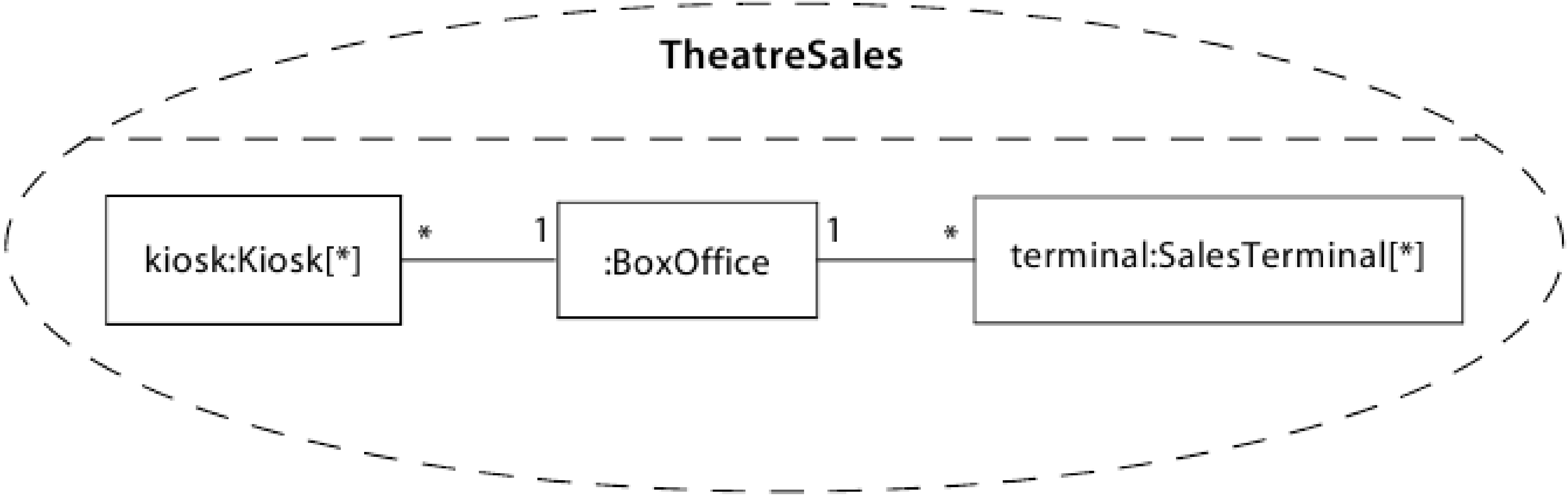
- From the above class diagram the classes are decomposed into three parts:
  - a ticket seller interface,
  - a performance guide that retrieves performances according to date and other criteria, and
  - a set of databases that contain the data on the performances and the tickets.
- Each part interacts through a well-defined interface specified by its ports.
- The ticket selling system interacts with the outside through a port.
- Messages on this port are dispatched to the ticket seller class, but the internal structure of the box office class is hidden from outside clients.

COLLABORATION DIAGRAM

❑A collaboration is a contextual relationship among a set of objects that work together to fulfill some purpose.

❑It contains a collection of roles—contextual slots within a generic pattern that can be played by, or bound to, individual objects.

❑There may be connectors providing contextual relationships among the roles.

