<u>contract AssEmp1 - addShiftsRequierments</u>

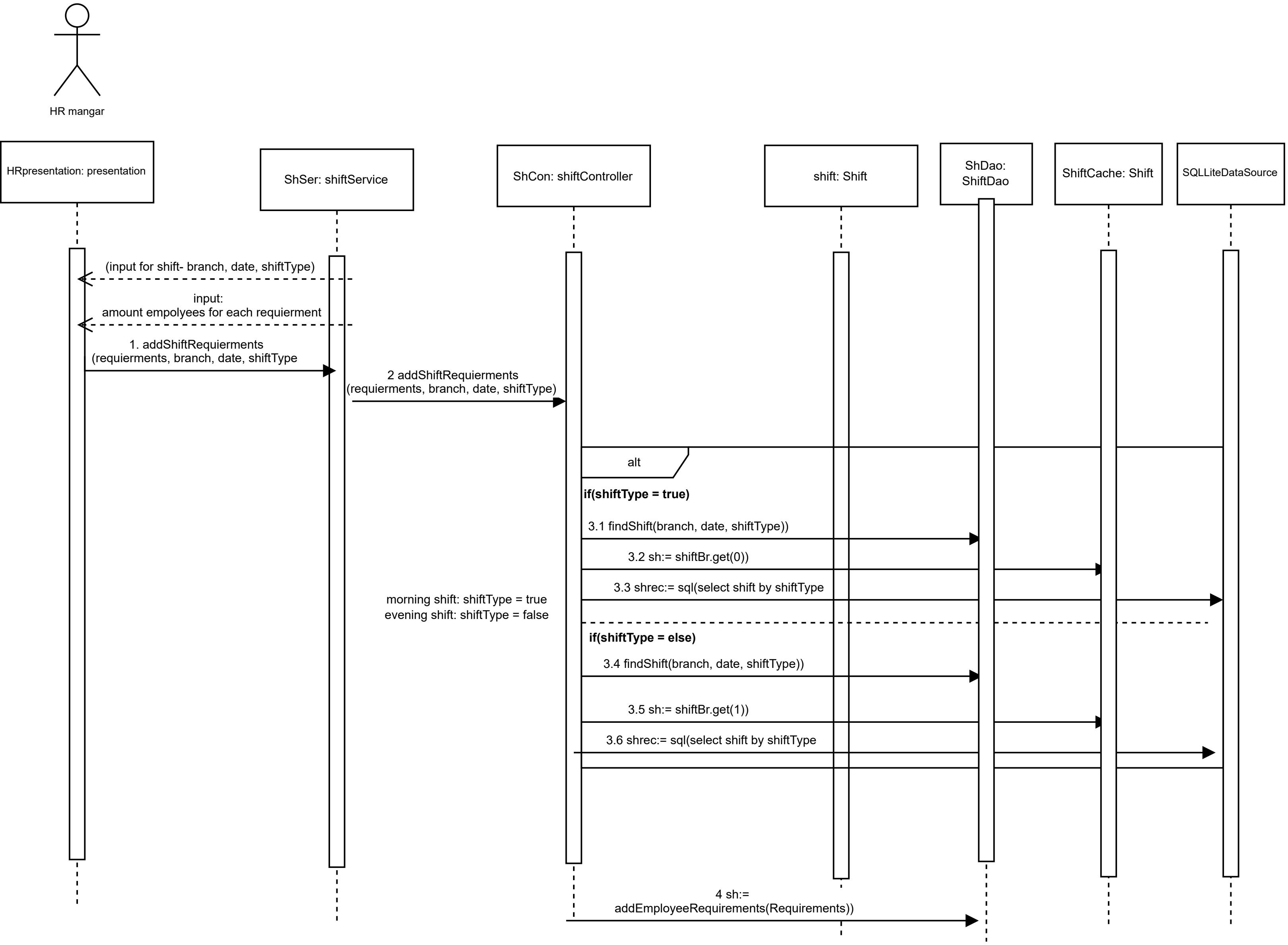
References: Use Cases: Assign employee to a shift

pre-condition:

1. There is exist shift in this date and shiftType (shift != null)
2. The number of employees needed for each position is >= 0.

post-condition:

1. If the number of employees for a such position before the the addition was x and y employees were added, then now there should be a total demand for x+y employees for the position


