2IW80 Software specification and architecture

# Recapitulation session

Kees Huizing
Sarmen Keshishzadeh
Alexander Serebrenik
Ulyana Tikhonova
Anton Wijs

Technische Universiteit
Eindhoven
University of Technology

Where innovation starts

# Exam

- **April 15, 9:00 – 12:00**
- **Location: check OASE**

- **Part I. 20 multiple choice questions (4 options)**
  - 2, if Indicated = Correct
  - 1, if Indicated $\subseteq$ Correct
  - 0, otherwise

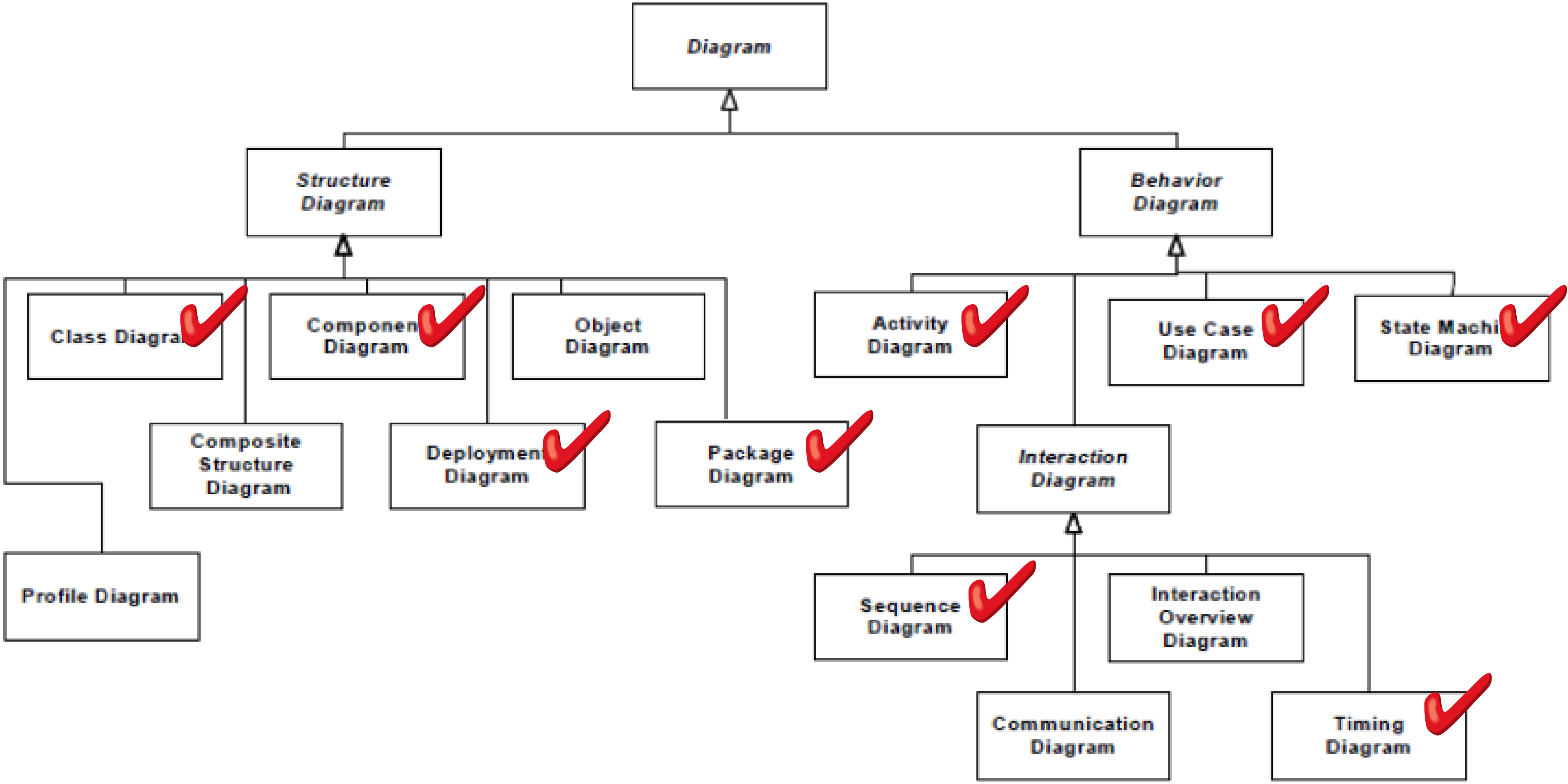
- **Part II. Five modeling exercises (15 points each)**
  - Choose four
  - We’ll grade the first four

# Recapitulation session

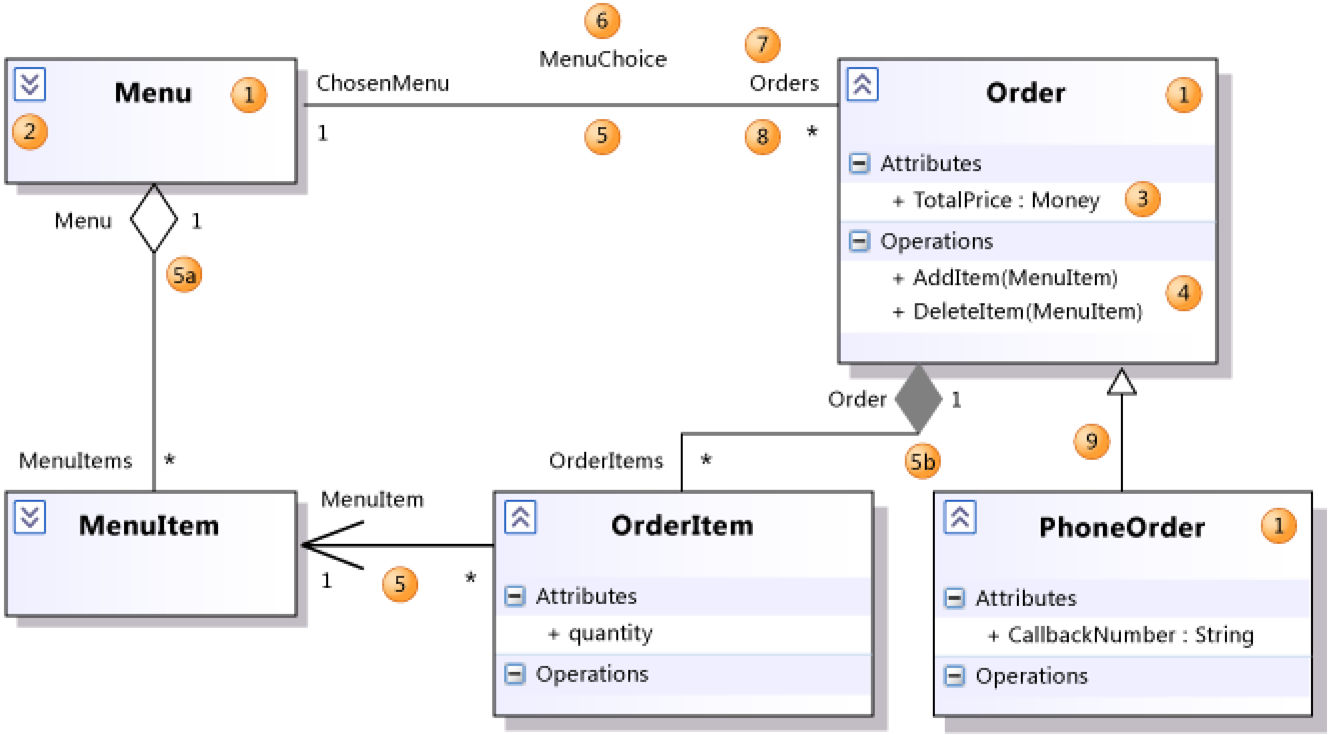
- What did you ask us to recapitulate…
  - **UML diagrams**
  - **Architectural styles**
  - **Viewpoints**
  - **Event-B**
  - ***topics that are most helpful in writing the exam***

**Exercises in this session are representative of the exam questions**

# UML diagram types: Overview



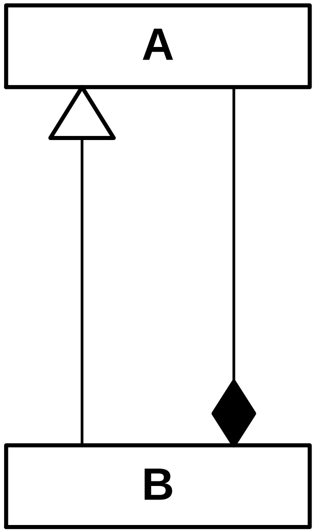
# UML class diagram: Overview



| | |
|---|---|
| 1 | Class |
| 3 | Attribute |
| 4 | Operation |
| 5 | Association |
| 5a | Aggregation |
| 5b | Composition |
| 6 | Association Name |
| 7 | Role Name |
| 8 | Multiplicity |
| 9 | Generalization |

http://i.msdn.microsoft.com/dynimg/IC315445.png

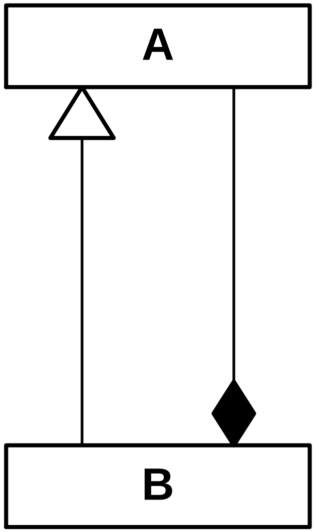
# Multiple choice question (1)



Class diagram on the left states that

a) A is a generalization of B
b) A is a composition of B
c) B is an implementation of A
d) The class diagram is syntactically incorrect

# Multiple choice question (1)



Class diagram on the left states that

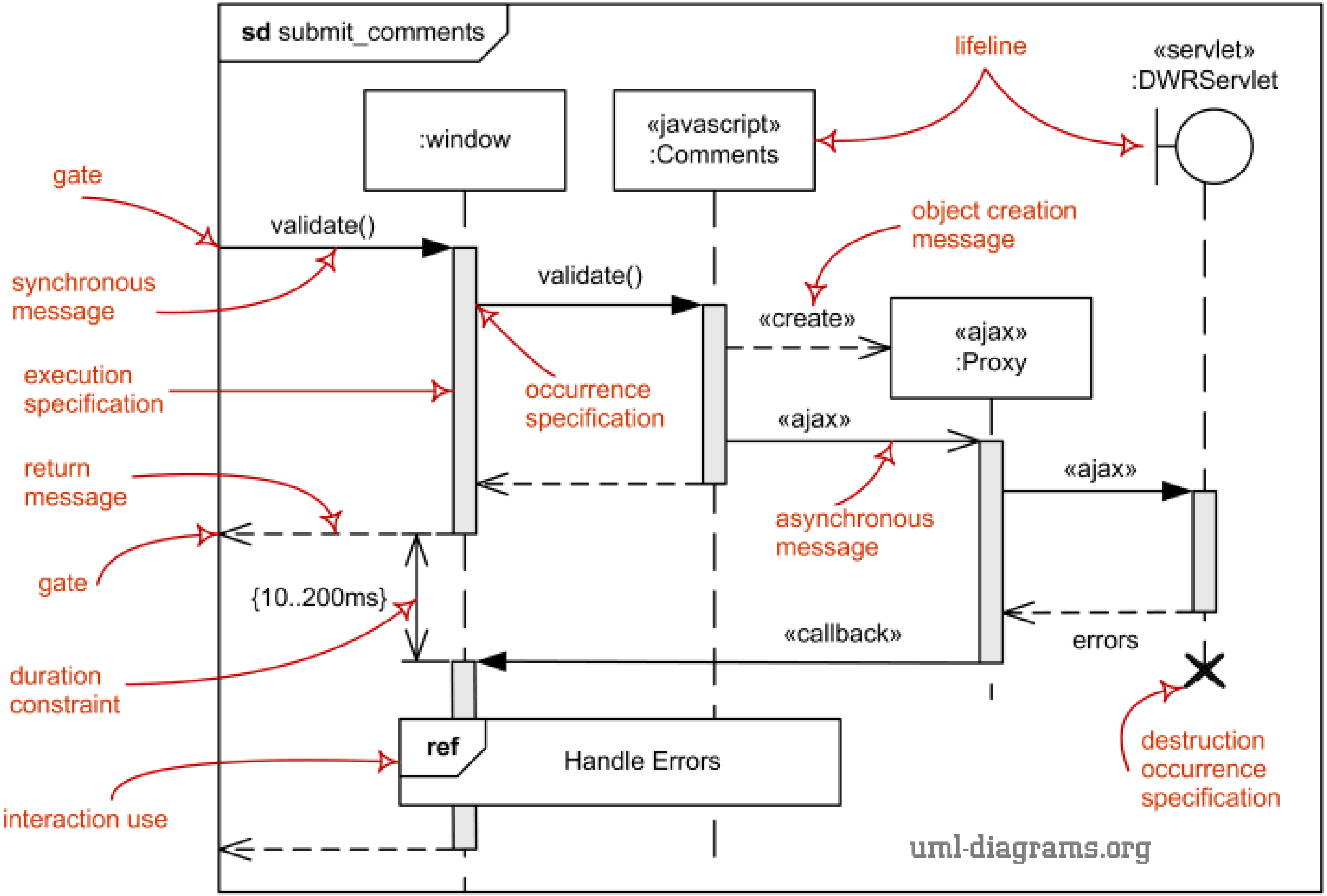
a) **A is a generalization of B**

b) A is a composition of B

c) B is an implementation of A

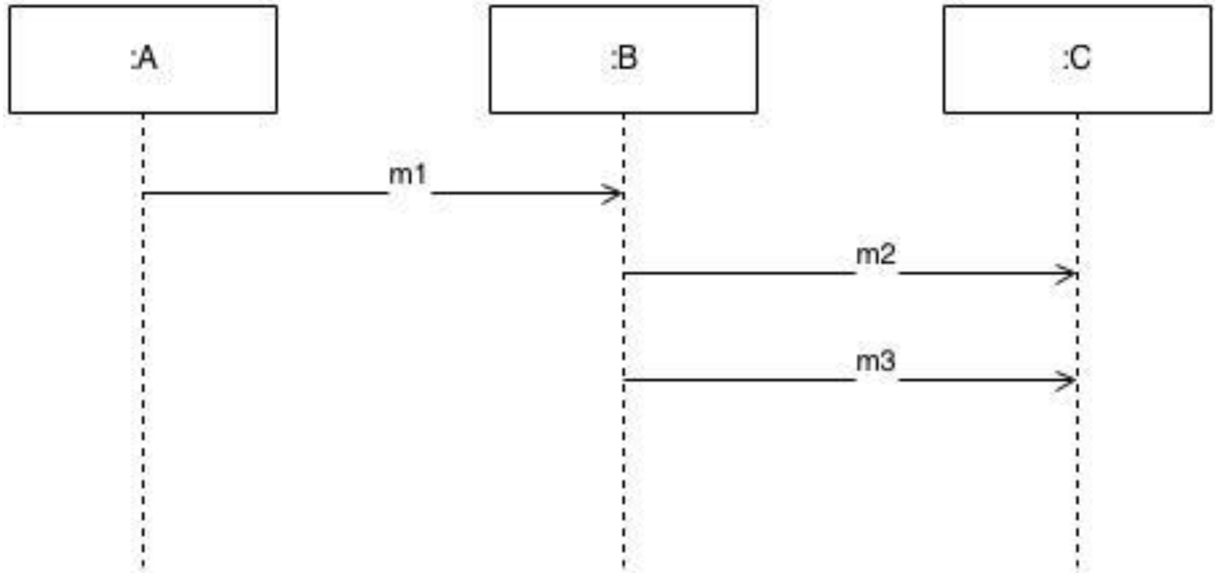
d) The class diagram is syntactically incorrect