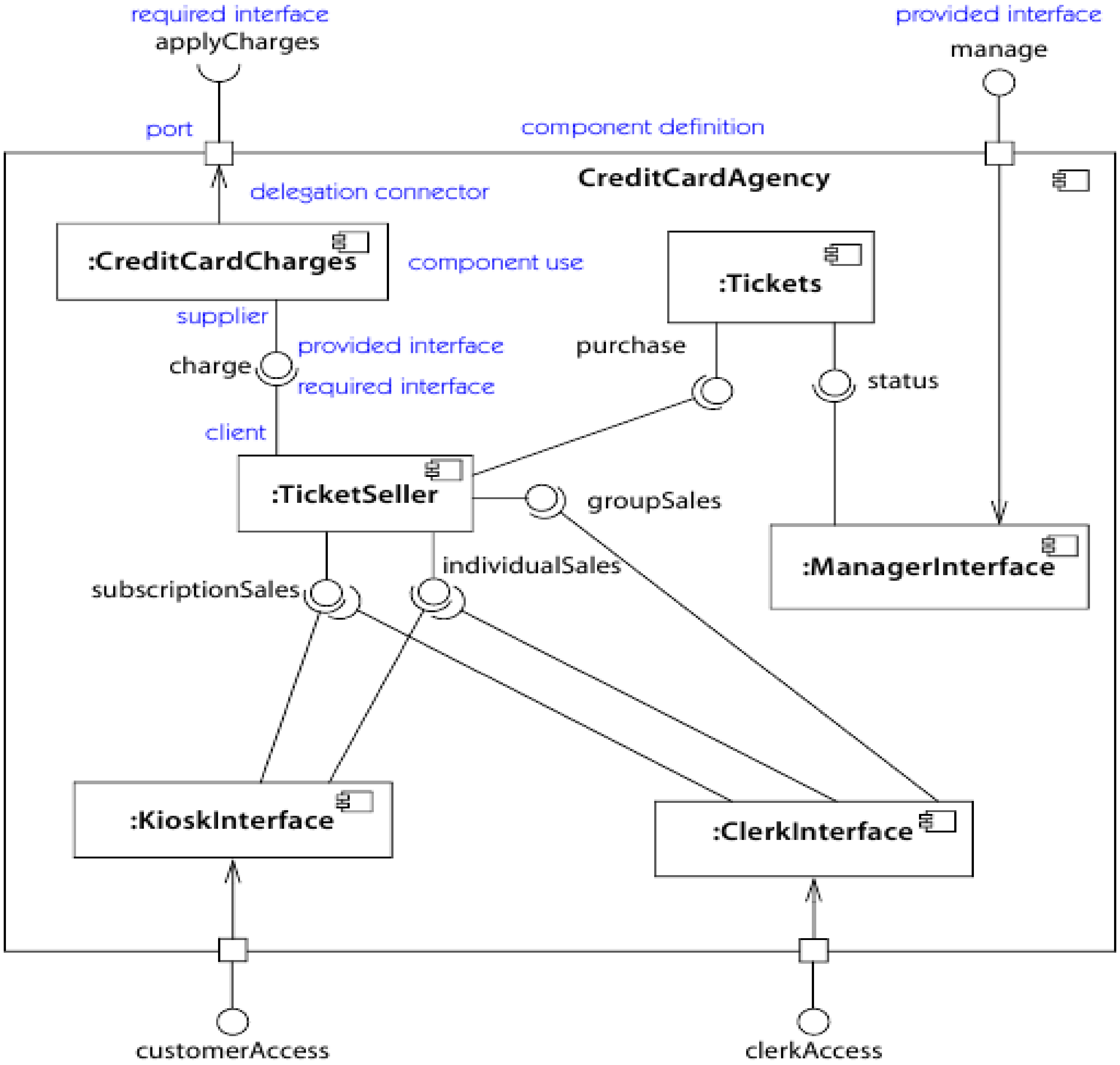

TheatreSales
kiosk:Kiosk[*]
*
1
:BoxOffice
1
*
terminal:SalesTerminal[*]

- Three kinds of separate components interact to provide the functionality of the system:
  - kiosks, sales terminals, and the box office application.
- These distinct components are not owned by a single overall class, but they cooperate in well-defined ways to provide services to the users.

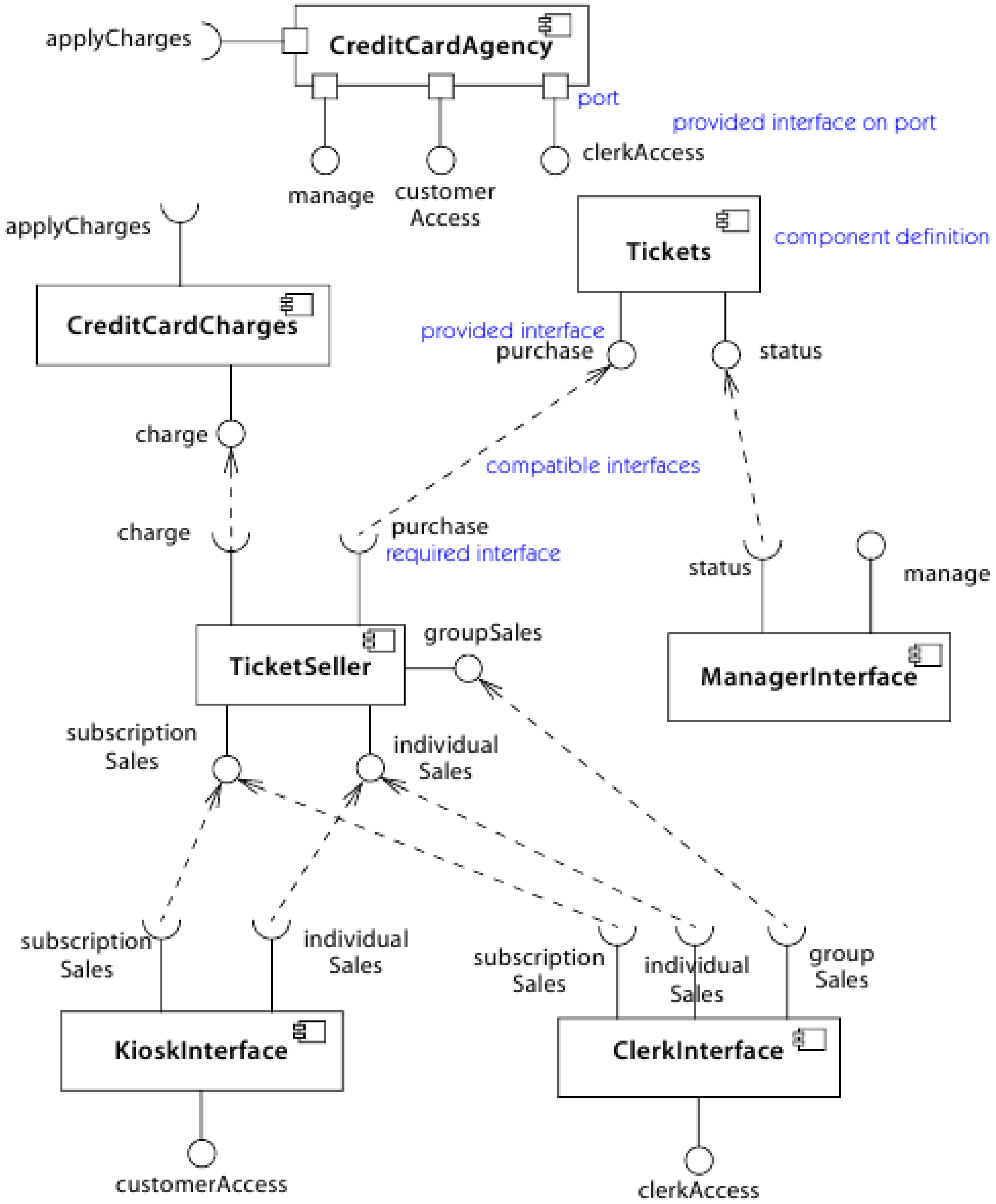
COMPONENT DIAGRAM

- A component diagram shows the components in a system—that is, the software units from which the application is constructed—as well as the dependencies among components so that the impact of a proposed change can be assessed.