HR mangar
HRpresentation: presentation
ShSer: shiftService
ShCon: shiftController
shift: Shift
ShDao: ShiftDao
ShiftCache: Shift
SQLLiteDataSource
(input for shift- branch, date, shiftType)
input:
amount empolyees for each requierment
1. addShiftRequierments
(requierments, branch, date, shiftType
2 addShiftRequierments
(requierments, branch, date, shiftType)
alt
if(shiftType = true)
3.1 findShift(branch, date, shiftType))
3.2 sh:= shiftBr.get(0))
3.3 shrec:= sql(select shift by shiftType
morning shift: shiftType = true
evening shift: shiftType = false
if(shiftType = else)
3.4 findShift(branch, date, shiftType))
3.5 sh:= shiftBr.get(1))
3.6 shrec:= sql(select shift by shiftType
4 sh:=
addEmployeeRequirements(Requirements))

<u>contract AssEmp2 - addSubmittion(date, shifyType)</u>

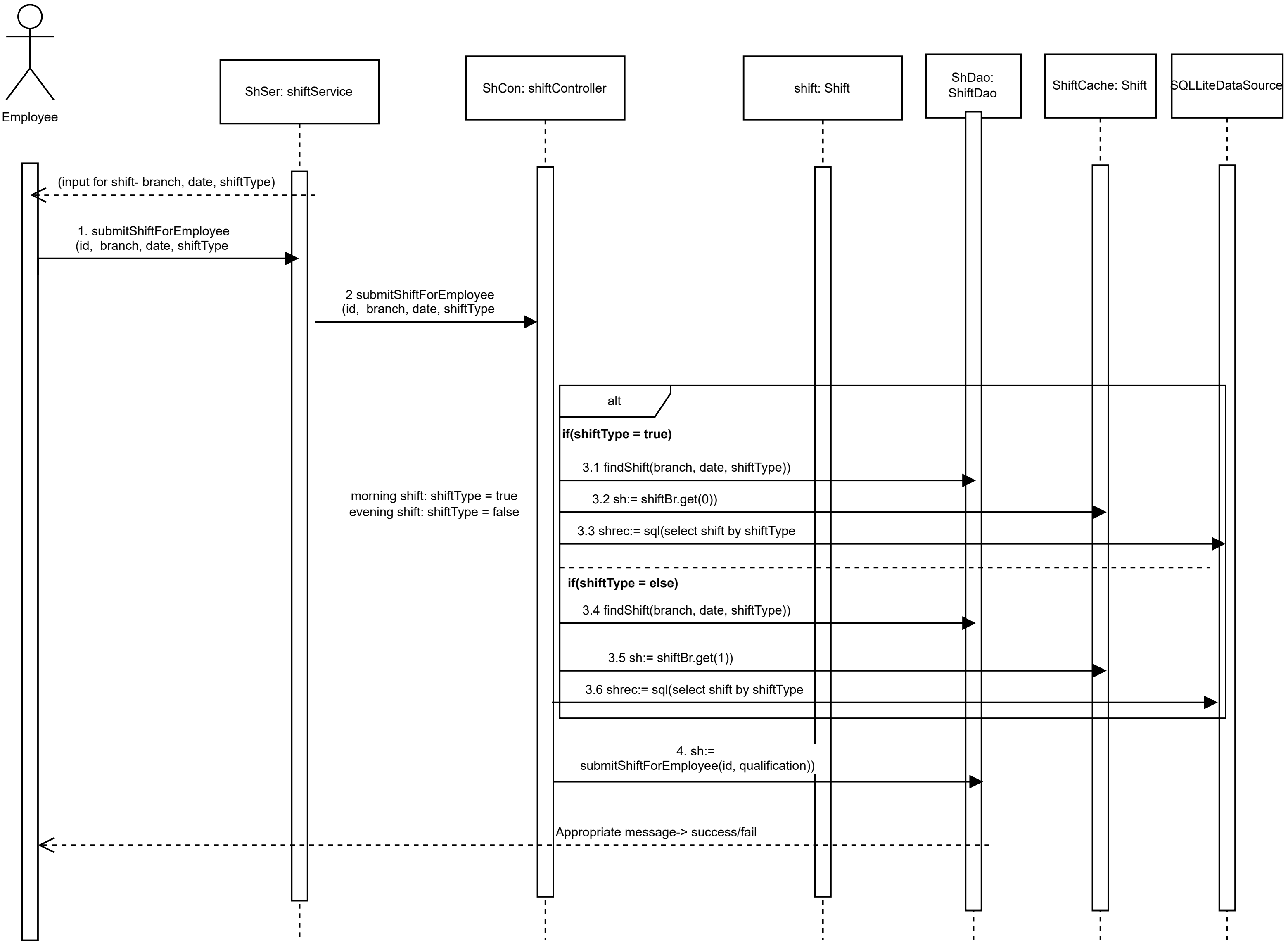
References: Use Cases: Assign employee to a shift

pre-condition:

1. The employee that add the submition is exist in the system- other
2. There is exist shift in this date and shiftType (shift != null)
3. There is at least one position that the employee is qualified to work in.
3. The employee has not yet submitted constraints for the shift - otherwise nothing will change

post-condition:

1. The system will submit a shift constraint for each position which the employee is qualified


Employee
ShSer: shiftService
ShCon: shiftController
shift: Shift
ShDao: ShiftDao
ShiftCache: Shift
SQLLiteDataSource
(input for shift- branch, date, shiftType)
1. submitShiftForEmployee (id, branch, date, shiftType
2 submitShiftForEmployee (id, branch, date, shiftType
alt
if(shiftType = true)
3.1 findShift(branch, date, shiftType))
3.2 sh:= shiftBr.get(0))
3.3 shrec:= sql(select shift by shiftType
morning shift: shiftType = true
evening shift: shiftType = false
if(shiftType = else)
3.4 findShift(branch, date, shiftType))
3.5 sh:= shiftBr.get(1))
3.6 shrec:= sql(select shift by shiftType
4. sh:= submitShiftForEmployee(id, qualification))
Appropriate message-> success/fail

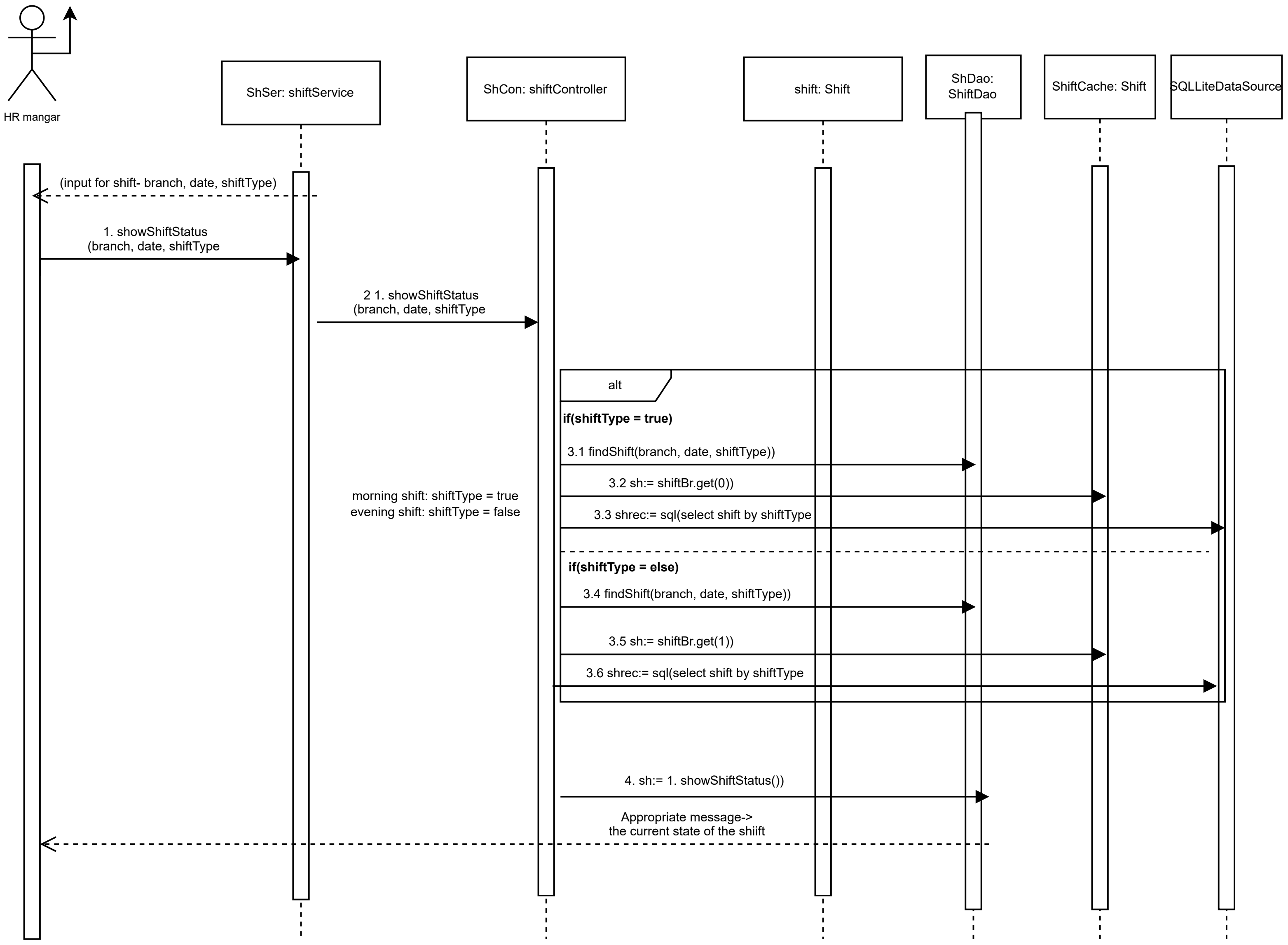
contract AssEmp3- showShiftStaus

References: Use Cases: Assign employee to a shift

pre-condition:

1. There is exist shift in this date and shiftType (shift != null)

post-condition: null


HR mangar
ShSer: shiftService
ShCon: shiftController
shift: Shift
ShDao: ShiftDao
ShiftCache: Shift
SQLLiteDataSource
(input for shift- branch, date, shiftType)
1. showShiftStatus (branch, date, shiftType
2 1. showShiftStatus (branch, date, shiftType
alt
if(shiftType = true)
3.1 findShift(branch, date, shiftType))
3.2 sh:= shiftBr.get(0))
morning shift: shiftType = true
evening shift: shiftType = false
3.3 shrec:= sql(select shift by shiftType
if(shiftType = else)
3.4 findShift(branch, date, shiftType))
3.5 sh:= shiftBr.get(1))
3.6 shrec:= sql(select shift by shiftType
4. sh:= 1. showShiftStatus())
Appropriate message-> the current state of the shiift

<u>contract AssEmp4 - assignAll/assignManually</u>

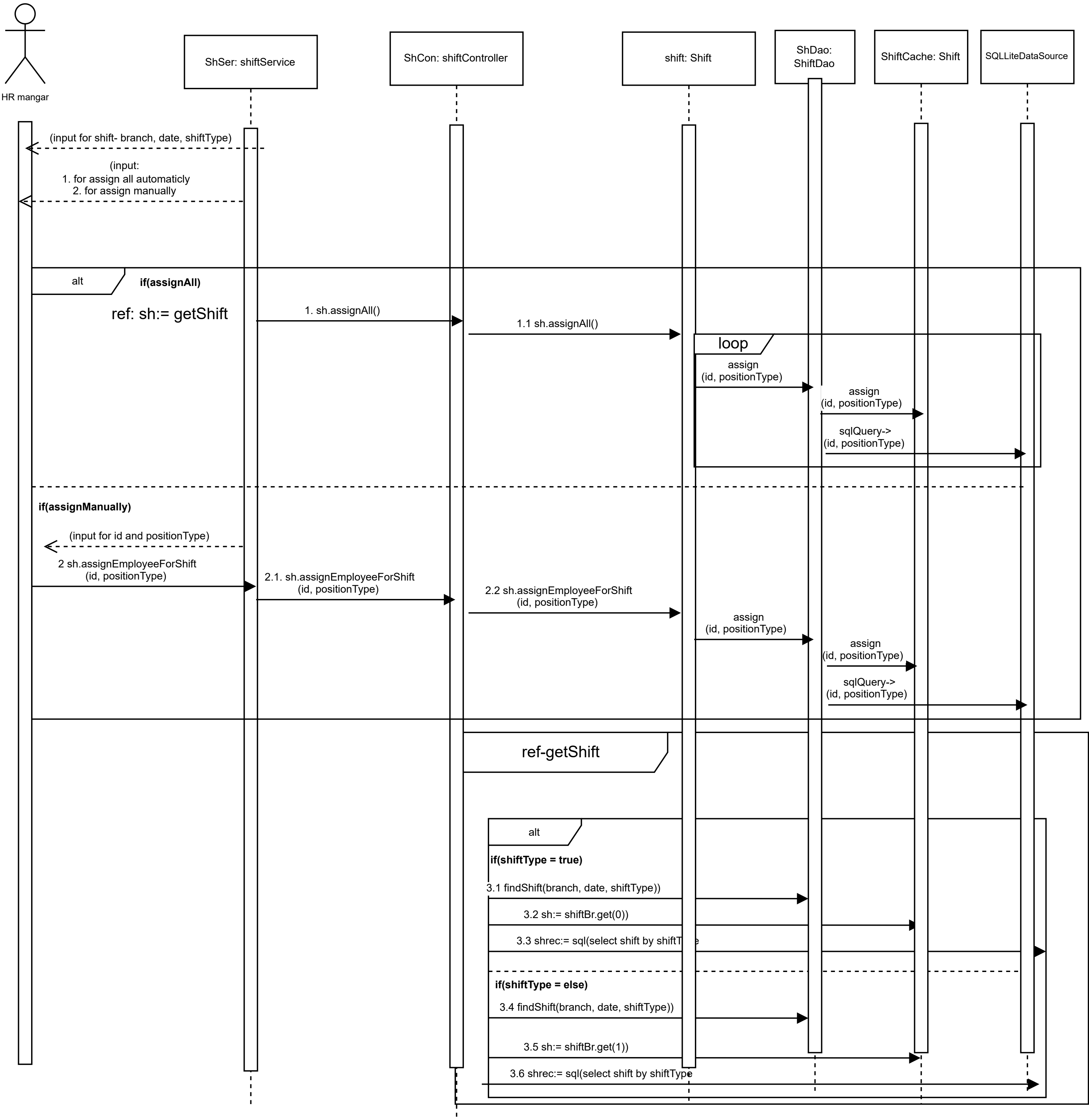
References: Use Cases: Assign employee to a shift

pre-condition:

1. There is exist shift in this date and shiftType (shift != null)
2. There are employees in the system who submitted a constraint for the shift
3. The employees who submitted a constraint for any role are qualified to perform that role

post-condition:

1. Each employee is assigned to a maximum of one shift for that date.
2. Each employee is assigned a maximum of 6 shifts that week.



morning shift: shiftType = true
evening shift: shiftType = false

Contract ED01 - Execute Delivery

Operation: Cross executeDelivery(delivery:Delivery)

References: use cases: Execute delivery

PreConditions: -There is a suitable and available driver assigned to delivery
-There is a storeKeeper assigned to both shifts in each branch where the transport arrives today

PostConditions: -The delivery executed
-Products that were not collected from suppliers due to overweight in the delivery were assigned to another delivery in the future.

Contract ED02 - Overweight Action

Operation: Cross OverWeightAction(action)

References: use cases: Execute delivery

PreConditions: -The delivery has an overweight after loading product from supplier
-The transport manager selected an action to perform

PostConditions: -The overweight handled according to the action selected, the delivery can continue according to the current state after the action

Contract ED03 - Enter weight

Operation: Cross enterWeightFunction(supplierAddress,deliveryId)

References: use cases: Execute delivery

PreConditions: -The delivery has arrived to supplier and loaded products

PostConditions: -The transport manager will enter the weight of the new loaded products