# UML sequence diagram: Overview



http://www.uml-diagrams.org/sequence-diagrams.html

# Multiple choice question (2)



**Which of the following statements are not enforced by the sequence diagram?**

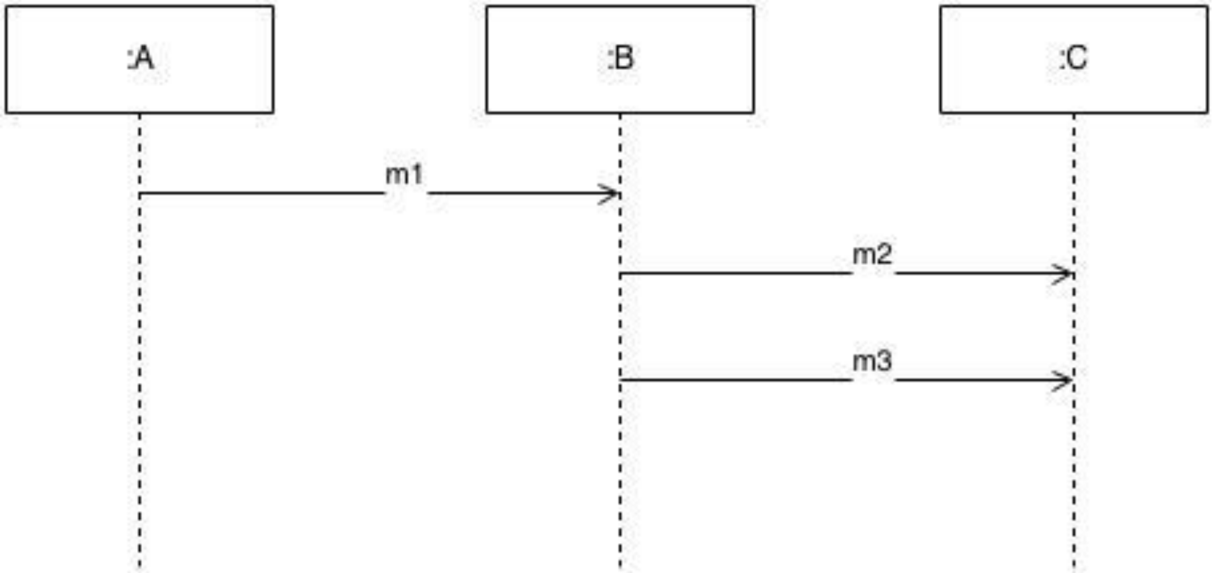
a) m1 is received before m2 is send

b) m2 is received before m3 is send

c) m1 is send before m2 is send

d) m2 is send before m3 is send

# Multiple choice question (2)



**Which of the following statements are not enforced by the sequence diagram?**

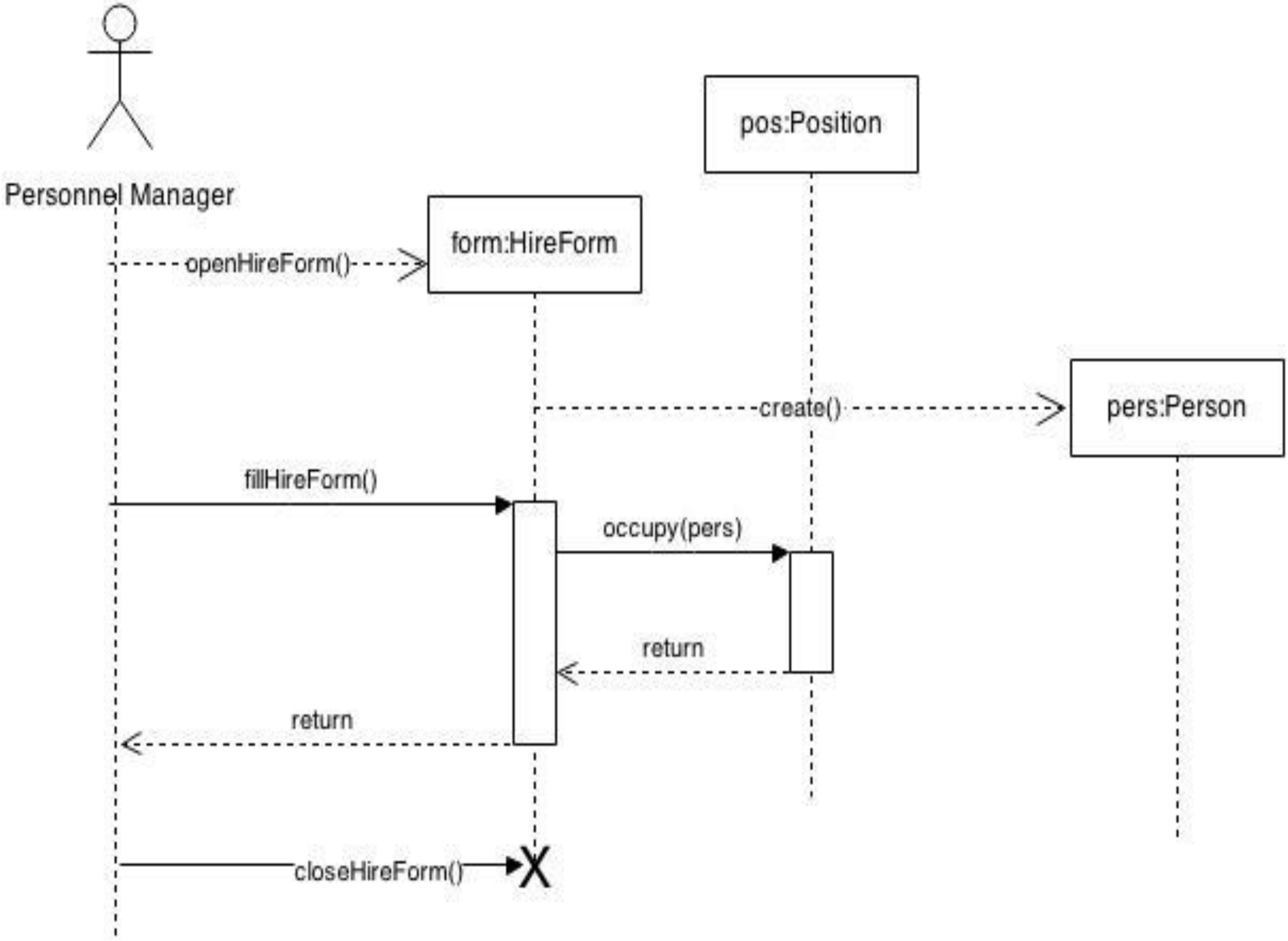
a) m1 is received before m2 is send

**b) m2 is received before m3 is send**

c) m1 is send before m2 is send

d) m2 is send before m3 is send

# Multiple choice question (3)



The method **occupy** should be implemented in

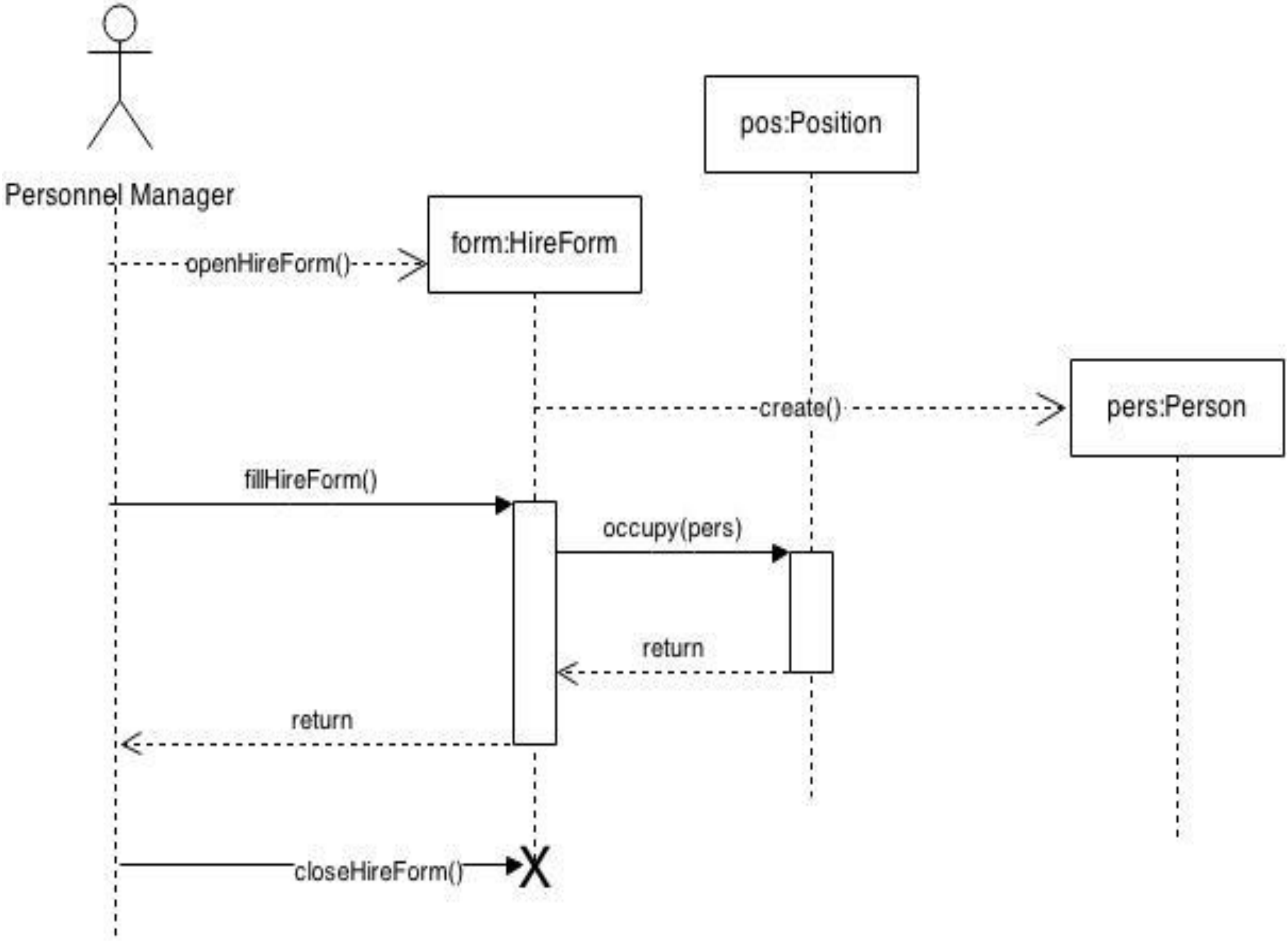
a) HireForm

b) Person

c) Position

d) not enough information in the diagram

# Multiple choice question (3)



The method **occupy** should be implemented in

a) HireForm

b) Person

**c) Position**

d) not enough information in the diagram

# Important!!!

- **Behavior should be consistent with structure**
  - **Behavior:** sequence diagrams, state machines, activity diagrams, …
  - **Structure:** class diagrams, package diagrams, …

# Exercise (4.2)

In a file system with a GUI, such as Microsoft’s Windows Explorer, the following objects were identified from a use case describing how to copy a file from a floppy disk to a hard disk: File, Icon, TrashCan, Folder, Disk, Pointer.

What are entity objects, boundary objects, and control objects?

# Exercise (4.2)

In a file system with a GUI, such as Microsoft's Windows Explorer, the following objects were identified from a use case describing how to copy a file from a floppy disk to a hard disk: File, Icon, TrashCan, Folder, Disk, Pointer.

What are entity objects, boundary objects, and control objects?

*Entity objects*: File, Folder, Disk

*Boundary objects*: Icon, Pointer, TrashCan

*Control objects*: none in this example.

Which object do you need to add to allow one to copy files? What kind of object is it? What would it be responsible for?

# Exercise (4.2)

In a file system with a GUI, such as Microsoft's Windows Explorer, the following objects were identified from a use case describing how to copy a file from a floppy disk to a hard disk: File, Icon, TrashCan, Folder, Disk, Pointer.

What are entity objects, boundary objects, and control objects?

- *Entity objects*: File, Folder, Disk
- *Boundary objects*: Icon, Pointer, TrashCan
- *Control objects*: none in this example.

Which object do you need to add to allow one to copy files? What kind of object is it? What would it be responsible for?