COMPONENT DEFINITION

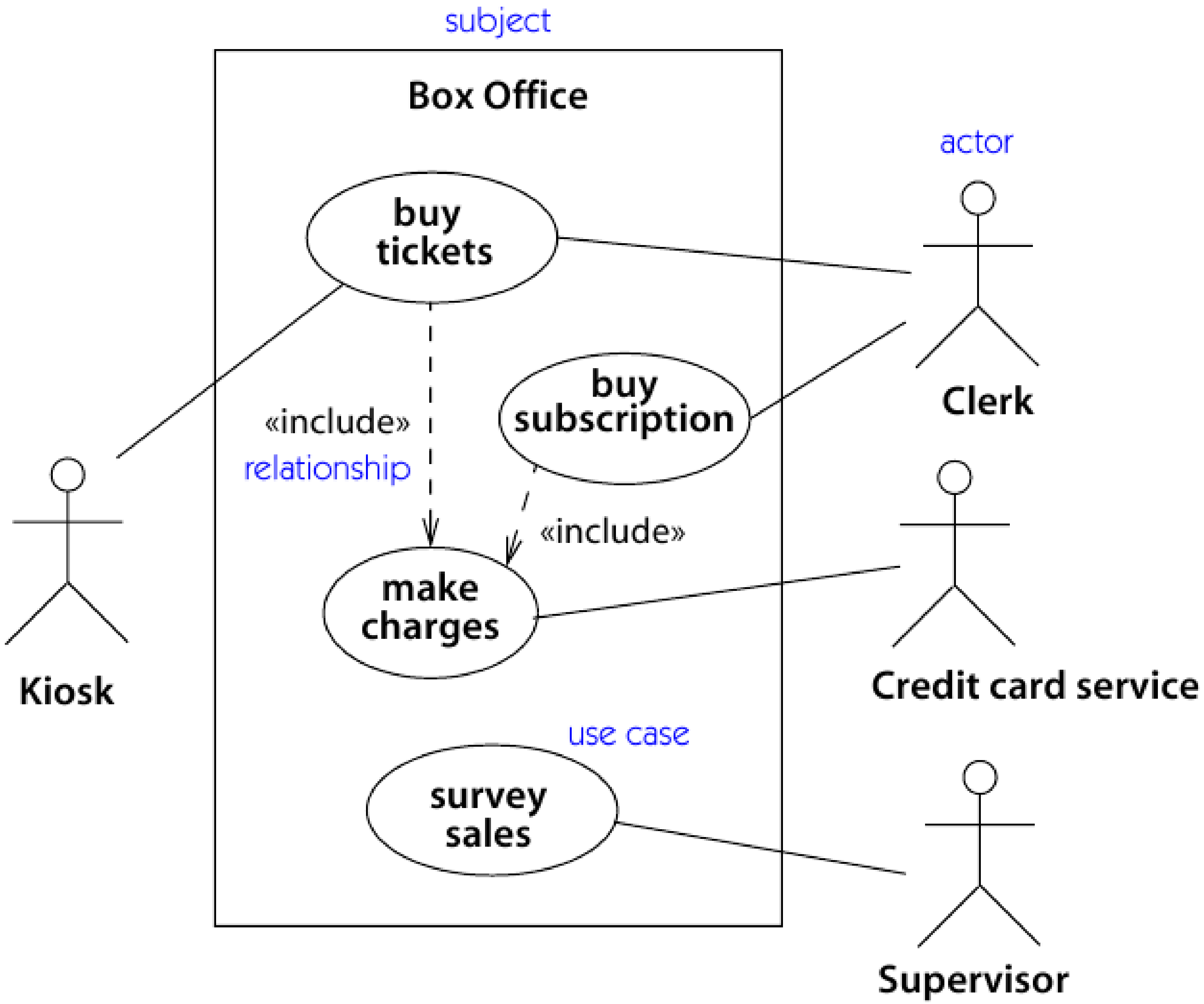


- The following figure shows a component diagram for the components used in the credit card agency component.
- The dashed dependency lines show compatible provided and required interfaces, but when the interfaces have the same names the dependency lines are redundant.
- In this example, the component diagram adds little to the internal structure diagram.
- In a larger example, the component diagram would combine components used in many different places.


applyCharges
CreditCardAgency
port
provided interface on port
manage
customer Access
clerkAccess
applyCharges
Tickets
component definition
CreditCardCharges
provided interface
purchase
status
charge
compatible interfaces
charge
purchase
required interface
status
manage
TicketSeller
groupSales
ManagerInterface
subscription Sales
individual Sales
subscription Sales
individual Sales
subscription Sales
individual Sales
group Sales
KioskInterface
ClerkInterface
customerAccess
clerkAccess

USE CASE VIEW

- The use case view models the functionality of a subject (such as a system) as perceived by outside agents, called actors, that interact with the subject from a particular viewpoint.
- A use case is a unit of functionality expressed as a transaction among actors and the subject.
- The purpose of the use case view is to list the actors and use cases and show which actors participate in each use case.
- The behavior of use cases is expressed using dynamic views, particularly the interaction view.


subject
Box Office
actor
buy tickets
Clerk
buy subscription
«include»
relationship
«include»
make charges
Kiosk
Credit card service
use case
survey sales
Supervisor

- Use cases can also be described at various levels of detail.
- They can be factored and described in terms of other, simpler use cases.
- A use case is implemented as a collaboration in the interaction view.

STATE MACHINE VIEW

❑ A state machine models the possible life histories of an object of a class.

❑ A state machine contains states connected by transitions.