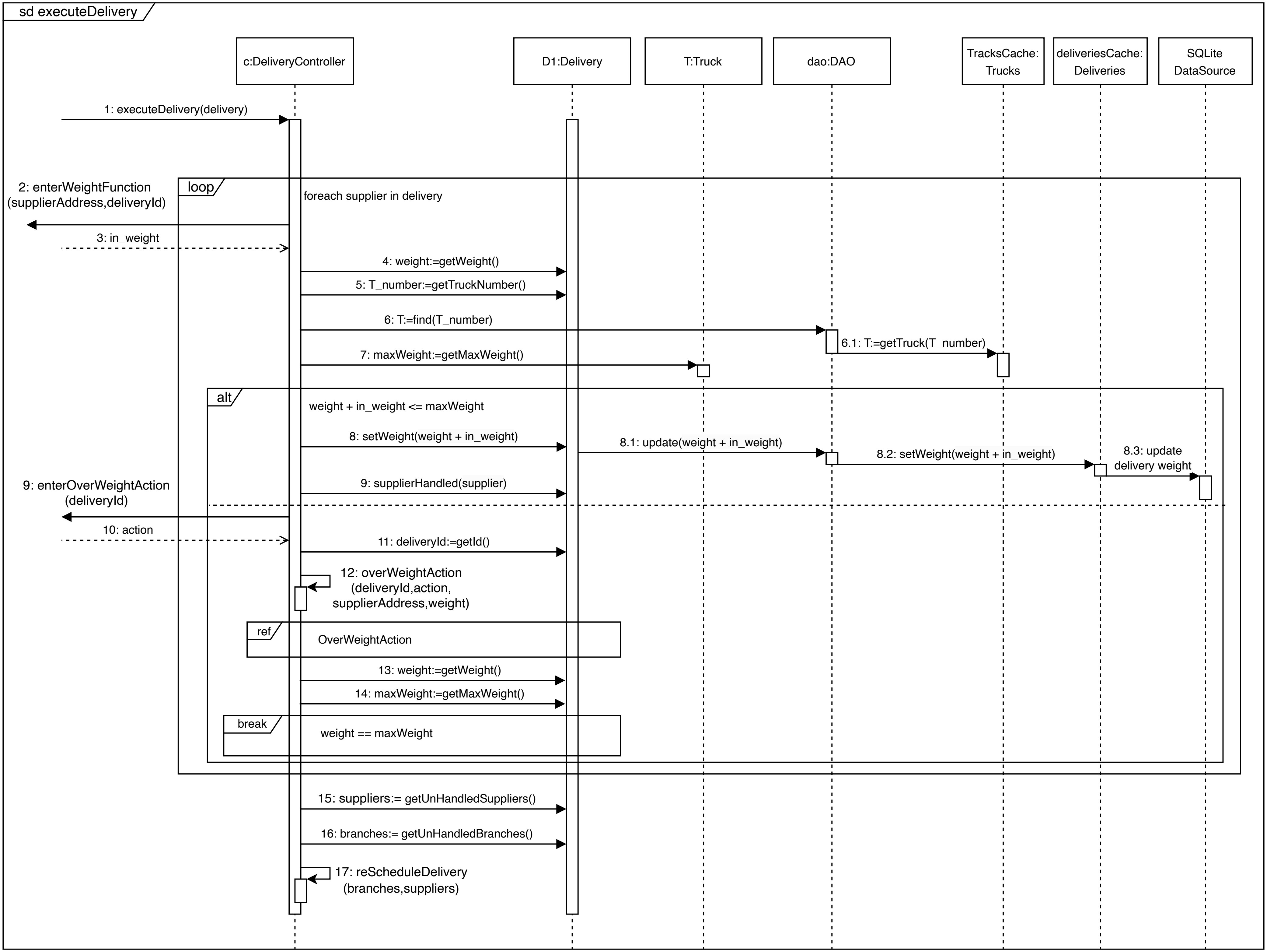
Contract ED04 - Enter overweight action

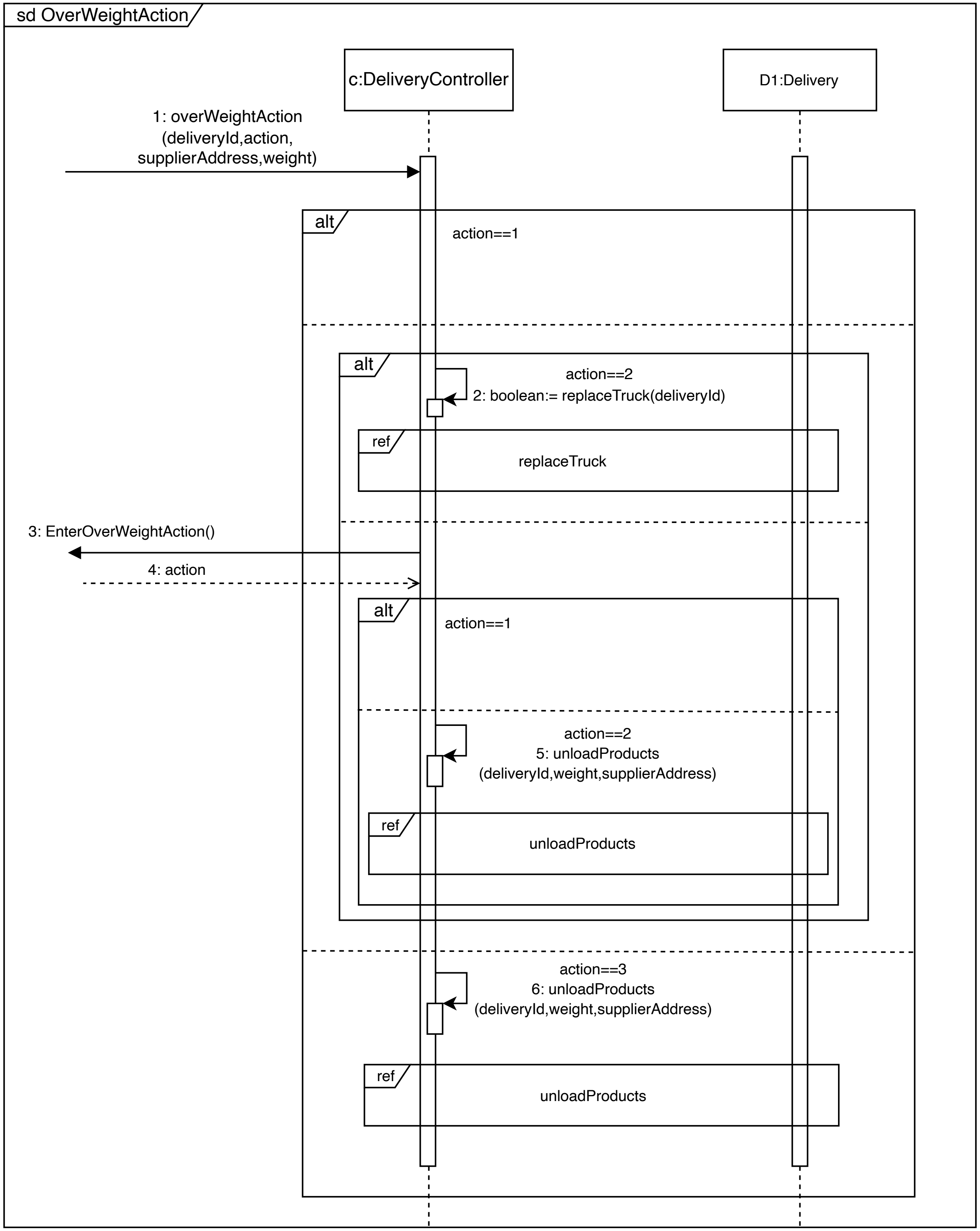
Operation: Cross enterOverweightFunction(deliveryId)