- CopyFile control object responsible for
  - Remembering the path of the destination folder
  - Checking if the file can be copied (access control and disk space).
  - Remembering the path of the original file
  - Initiating the file copying

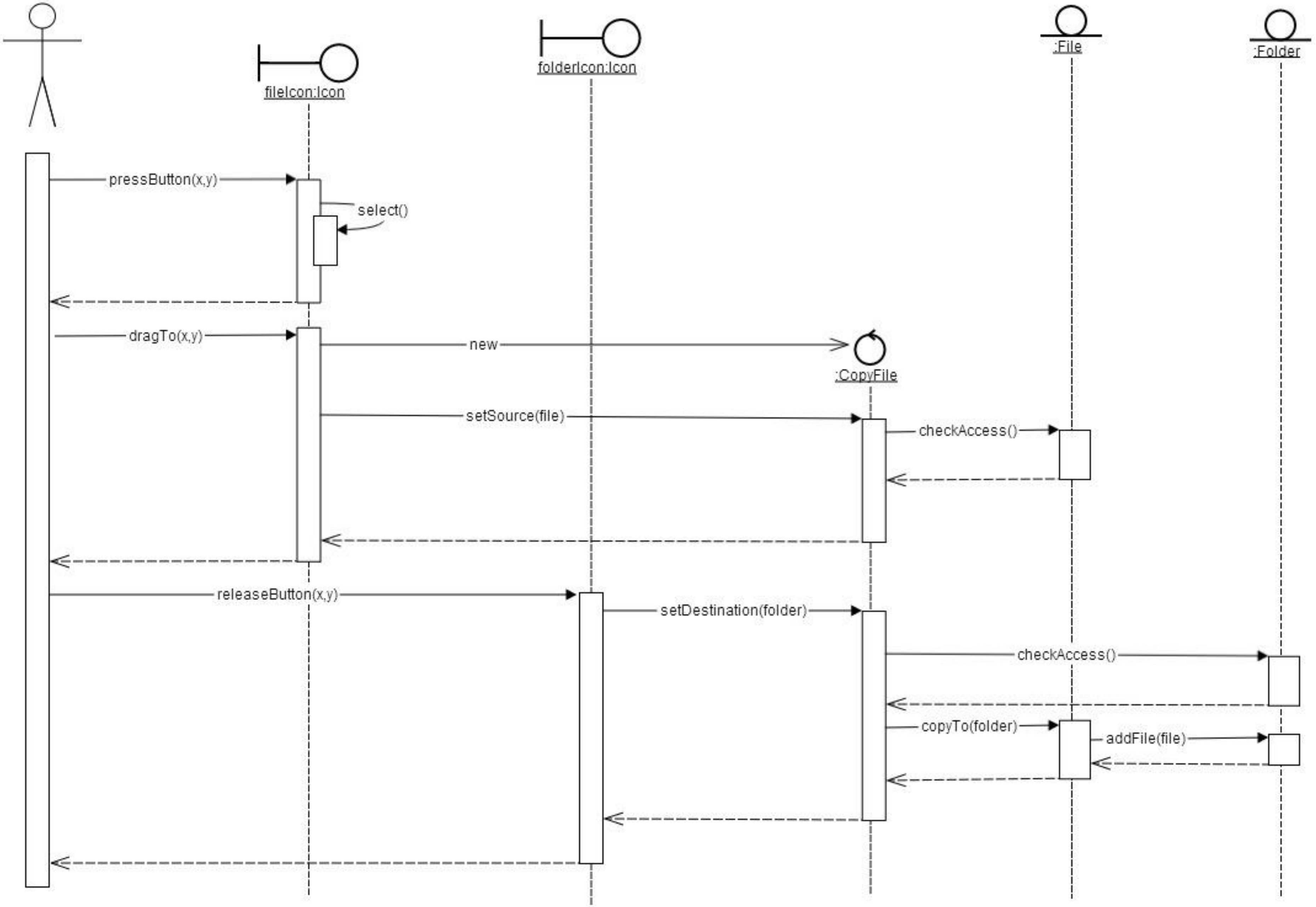
# Exercise (4.2)

In a file system with a GUI, such as Microsoft’s Windows Explorer, the following objects were identified from a use case describing how to copy a file from a floppy disk to a hard disk: File, Icon, TrashCan, Folder, Disk, Pointer.

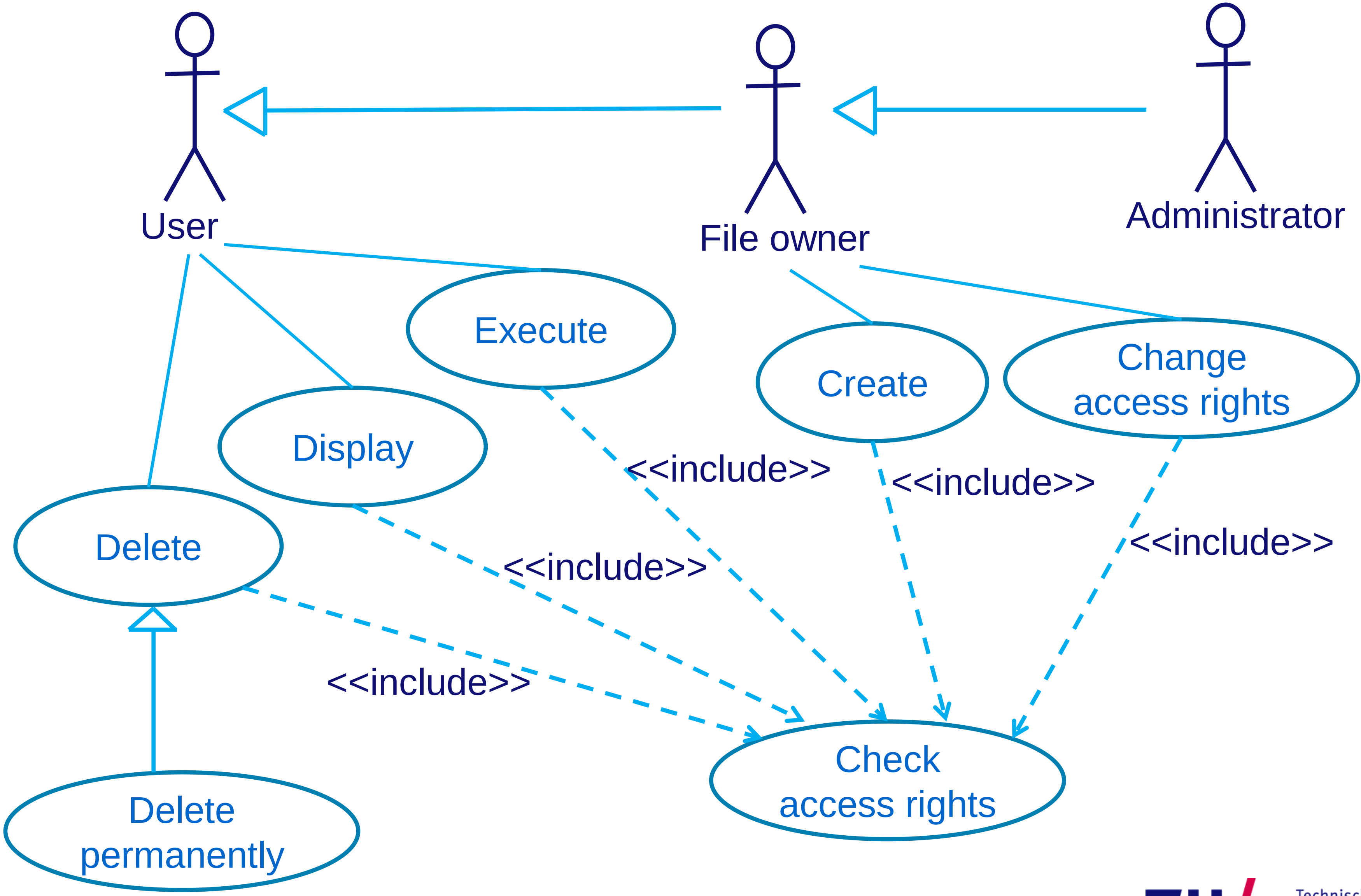
Draw the sequence diagram representing interactions resulting from dropping the file into a folder. Ignore the exceptional cases (such as lack of disk space or dropping the file to TrashCan).

Reminder:

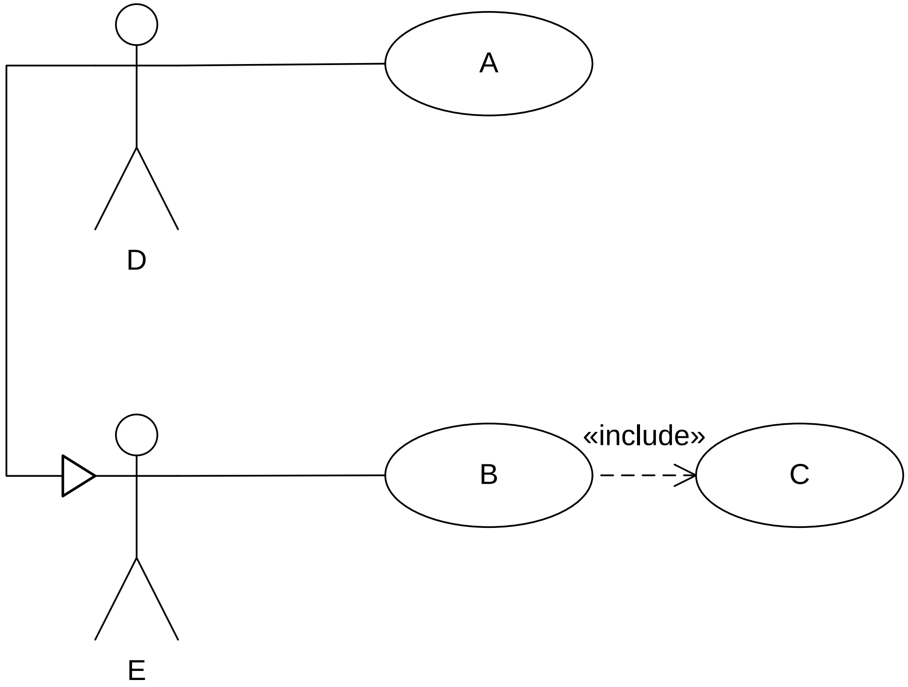
| | |
|---|---|
| *Entity objects*: | File, Folder, Disk |
| *Boundary objects*: | Icon, Pointer, TrashCan |
| *Control objects*: | CopyFile |


fileIcon:Icon
folderIcon:Icon
:File
:Folder
pressButton(x,y)
select()
dragTo(x,y)
new
:CopyFile
setSource(file)
checkAccess()
releaseButton(x,y)
setDestination(folder)
checkAccess()
copyTo(folder)
addFile(file)

# UML use cases: Overview



# Multiple choice question (4)



**Actor E is associated with activities**

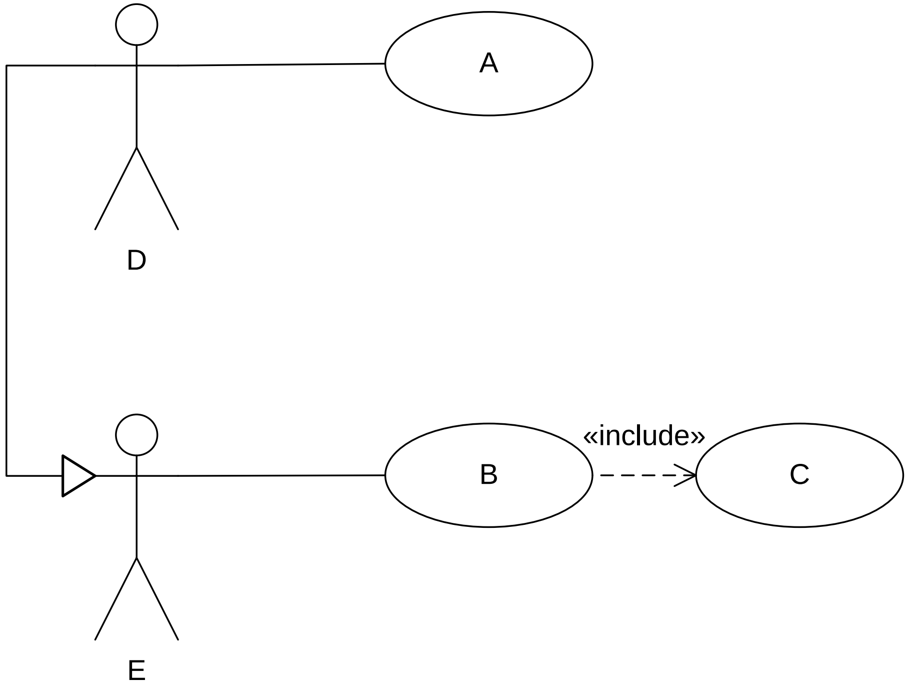
a) A

b) B

c) C

d) none

# Multiple choice question (4)



**Actor E is associated with activities**

a) A

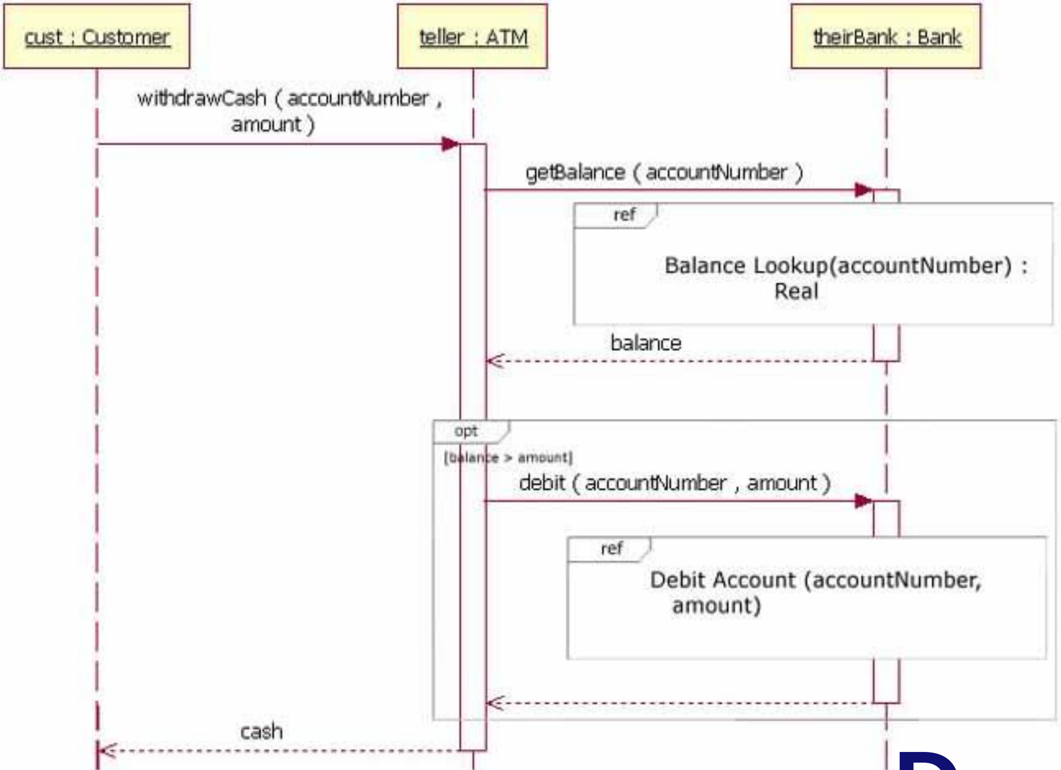
**b) B**

**c) C**

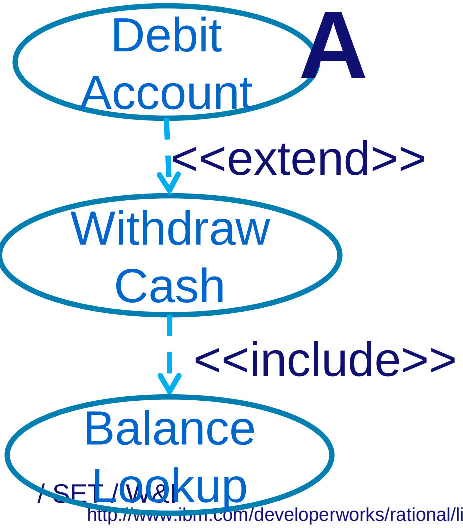
d) none

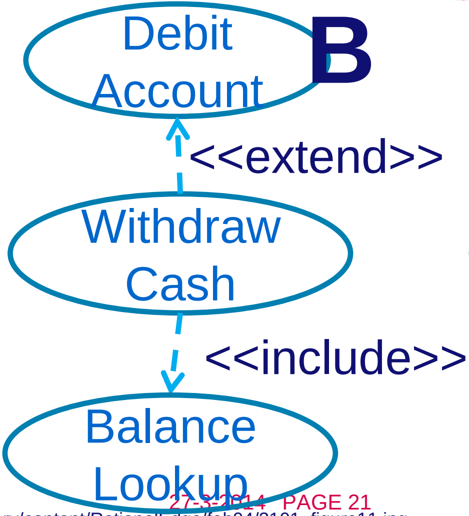
# Multiple choice question (5)