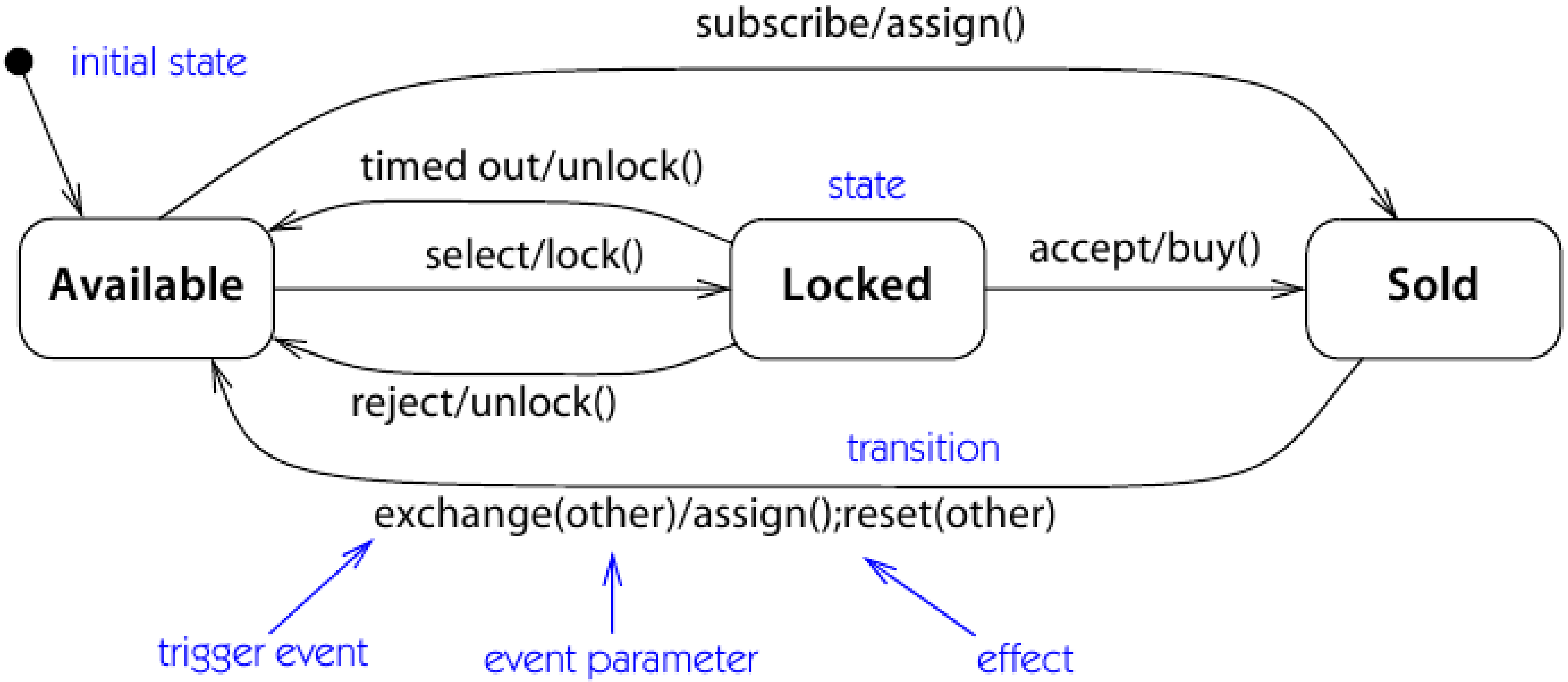
❑ Each state models a period of time during the life of an object during which it satisfies certain conditions.

❑ When an event occurs, it may cause the firing of a transition that takes the object to a new state.

❑ When a transition fires, an effect (action or activity) attached to the transition may be executed.

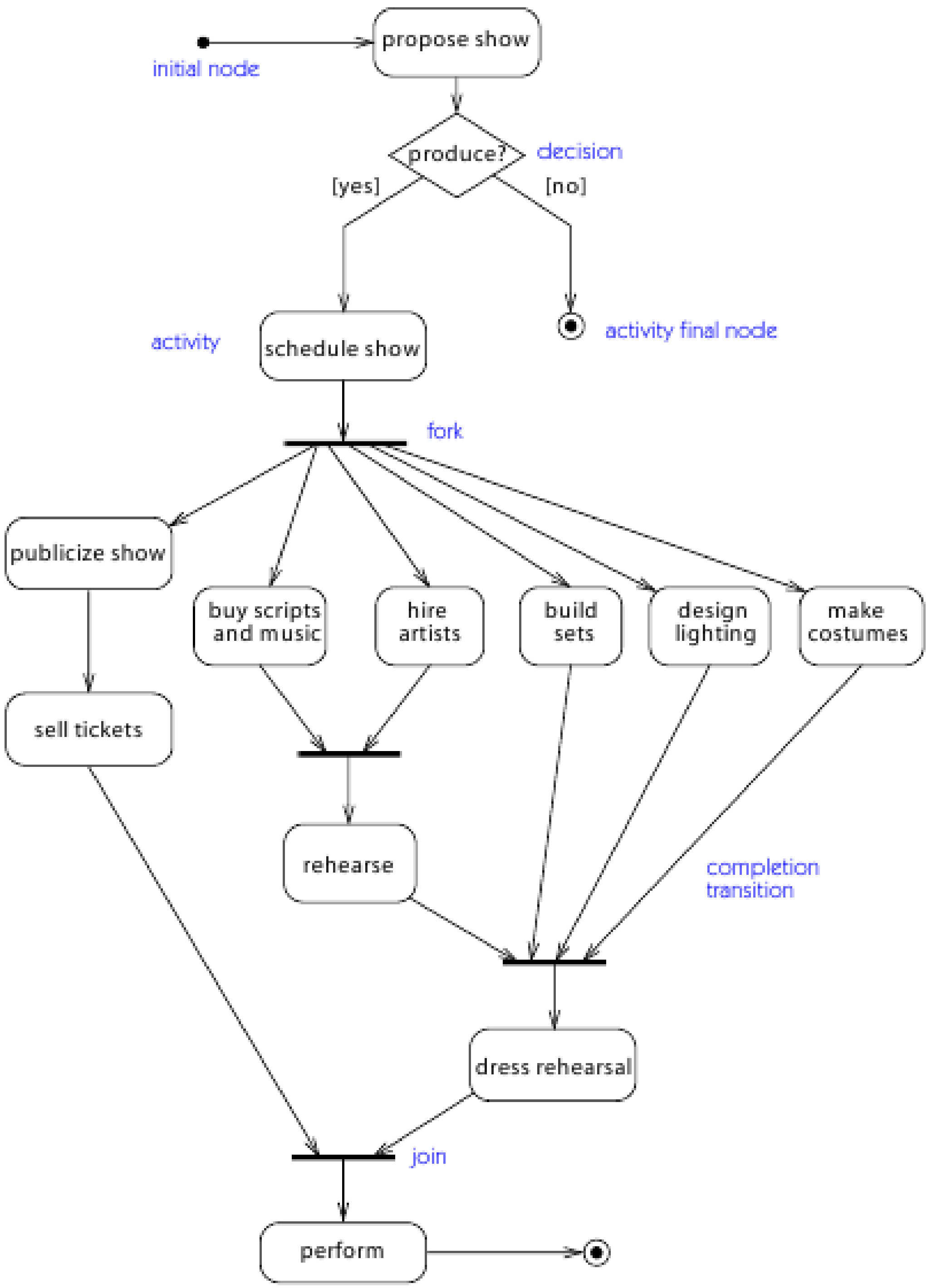
❑ State machines are shown as state machine diagrams.

- State machines may be used to describe user interfaces, device controllers, and other reactive subsystems.
- They may also be used to describe passive objects that go through several qualitatively distinct phases during their lifetime, each of which has its own special behavior.

subscribe/assign()
initial state
timed out/unlock()
state
Available
select/lock()
Locked
accept/buy()
Sold
reject/unlock()
transition
exchange(other)/assign();reset(other)
trigger event
event parameter
effect

ACTIVITY VIEW

- ❑An activity shows the flow of control among the computational activities involved in performing a calculation or a workflow.
- ❑An action is a primitive computational step. An activity node is a group of actions or sub-activities.
- ❑An activity describes both sequential and concurrent computation. Activities are shown on activity diagrams.

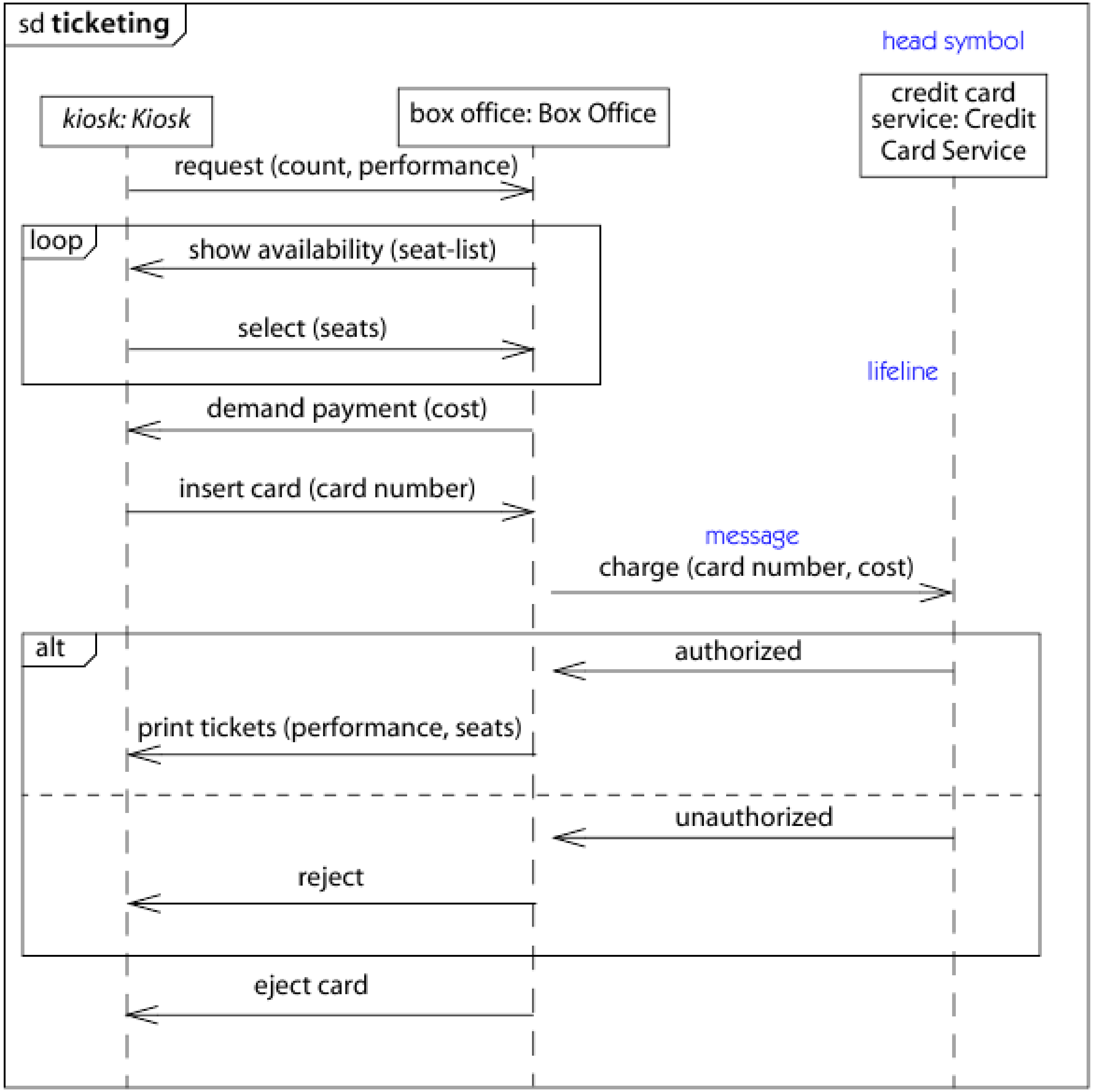

propose show
initial node
produce?
decision
[yes]
[no]
activity
schedule show
activity final node
fork
publicize show
buy scripts and music
hire artists
build sets
design lighting
make costumes
sell tickets
rehearse
completion transition
dress rehearsal
join
perform

INTERACTION VIEW

- The interaction view describes sequences of message exchanges among the parts of a system.
- An interaction is based on a structured classifier or a collaboration.
- A role is a slot that may be filled by objects in a particular use of an interaction.
- The interaction view is displayed in two diagrams focused on different aspects: sequence diagrams and communication Diagrams.

SEQUENCE DIAGRAM

- A sequence diagram shows a set of messages arranged in time sequence.
- Each role is shown as a lifeline—that is, a vertical line that represents the role over time through the entire interaction. Messages are shown as arrows between lifelines.
- A sequence diagram can show a scenario—that is, an individual history of a transaction.
- Structured control constructs, such as loops, conditionals, and parallel execution, are shown as nested rectangles with keywords and one or more regions.
- One use of a sequence diagram is to show the behavior sequence of a use case.
- When the behavior is implemented, each message on a sequence diagram corresponds to an operation on a class or an event trigger on a transition in a state machine.