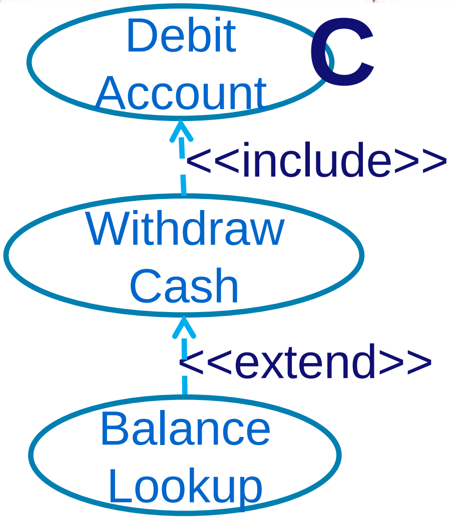
Let **Withdraw Cash**, **Balance Lookup** and **Debit Account** be use cases corresponding to the diagram above. Which one of the following use case diagrams is correct?

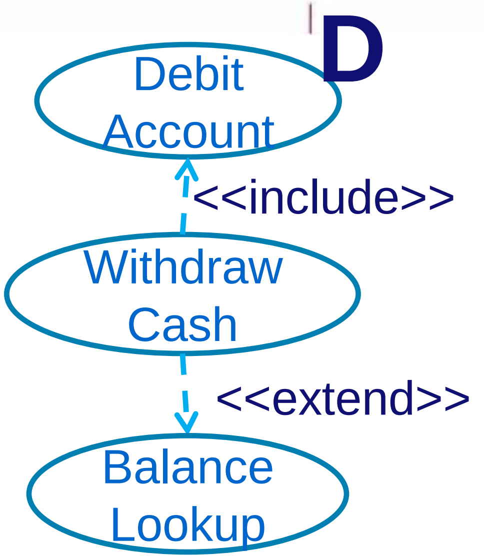

cust : Customer
teller : ATM
theirBank : Bank
withdrawCash ( accountNumber , amount )
getBalance ( accountNumber )
ref
Balance Lookup(accountNumber) : Real
balance
opt
[balance > amount]
debit ( accountNumber , amount )
ref
Debit Account (accountNumber, amount)
cash

http://www.ibm.com/developerworks/rational/library/content/RationalEdge/feb04/3101_figure11.jpg

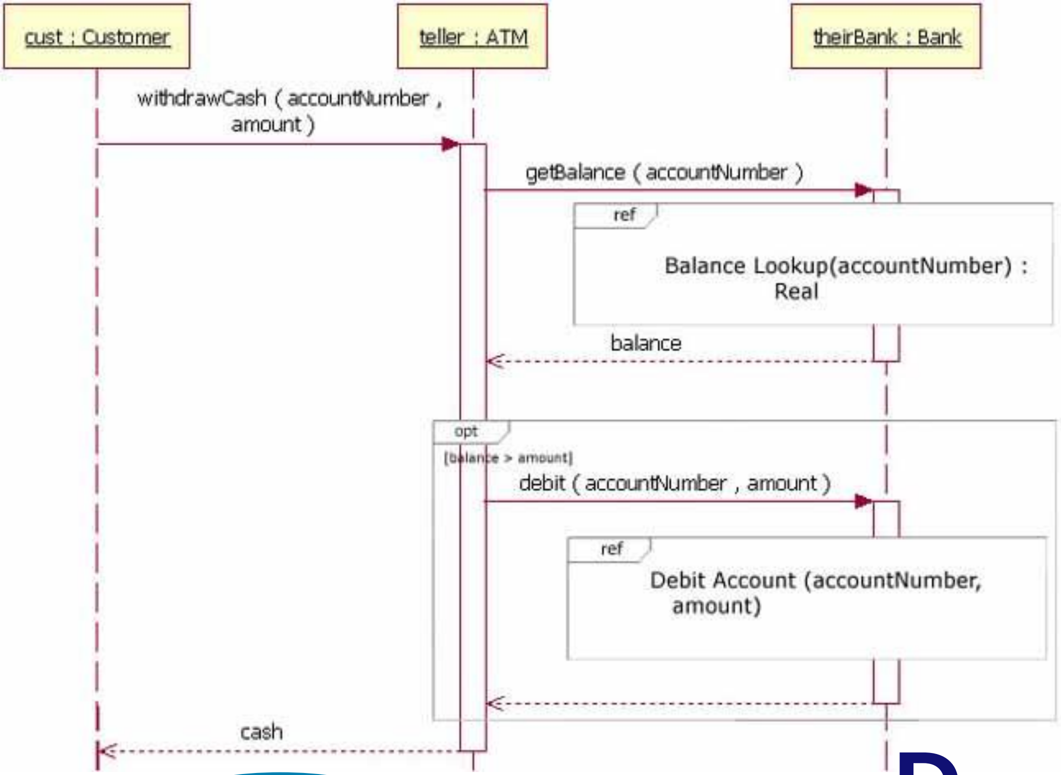






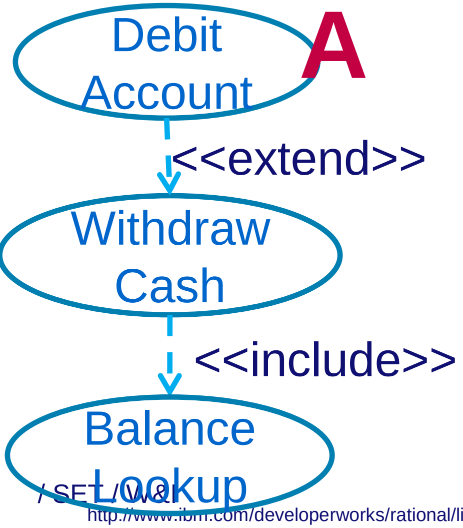


# Multiple choice question (5)

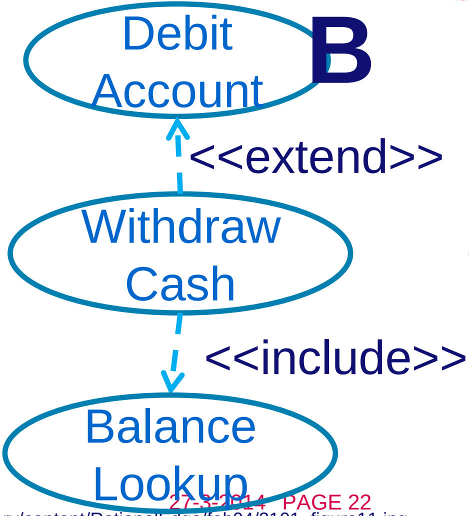

cust : Customer
teller : ATM
theirBank : Bank
withdrawCash ( accountNumber , amount )
getBalance ( accountNumber )
ref
Balance Lookup(accountNumber) : Real
balance
opt
[balance > amount]
debit ( accountNumber , amount )
ref
Debit Account (accountNumber, amount)
cash

Let **Withdraw Cash**, **Balance Lookup** and **Debit Account** be use cases corresponding to the diagram above. Which one of the following use case diagrams is correct?