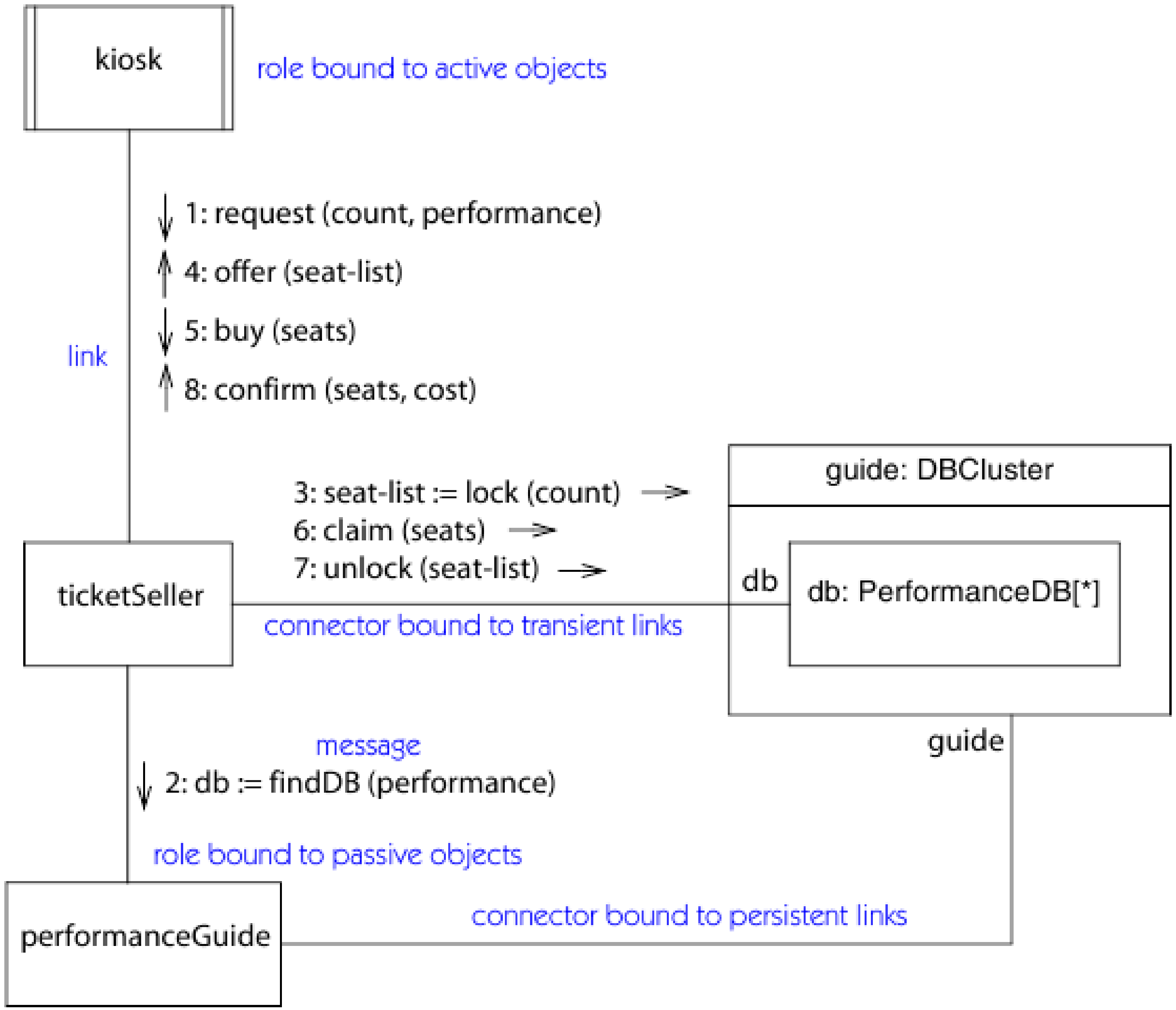

sd ticketing
head symbol
kiosk: Kiosk
box office: Box Office
credit card service: Credit Card Service
request (count, performance)
loop
show availability (seat-list)
select (seats)
lifeline
demand payment (cost)
insert card (card number)
message
charge (card number, cost)
alt
authorized
print tickets (performance, seats)
unauthorized
reject
eject card

Sequence diagram

COMMUNICATION DIAGRAM

- ❑A communication diagram shows roles in an interaction as a geometric arrangement.
- ❑Each rectangle shows a role—more precisely, a lifeline representing the life of an object over time.
- ❑The messages among objects playing roles are shown as arrows attached to connectors.
- ❑The sequence of messages is indicated by sequence numbers prepended to message descriptions.
- ❑One use of a communication diagram is to show the implementation of an operation