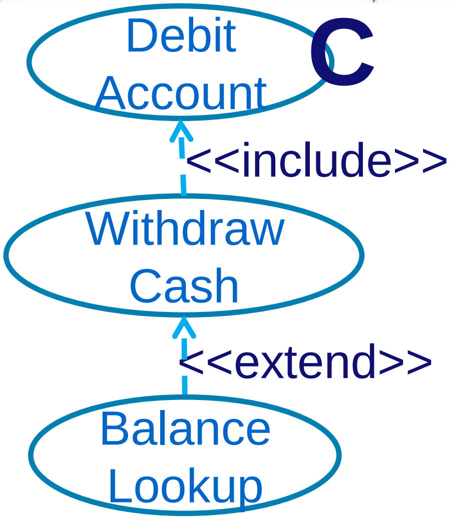
A


Debit Account
<<extend>>
Withdraw Cash
<<include>>
Balance Lookup

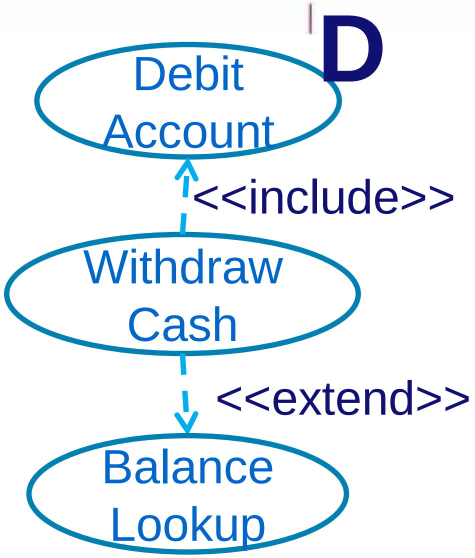
B


Debit Account
<<extend>>
Withdraw Cash
<<include>>
Balance Lookup

C

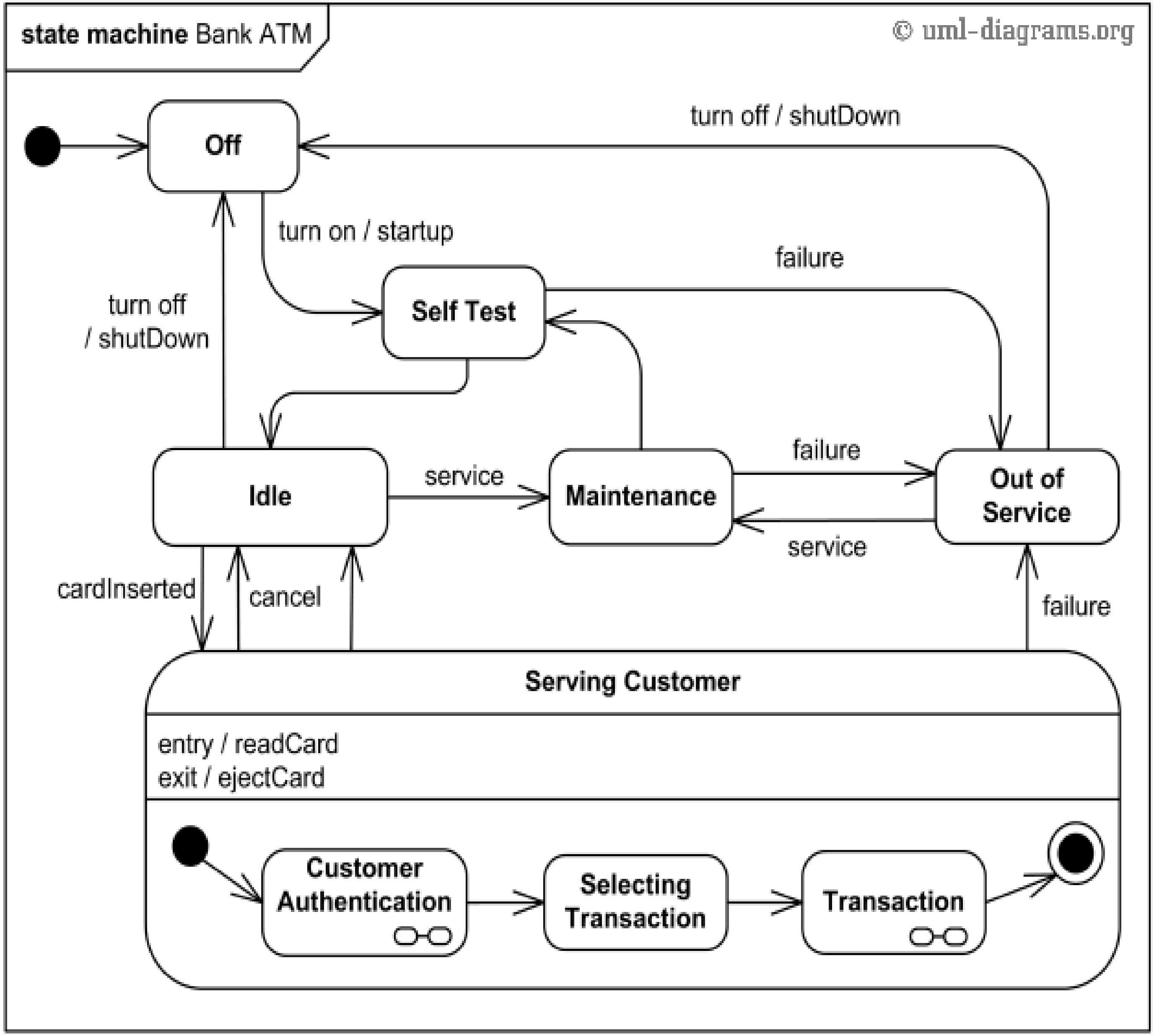

Debit Account
<<include>>
Withdraw Cash
<<extend>>
Balance Lookup

D

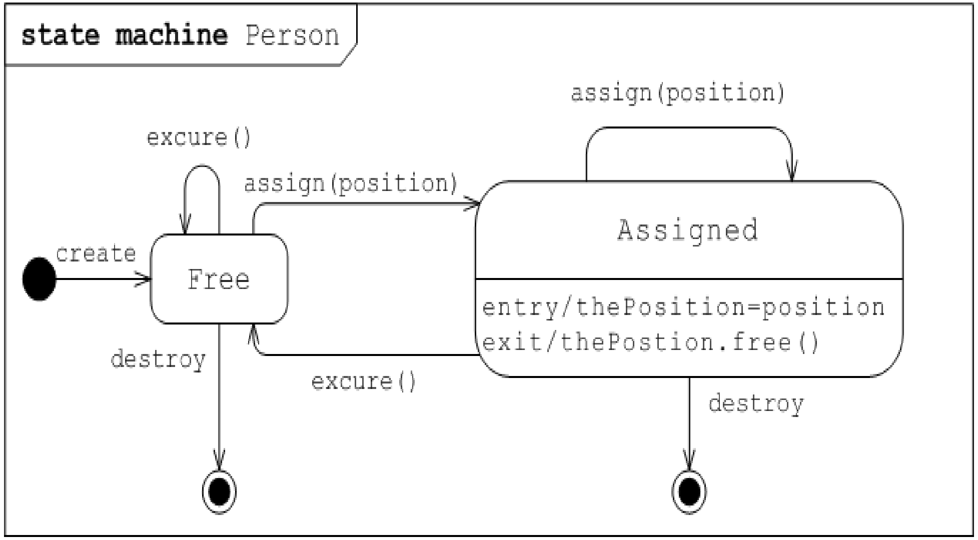

Debit Account
<<include>>
Withdraw Cash
<<extend>>
Balance Lookup

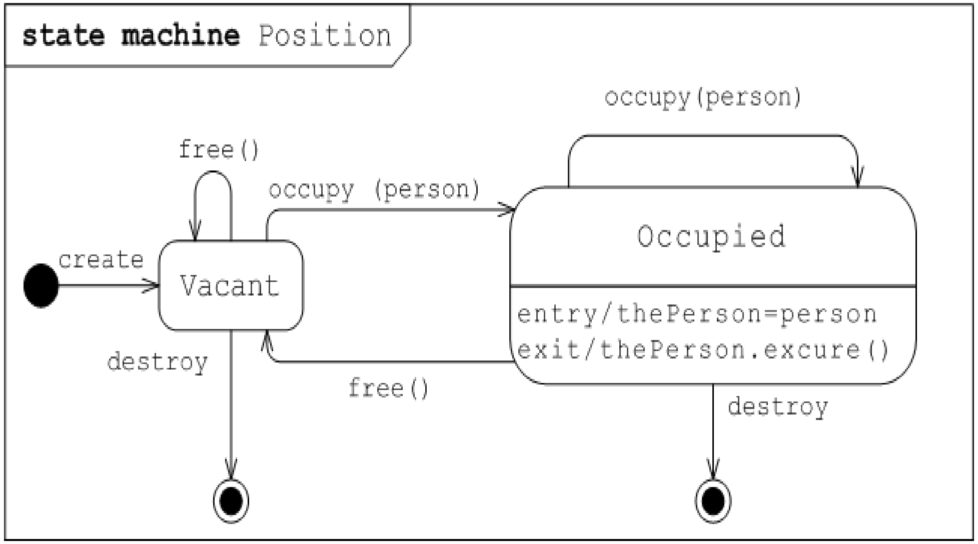
http://www.ibm.com/developerworks/rational/library/content/RationalEdge/feb04/3101_figure11.jpg

# UML state machines: example



# Multiple choice question (6)





For the Human Resources mang’t system on the left which of the following statements **is not necessarily correct**:

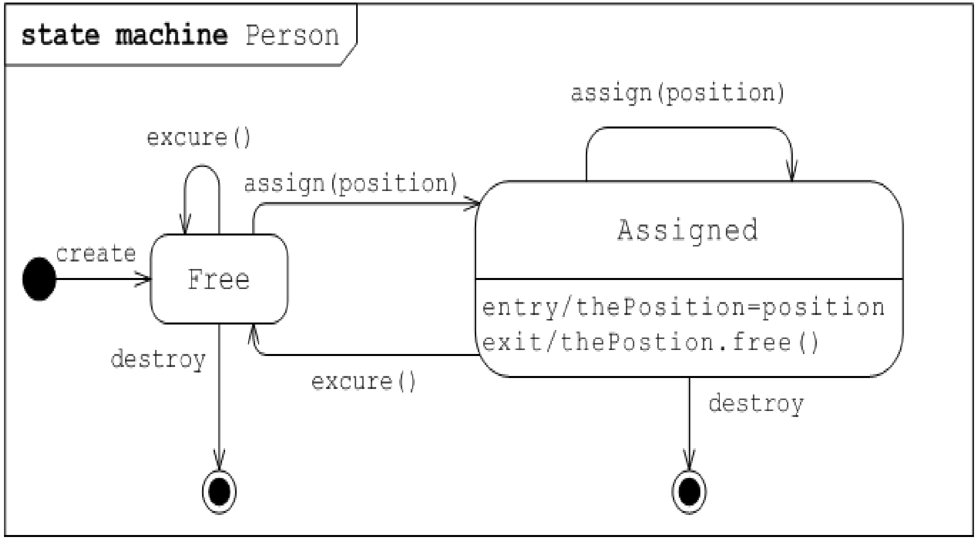
a) class Person has at least ~~4 methods~~ 2 methods in addition to evt. constructor/destructor

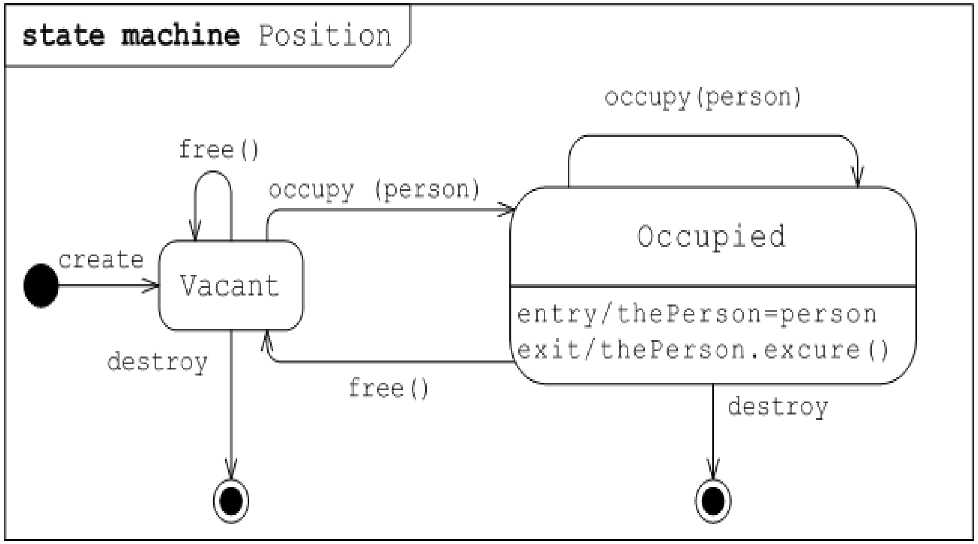
b) class Position has at most 2 fields

c) whenever person is “excured” the position is vacant

d) In the class diagram, cardinality of Person / Position is 0..1 / 0..1

# Multiple choice question (6)





For the Human Resources mang't system on the left which of the following statements **is not necessarily correct**:

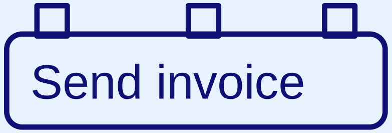
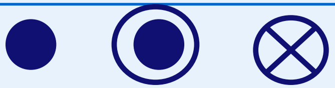
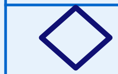
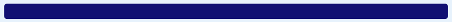
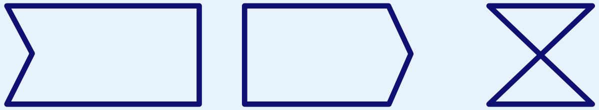
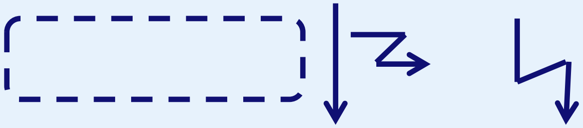
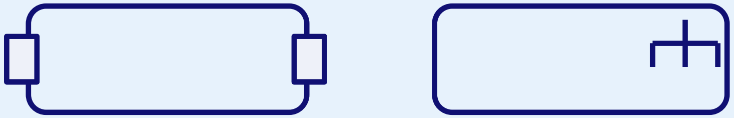
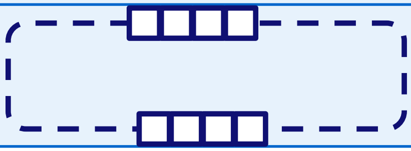
a) class Person has at least ~~4 methods~~ 2 methods in addition to evt. constructor/destructor

b) **class Position has at most 2 fields**

c) whenever person is "excured" the position is vacant

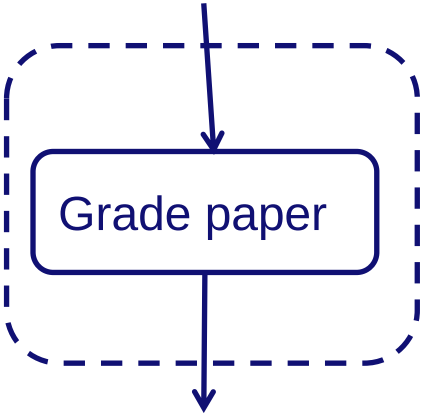
d) In the class diagram, cardinality of Person / Position is 0..1 / 0..1

# UML activity diagram: summary

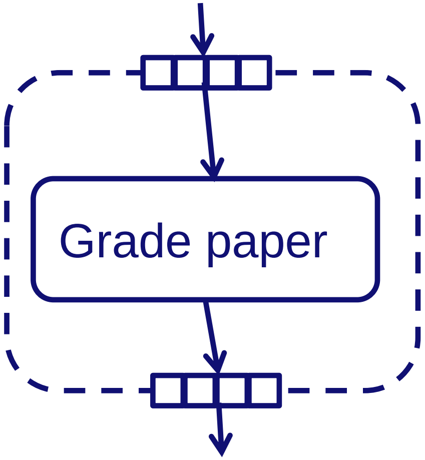
| | Graphical representation | Description |
|---|---|---|
| Action |  | action with three inputs |
| Control flow |  | start / stop markers |
| |  | decision, merge |
| |  | fork / join |
| Signals |  | incoming (accept), outgoing (send), time-based |
| Interrupts |  | interruptible activity region, interrupting edge |
| Subactivity |  | activity with input/output parameters, activity invocation |
| Collection |  | expansion region |

# Multiple choice question (7)

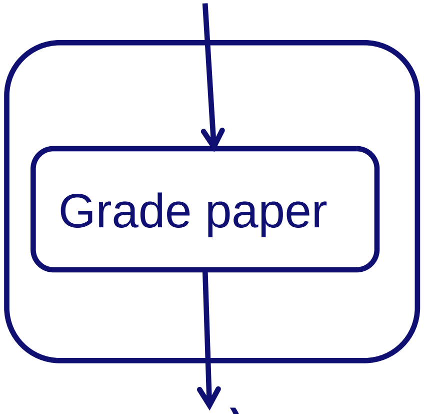
- Dr. Smith is grading students’ exam papers. How would you model her grading process?



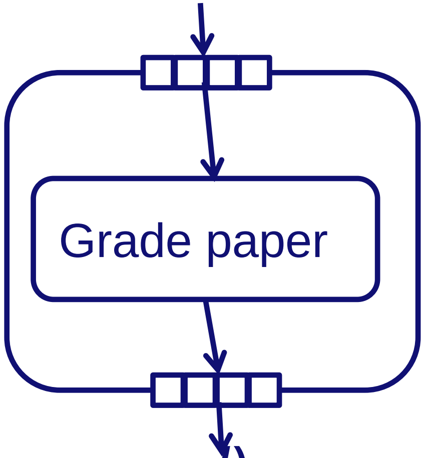
a)



b)



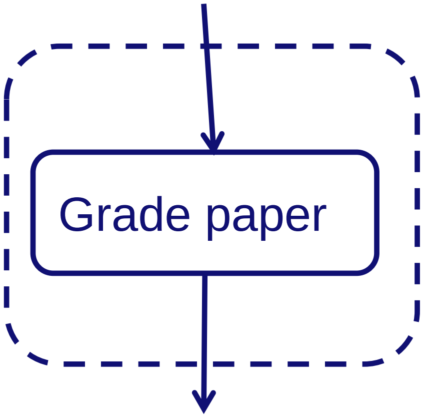
c)



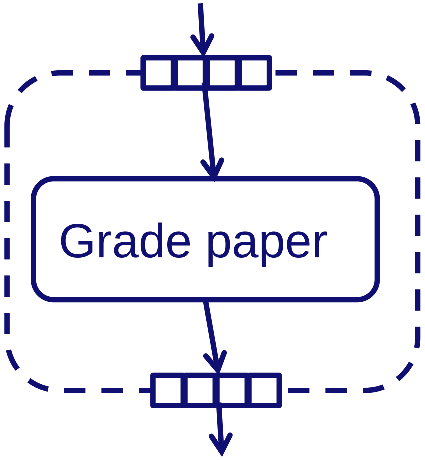
d)

# Multiple choice question (7)

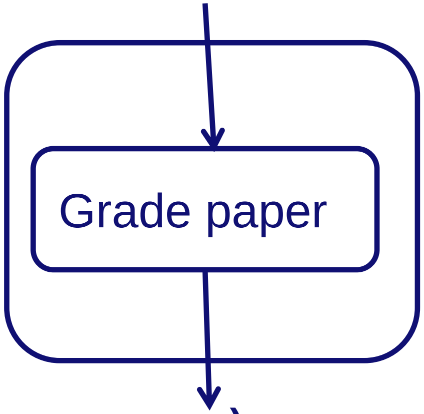
- Dr. Smith is grading students’ exam papers. How would you model her grading process?