COMMUNICATION DIAGRAM

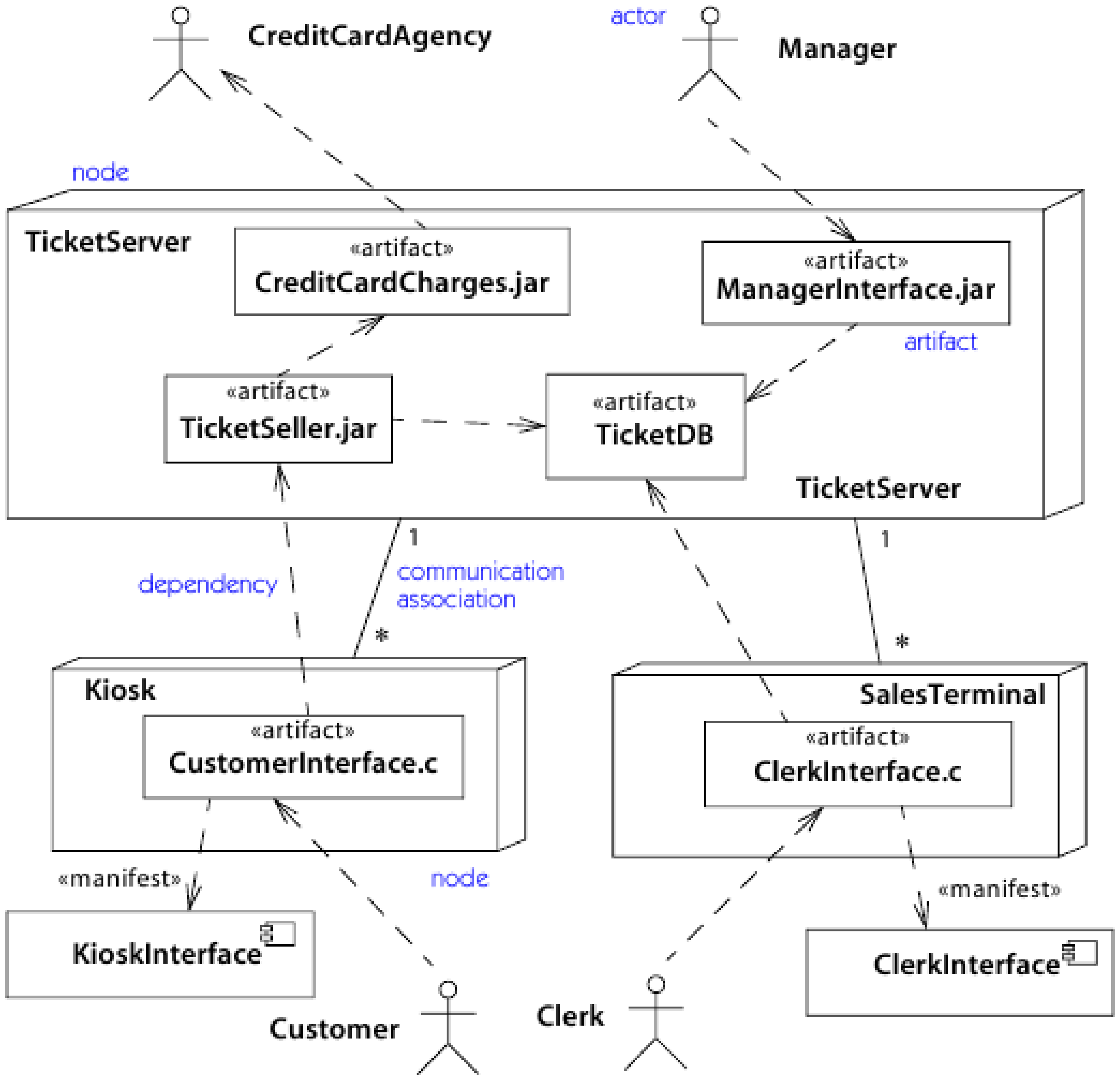

kiosk
role bound to active objects
1: request (count, performance)
4: offer (seat-list)
5: buy (seats)
8: confirm (seats, cost)
link
guide: DBCluster
3: seat-list := lock (count)
6: claim (seats)
7: unlock (seat-list)
db
db: PerformanceDB[*]
ticketSeller
connector bound to transient links
message
guide
2: db := findDB (performance)
role bound to passive objects
connector bound to persistent links
performanceGuide

- ❑Both sequence diagrams and communication diagrams show interactions, but they emphasize different aspects.
- ❑A sequence diagram shows time sequence as a geometric dimension, but the relationships among roles are implicit.
- ❑A communication diagram shows the relationships among roles geometrically and relates messages to the connectors, but time sequences are less clear because they are implied by the sequence numbers.