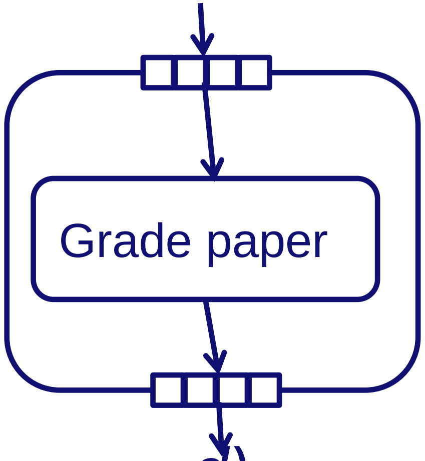
a)



**b)**

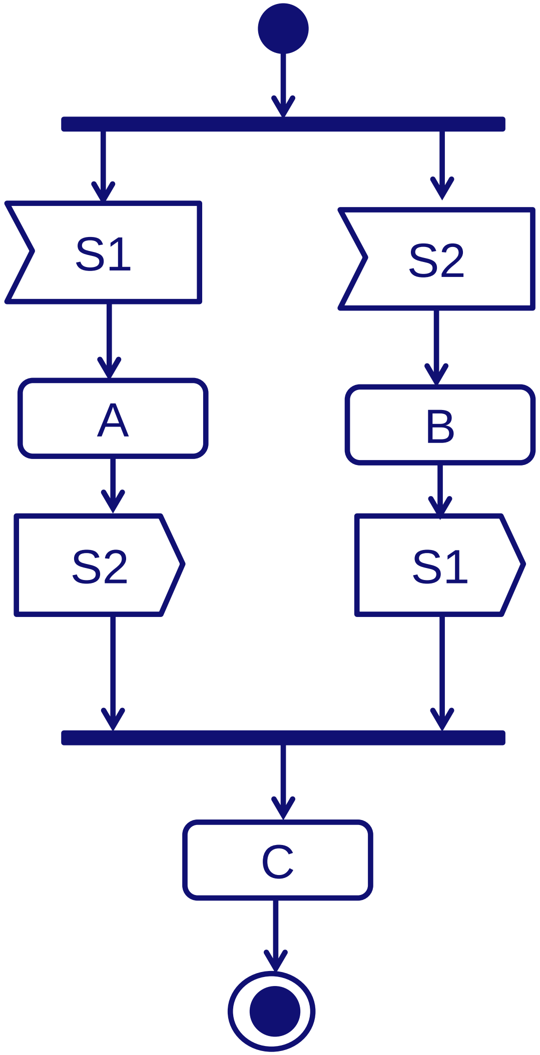


c)



d)

# Multiple choice question (8)



**The activity diagram on the left which one of the executions is possible:**

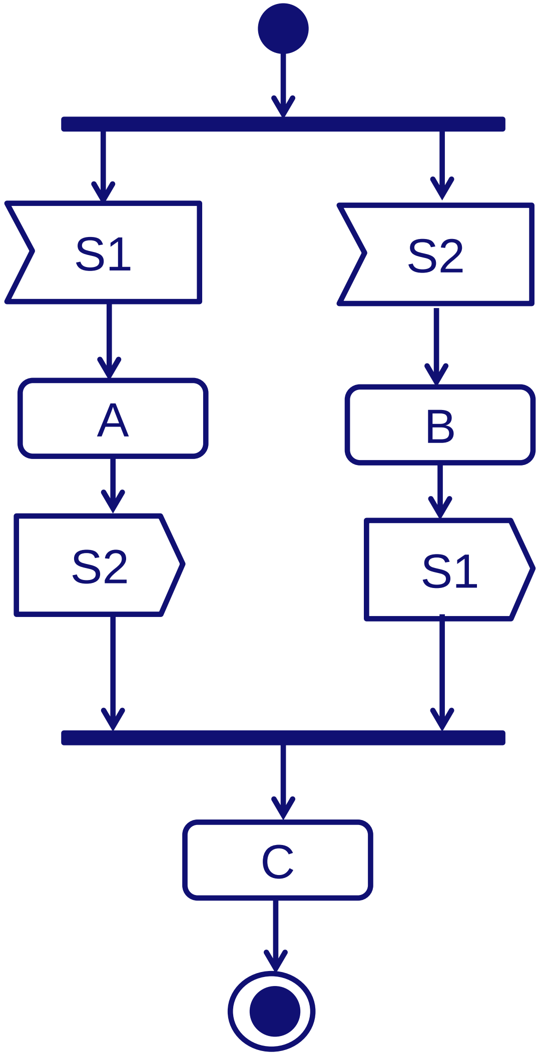
a) A followed by B followed by C

b) B followed by A followed by C

c) A followed by C

d) none of the above

# Multiple choice question (8)



**The activity diagram on the left which one of the executions is possible:**

a) A followed by B followed by C

b) B followed by A followed by C

c) A followed by C

**d) none of the above**

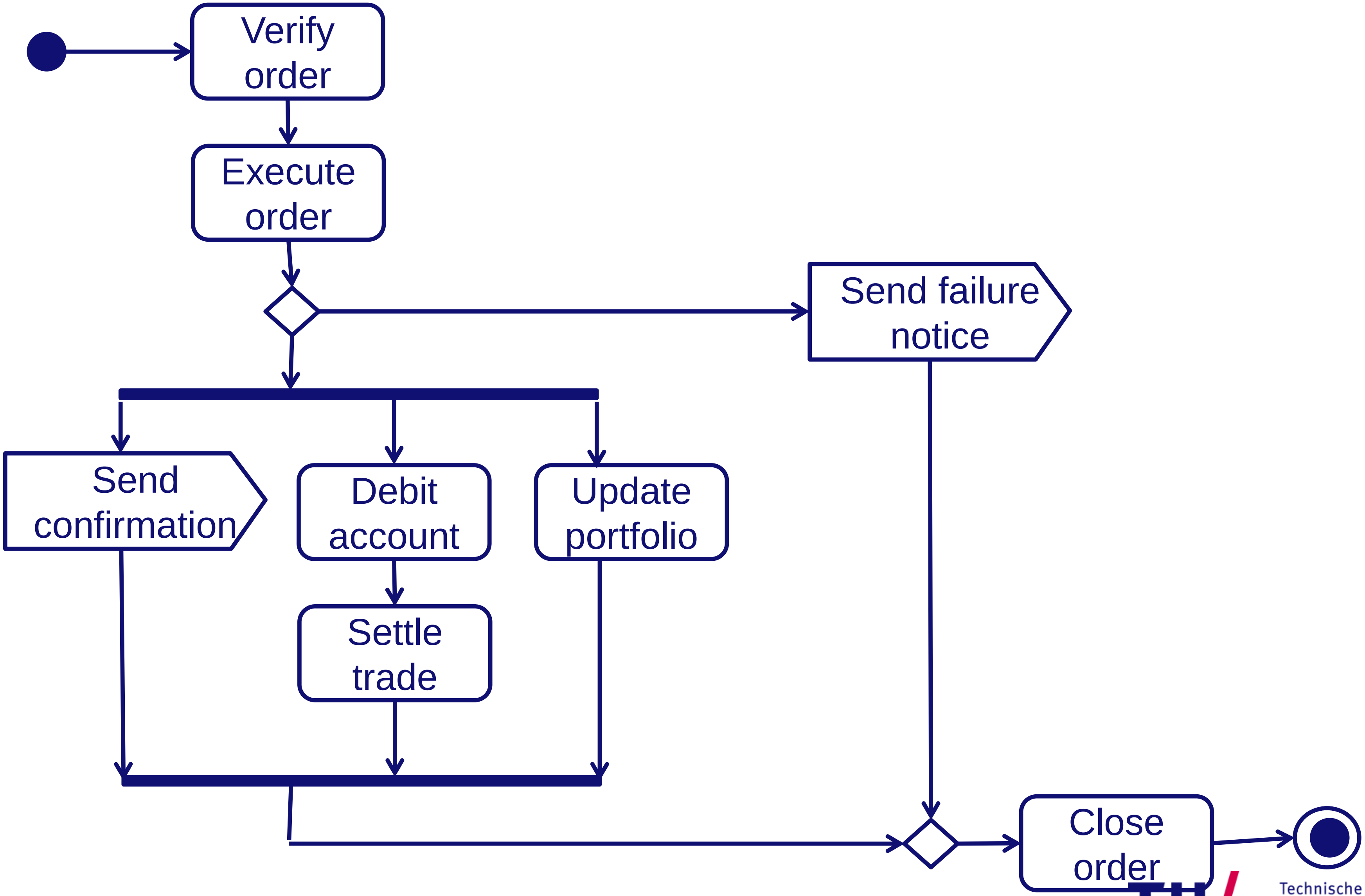
# Modeling exercise (Exercise 2.4)

The online stock broker first verifies the order against the customer’s account, then executes it with the stock exchange.

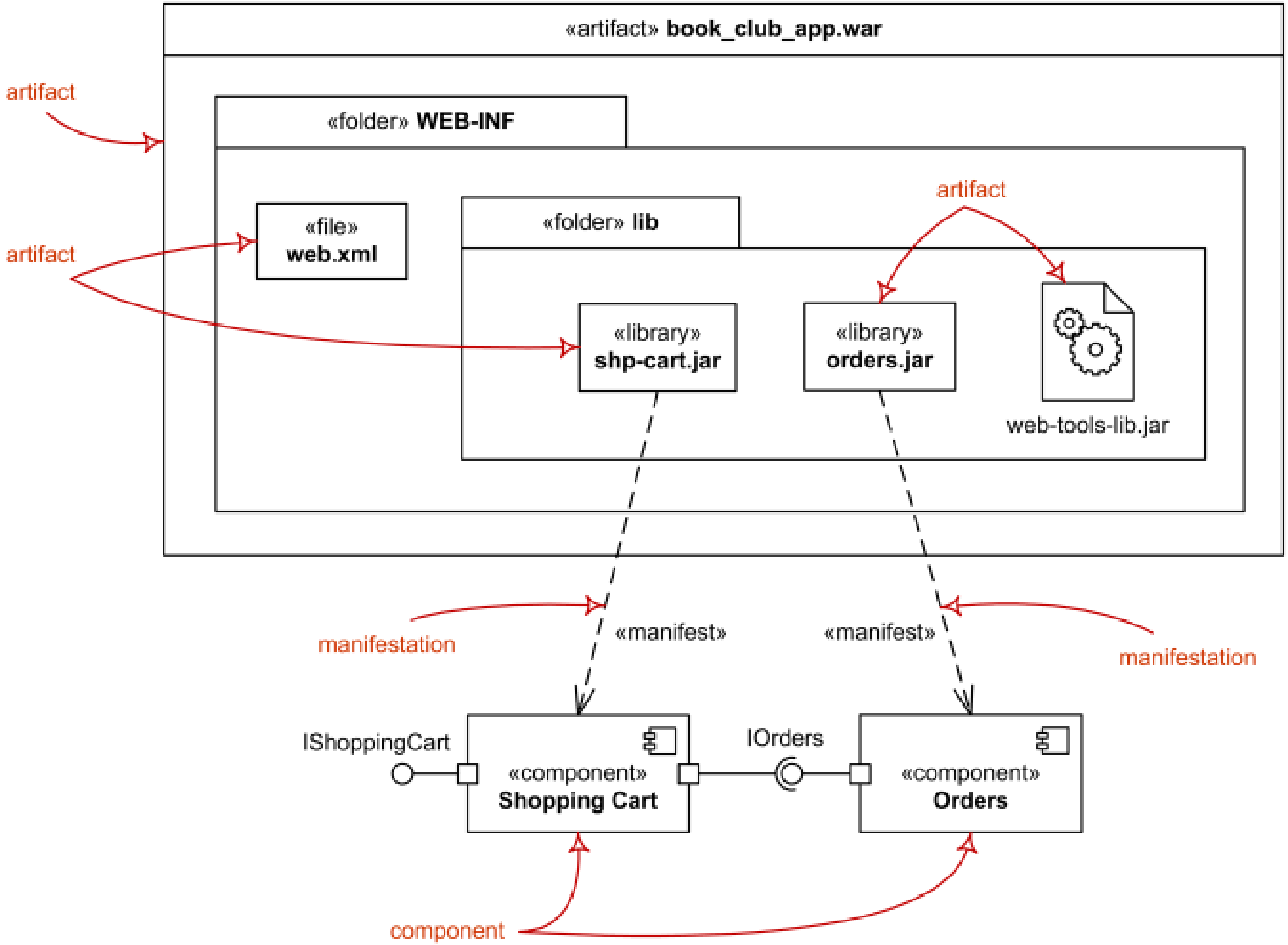
If the order executes successfully, the system does three things concurrently: mails trade confirmation to the customer, updates the online portfolio to reflect the results of the trade, and settles the trade with the other party by debiting the account and transferring cash or securities.

When all three concurrent threads have been completed, the system merges control into a single thread and closes the order.

If the order execution fails, then the system sends a failure notice to the customer and closes the order.


Verify order
Execute order
Send failure notice
Send confirmation
Debit account
Update portfolio
Settle trade
Close order

# Deployment diagram: an overview



# Multiple choice question (9)

- Dependency relation between an archive *eLib.jar* and the class diagram of *eLib* is called
  a) deployment
  b) node
  c) manifestation
  d) artifact

# Multiple choice question (9)

- Dependency relation between an archive *eLib.jar* and the class diagram of *eLib* is called

  a) deployment

  b) node

  **c) manifestation**

  d) artifact

# Architectural Styles

- **Traditional, language-influenced** styles
  - Main program and subroutines
  - Object-oriented
- **“Layered”**
  - Layered (“Virtual machines”)
  - Client-server
- **Data-flow** styles
  - Batch sequential
  - Pipe and filter
- **Shared memory**
  - Blackboard
- **Interpreter**
  - Interpreter
  - Mobile code
    - Code-on-demand
    - Remote execution/evaluation
    - Mobile agent
- **Implicit invocation**
  - Event-based
  - Publish-subscribe
- **Peer-to-peer**

*Software Architecture: Foundations, Theory, and Practice*; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with permission.

# Style Summary (1/4)

| Style Category & Name | Summary | Use It When | Avoid It When |
|---|---|---|---|
| ***Language-influenced styles*** | | | |
| Main Program and Subroutines | Main program controls program execution, calling multiple subroutines. | Application is small and simple. | Complex data structures needed.<br>Future modifications likely. |
| Object-oriented | Objects encapsulate state and accessing functions | Close mapping between external entities and internal objects is sensible.<br>Many complex and interrelated data structures. | Application is distributed in a heterogeneous network.<br>Strong independence between components necessary.<br>High performance required. |
| ***Layered*** | | | |
| Virtual Machines | Virtual machine, or a layer, offers services to layers above it | Many applications can be based upon a single, common layer of services.<br>Interface service specification resilient when implementation of a layer must change. | Many levels are required (causes inefficiency).<br>Data structures must be accessed from multiple layers. |
| Client-server | Clients request service from a server | Centralization of computation and data at a single location (the server) promotes manageability and scalability; end-user processing limited to data entry and presentation. | Centrality presents a single-point-of-failure risk;  Network bandwidth limited; Client machine capabilities rival or exceed the server’s. |

*Software Architecture: Foundations, Theory, and Practice*; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with permission.

# Style Summary, continued (2/4)

| | | | |
|---|---|---|---|
| ***Data-flow styles*** | | | |
| Batch sequential | Separate programs executed sequentially, with batched input | Problem easily formulated as a set of sequential, severable steps. | Interactivity or concurrency between components necessary or desirable.<br>Random-access to data required. |
| Pipe-and-filter | Separate programs, a.k.a. filters, executed, potentially concurrently. Pipes route data streams between filters | [As with batch-sequential] Filters are useful in more than one application. Data structures easily serializable. | Interaction between components required. Exchange of complex data structures between components required. |
| ***Shared memory*** | | | |
| Blackboard | Independent programs, access and communicate exclusively through a global repository known as blackboard | All calculation centers on a common, changing data structure;<br>Order of processing dynamically determined and data-driven. | Programs deal with independent parts of the common data. Interface to common data susceptible to change. When interactions between the independent programs require complex regulation. |

# Style Summary, continued (3/4)

| | | | |
|---|---|---|---|
| ***Interpreter*** | | | |
| Interpreter | Interpreter parses and executes the input stream, updating the state maintained by the interpreter | Highly dynamic behavior required. High degree of end-user customizability. | High performance required. |
| Mobile Code | Code is mobile, that is, it is executed in a remote host | When it is more efficient to move processing to a data set than the data set to processing.<br>When it is desirous to dynamically customize a local processing node through inclusion of external code | Security of mobile code cannot be assured, or sandboxed.<br>When tight control of versions of deployed software is required. |

# Style Summary, continued (4/4)

| | | | |
|---|---|---|---|
| ***Implicit Invocation*** | | | |
| Publish-subscribe | Publishers broadcast messages to subscribers | Components are very loosely coupled. Subscription data is small and efficiently transported. | When middleware to support high-volume data is unavailable. |
| Event-based | Independent components asynchronously emit and receive events communicated over event buses | Components are concurrent and independent.<br>Components heterogeneous and network-distributed. | Guarantees on real-time processing of events is required. |
| ***Peer-to-peer*** | Peers hold state and behavior and can act as both clients and servers | Peers are distributed in a network, can be heterogeneous, and mutually independent.<br>Robust in face of independent failures.<br>Highly scalable. | Trustworthiness of independent peers cannot be assured or managed.<br>Resource discovery inefficient without designated nodes. |

# Multiple choice question (10)

http://kevinnam.com/?page_id=175

Read the following text and replace **XXX** with one of the answers:

Virtual communities, like any community of people, often have trouble regulating their participants. This project's approach includes the design and implementation of an agent-based system written in Java that helps facilitate and regulate online social spaces appropriately and also helps maintain a pleasurable environment for users. The system comprises of a number of agents that work collaboratively  through a **XXX** architecture. Each agent looks for a specific problem,  and new agents can be built and added to the system as required. This  system as a whole is not designed to completely eliminate the need for human regulator, but rather to help reduce human intervention in regulating online communities.

a) publish-subscribe
b) client-server
c) pipe-and-filter
d) blackboard

# Multiple choice question (10)

http://kevinnam.com/?page_id=175

Read the following text and replace **XXX** with one of the answers:

Virtual communities, like any community of people, often have trouble regulating their participants. This project's approach includes the design and implementation of an agent-based system written in Java that helps facilitate and regulate online social spaces appropriately and also helps maintain a pleasurable environment for users. The system comprises of a number of agents that work collaboratively through a **XXX** architecture. Each agent looks for a specific problem, and new agents can be built and added to the system as required. This system as a whole is not designed to completely eliminate the need for human regulator, but rather to help reduce human intervention in regulating online communities.

a) publish-subscribe
b) client-server
c) pipe-and-filter
**d) blackboard**

# Multiple-choice question (11)

http://msdn.microsoft.com/en-us/library/ff647328.aspx

Consider a financial application that integrates trading tools, portfolio management applications, modeling and risk analysis tools, trend indicators, and tickers. Market activity causes interaction between these systems. For example, a trading system communicates the completion of a sell transaction by sending a message to all other trading applications. <…> Managing the addition or removal of trading applications should not interfere with processing trades.

Which architectural style would you apply to implement this application?

a) code-on-demand
b) event-based
c) batch-sequential
d) peer-to-peer

# Multiple-choice question (11)

http://msdn.microsoft.com/en-us/library/ff647328.aspx

Consider a financial application that integrates trading tools, portfolio management applications, modeling and risk analysis tools, trend indicators, and tickers. Market activity causes interaction between these systems. For example, a trading system communicates the completion of a sell transaction by sending a message to all other trading applications. <…> Managing the addition or removal of trading applications should not interfere with processing trades.

Which architectural style would you apply to implement this application?

a) code-on-demand
b) **event-based**
c) batch-sequential
d) peer-to-peer

# Exercise (6.3)

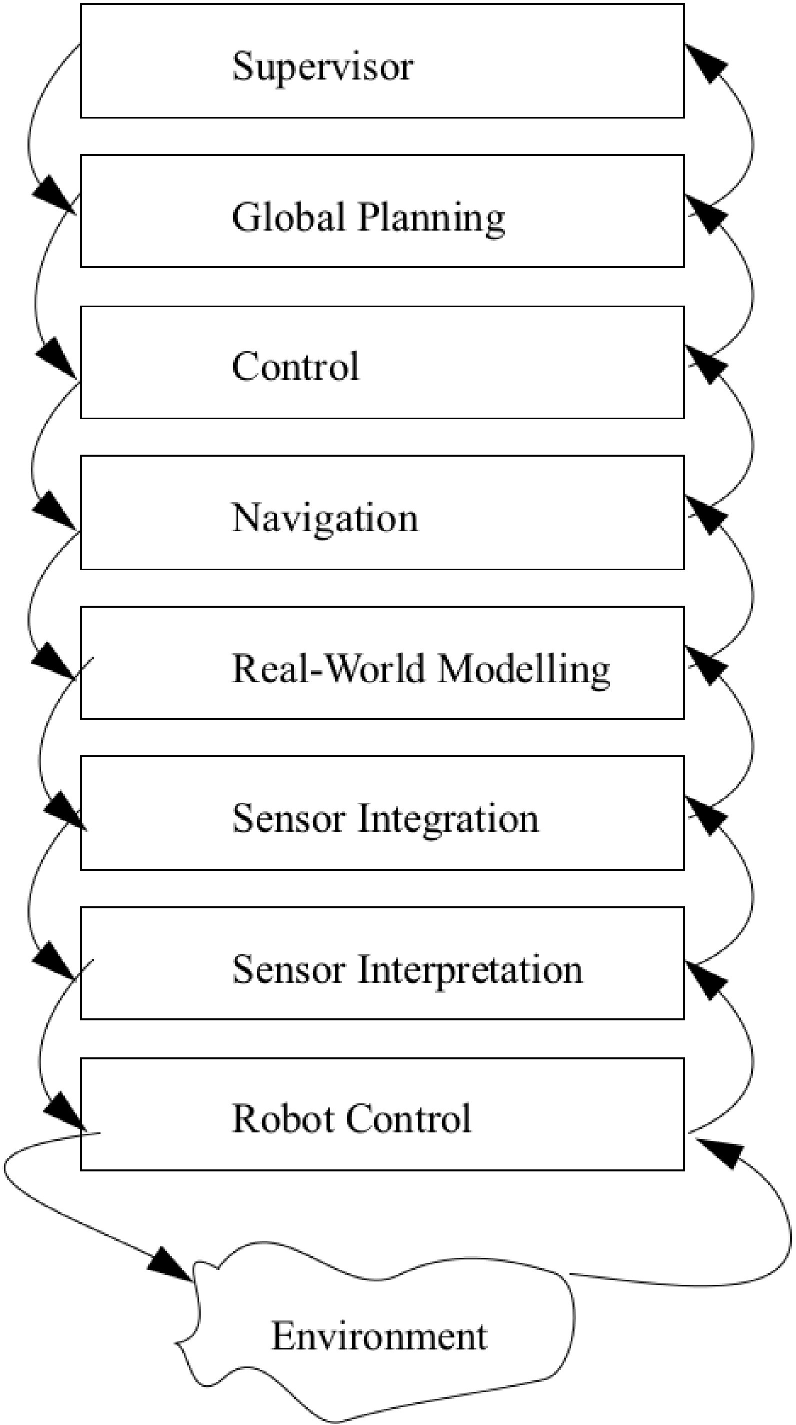
A mobile robot has to acquire the input provided by its sensors, control the motion of its wheels and plan its future path. Beware:

- Obstacles may block the robot's path.
- The sensor input may be imperfect.
- The robot may run out of power.
- Mechanical limitations may restrict the accuracy with which the robot moves.
- The robot may manipulate hazardous materials.
- Unpredictable events may leave little time for responding.

Design an architecture for the mobile robot applying:

- (Strict) layered style.
- Blackboard style.

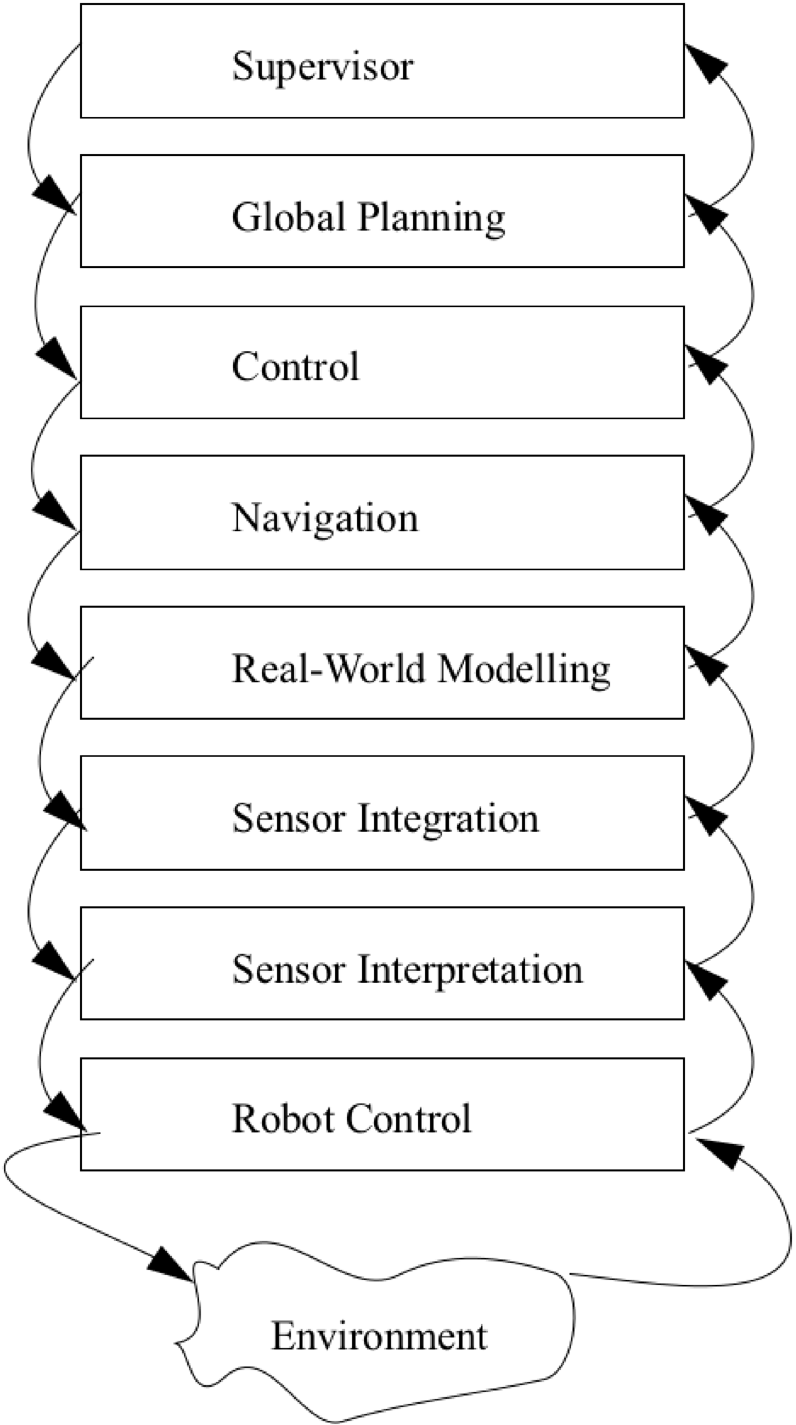
# Strict layered style



# Strict layered style

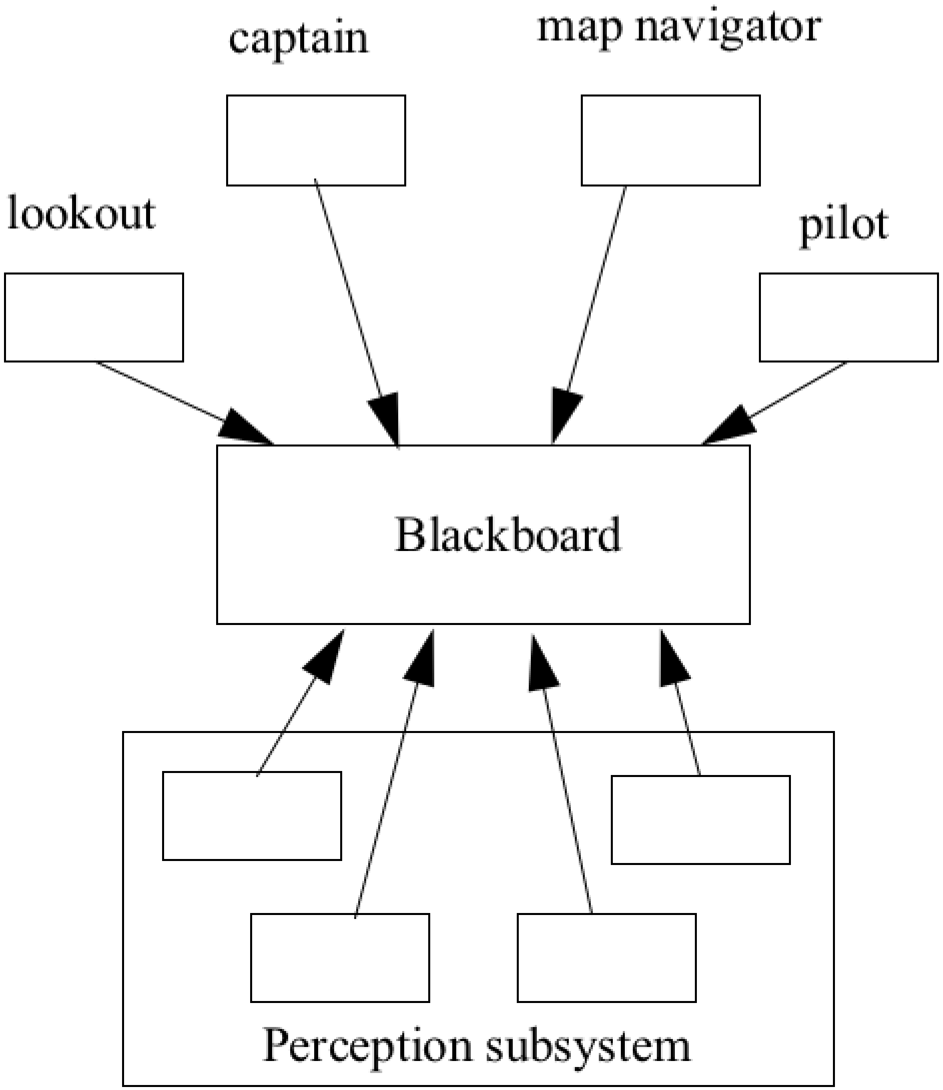
- In reality the information exchange is less straightforward
  - Levels may be skipped because of the timing constraints.