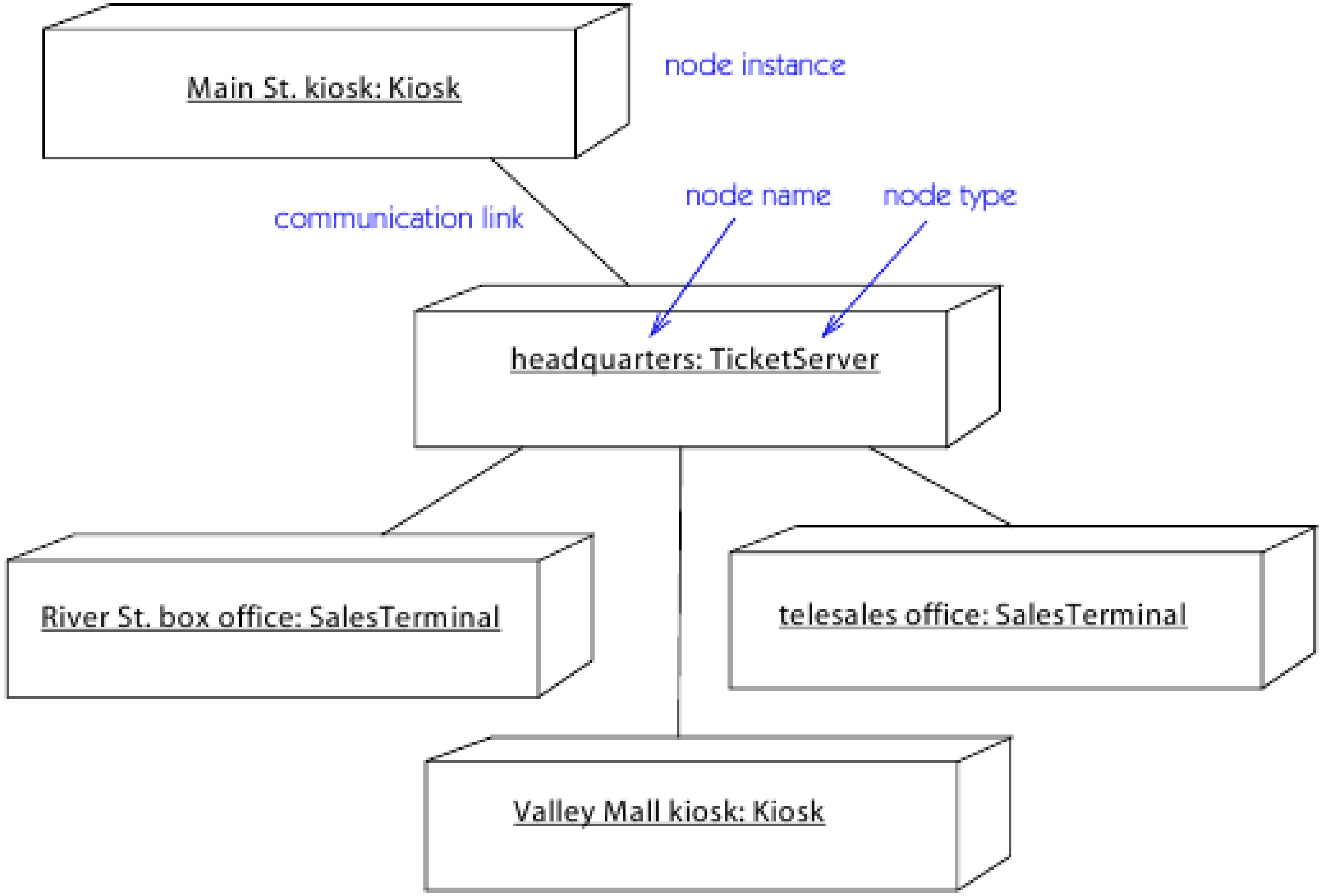
DEPLOYMENT VIEW

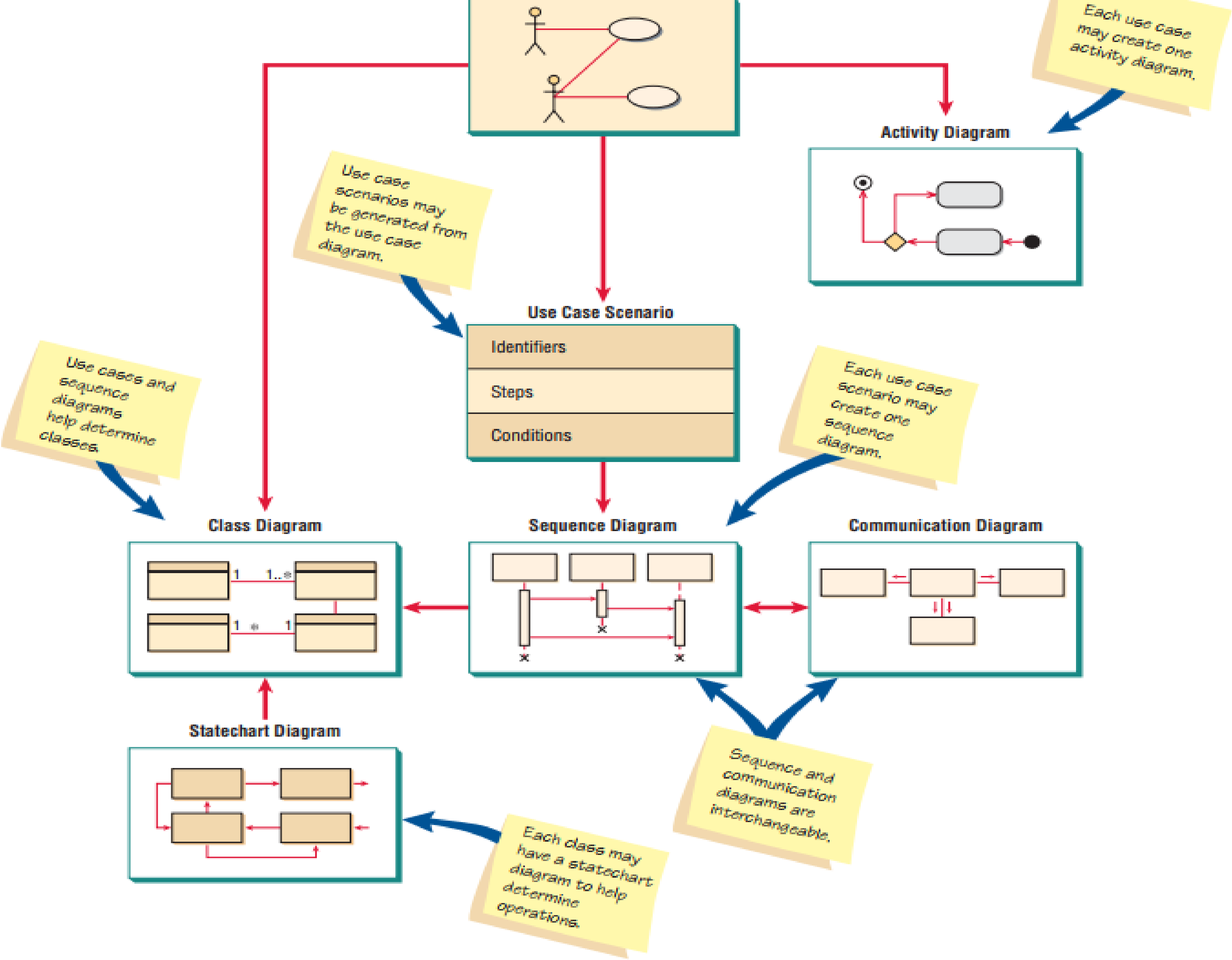
- A deployment diagram represents the deployment of run-time artifacts on nodes.
- An artifact is a physical implementation unit, such as a file.
- A node is a run-time resource, such as a computer, device, or memory.
- An artifact may be a manifestation (implementation) of one or more components. This view permits the consequences of distribution and resource allocation to be assessed.

DEPLOYMENT DIAGRAM (DESCRIPTOR LEVEL)



DEPLOYMENT DIAGRAM (INSTANCE LEVEL)





Each use case may create one activity diagram.

Activity Diagram

Use case scenarios may be generated from the use case diagram.

Use Case Scenario

Identifiers

Steps

Conditions

Use cases and sequence diagrams help determine classes.

Each use case scenario may create one sequence diagram.

Class Diagram

Sequence Diagram

Communication Diagram

Statechart Diagram

Sequence and communication diagrams are interchangeable.

Each class may have a statechart diagram to help determine operations.