- Advantage:
  - abstraction layers stress roles and organization of components
- Disadvantage:
  - unrealistic communication patterns



# Blackboard style

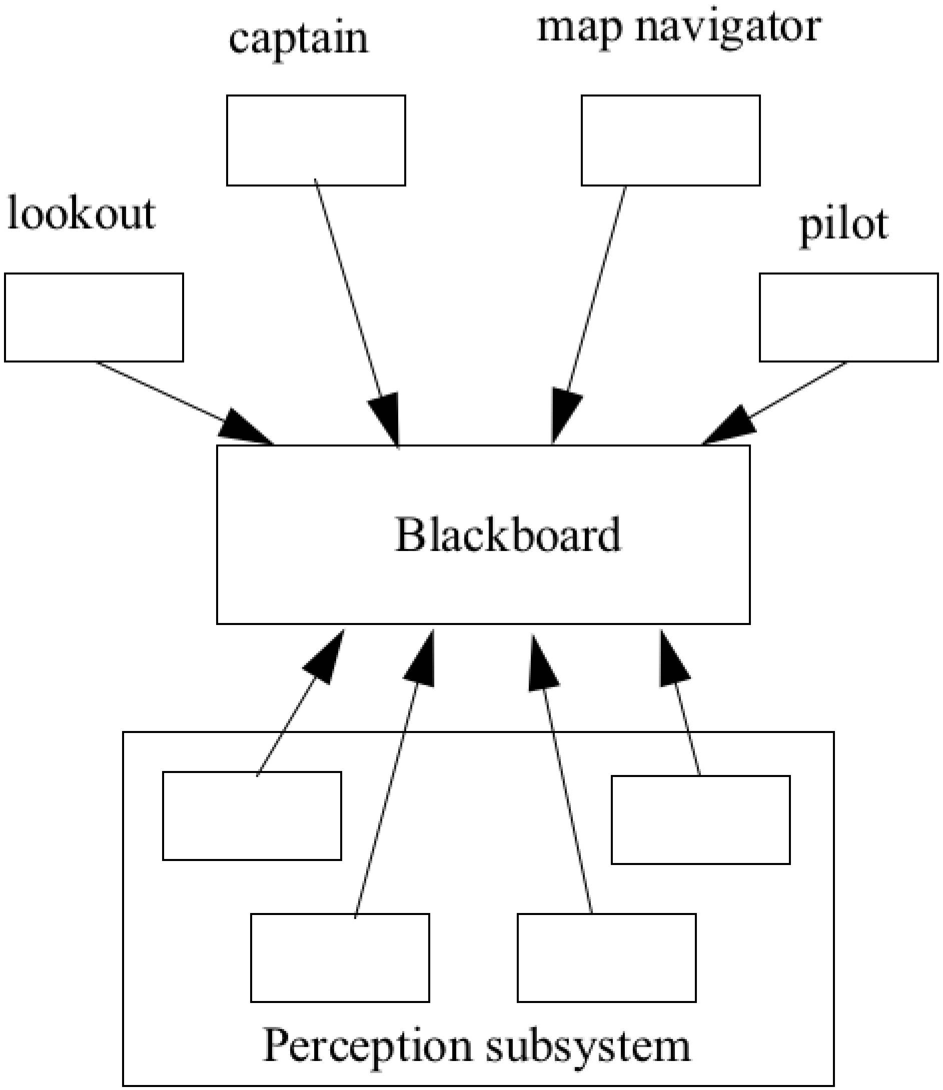
- **Captain**: overall supervisor.
- **Map navigator**: high level path planner.
- **Lookout**: monitors the environment for landmarks.
- **Pilot**: low level path planner and motor controller.
- **Perception subsystem**: modules that accept the raw input from multiple sensors and integrate it into a coherent interpretation.



# Blackboard style

- Advantages
  - Support for collaboration, dealing with uncertainty, flexible

- Disadvantages (challenges)
  - Performance, fault tolerance, safety



# Viewpoints and views

- **Viewpoint** a way of looking at a system from the position of a certain *stakeholder* with a particular *concern*
  - defines creation, depiction and analysis of a view
  - language, used models, notation, methods, analysis techniques

- **View**: whatever you see in a system from a particular viewpoint
  - collection of system models
  - conforming to the viewpoint

# Multiple choice question (12)

Which of the following statements from the following list are not correct?

a) using viewpoints the architects can separate out different kinds of information

b) the need to maintain multiple views describing a software system as well as consistency between those views hinders maintenance of the architecture documentation

c) pipe-and-filter architectural style can be considered as a structural viewpoint

d) according to IEEE Std. 42010-2011 viewpoint can exist without stakeholders.

# Multiple choice question (12)

Which of the following statements from the following list are not correct?

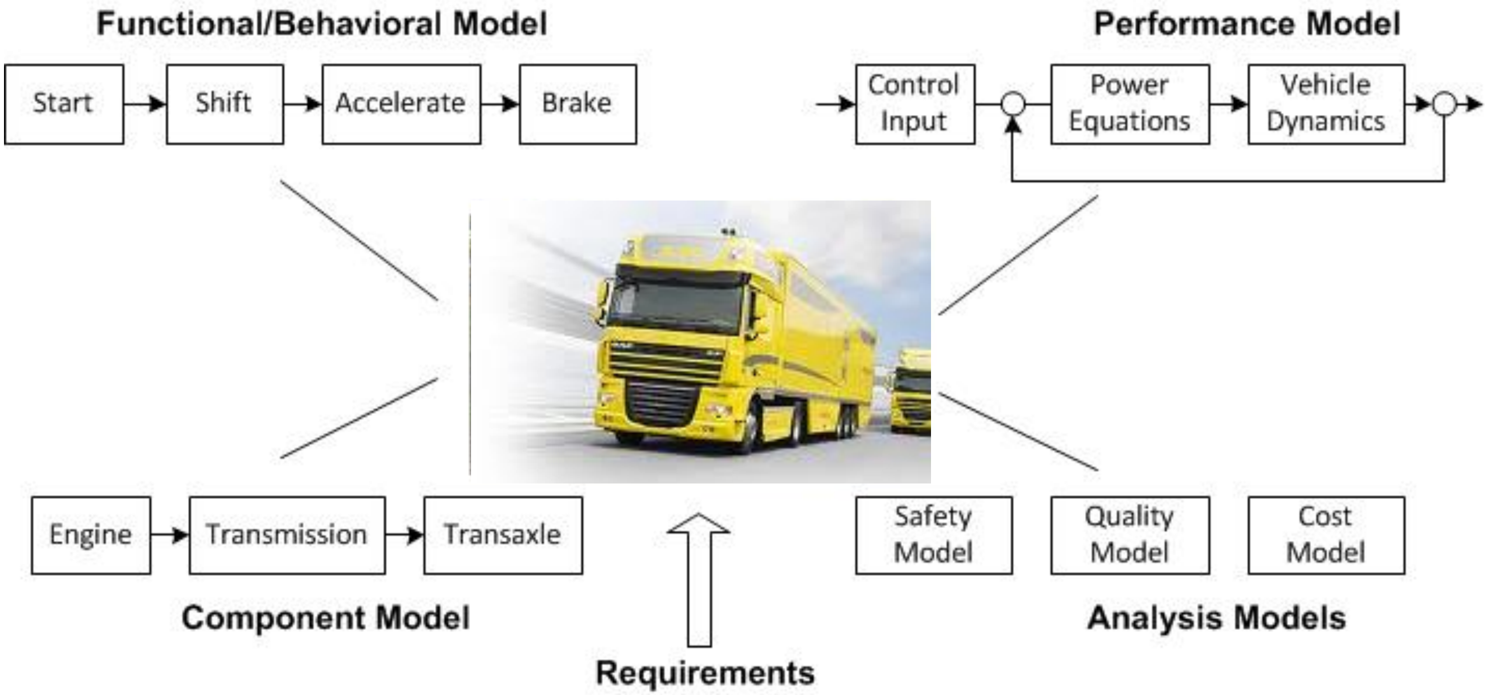
a) using viewpoints the architects can separate out different kinds of information

b) the need to maintain multiple views describing a software system as well as consistency between those views hinders maintenance of the architecture documentation

c) pipe-and-filter architectural style can be considered as a structural viewpoint

**d) according to IEEE Std. 42010-2011 viewpoint can exist without stakeholders.**

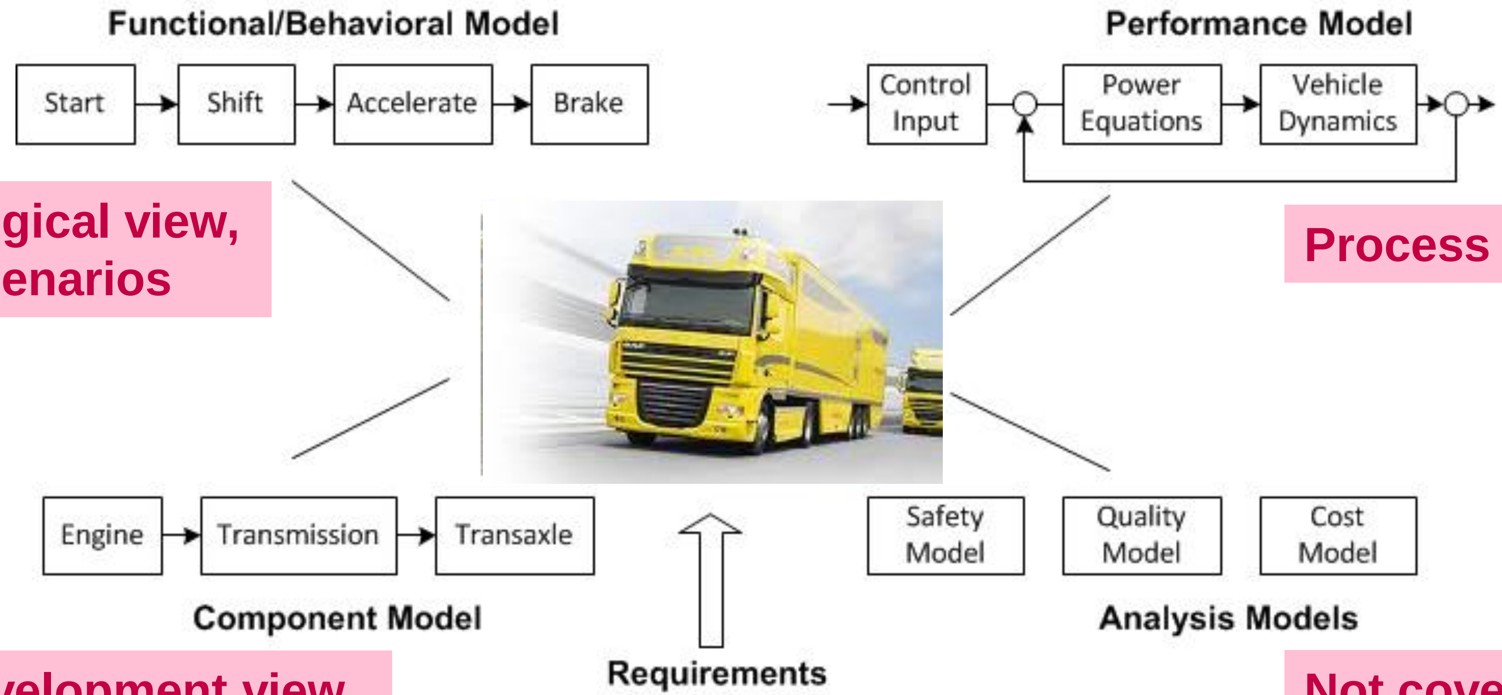
# Question



Another take on *viewpoints* (here called *model(s)*)

Can you map those models to Kruchten's 4+1?

# Question



Functional/Behavioral Model

Start → Shift → Accelerate → Brake

Performance Model

Control Input → Power Equations → Vehicle Dynamics

Engine → Transmission → Transaxle

Component Model

Requirements

Safety Model

Quality Model

Cost Model

Analysis Models

**Logical view, Scenarios**

**Process view**

**Development view, Deployment view**

**Not covered**

Another take on *viewpoints* (here called *model(s)*)

Can you map those models to Kruchten’s 4+1?

# Multiple choice question (13)

Rozanski and Woods have identified a concurrency viewpoint:

> describes the concurrency structure of the system and maps functional elements to concurrency units to clearly identify the parts of the system that can execute concurrently and how this is coordinated and controlled. This entails the creation of models that show the process and thread structures that the system will use and the interprocess communication mechanisms used to coordinate their operation.

Which one of the Kruchten’s 4+1 views addresses this concern

a) Process view

b) Logical view

c) Scenarios

d) None of the above

# Multiple choice question (13)

Rozanski and Woods have identified a concurrency viewpoint:

> describes the concurrency structure of the system and maps functional elements to concurrency units to clearly identify the parts of the system that can execute concurrently and how this is coordinated and controlled. This entails the creation of models that show the process and thread structures that the system will use and the interprocess communication mechanisms used to coordinate their operation.

Which one of the Kruchten's 4+1 views addresses this concern

a) **Process view**

b) Logical view

c) Scenarios

d) None of the above

# Exam

- **April 15, 9:00 – 12:00**
- **Location: check OASE**

- **Part I. 20 multiple choice questions (4 options)**
  - 2, if Indicated = Correct
  - 1, if Indicated $\subseteq$ Correct
  - 0, otherwise

**Questions like the ones we have seen today**

- **Part II. Five modeling exercises (15 points each)**
  - Choose four
  - We’ll grade the first four

**Exercises like the ones you have seen during the instruction sessions + today**

# Conclusions

**Success on the final exam!**

**Alexander, Anton, Kees, Sarmen, Ulyana, Yanja**