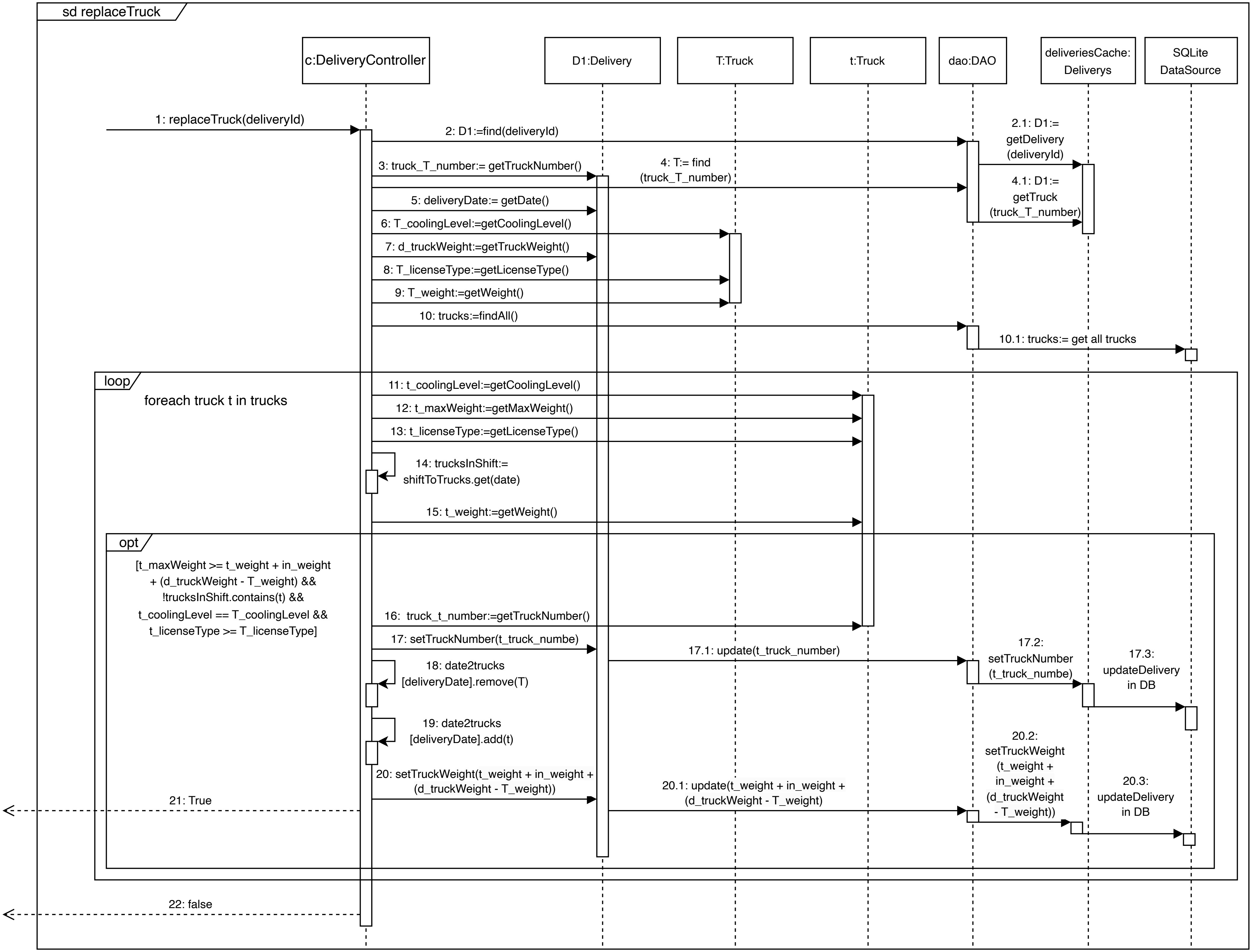
References: use cases: Execute delivery

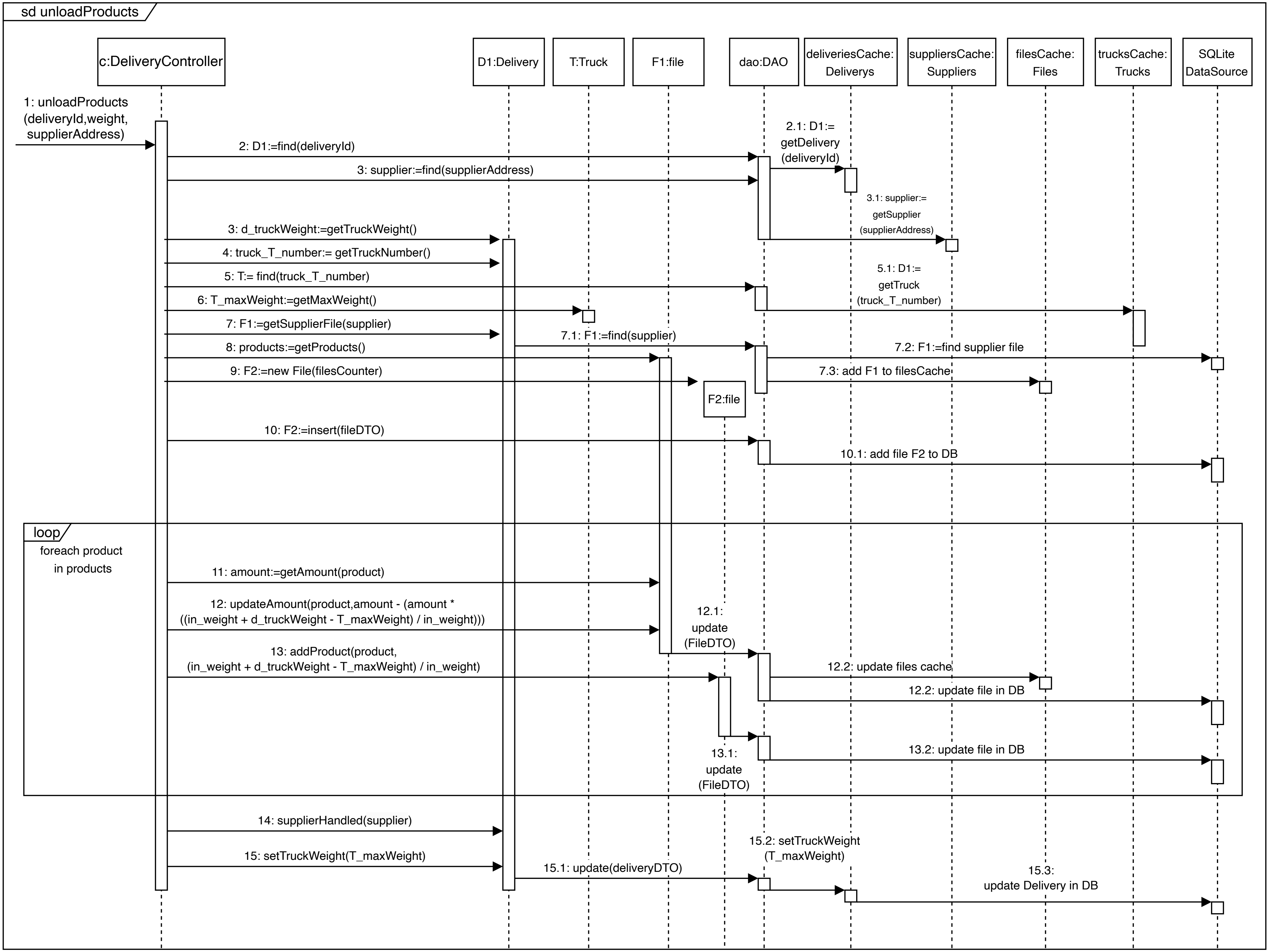
PreConditions: -The delivery has an overweight after loading product from supplier

PostConditions: -The transport manager selected an action to handle the overweight


sd executeDelivery
c:DeliveryController
D1:Delivery
T:Truck
dao:DAO
TracksCache:
Trucks
deliveriesCache:
Deliveries
SQLite
DataSource
1: executeDelivery(delivery)
loop
foreach supplier in delivery
2: enterWeightFunction
(supplierAddress,deliveryId)
3: in_weight
4: weight:=getWeight()
5: T_number:=getTruckNumber()
6: T:=find(T_number)
6.1: T:=getTruck(T_number)
7: maxWeight:=getMaxWeight()
alt
weight + in_weight <= maxWeight
8: setWeight(weight + in_weight)
8.1: update(weight + in_weight)
8.2: setWeight(weight + in_weight)
8.3: update
delivery weight
9: supplierHandled(supplier)
9: enterOverWeightAction
(deliveryId)
10: action
11: deliveryId:=getId()
12: overWeightAction
(deliveryId,action,
supplierAddress,weight)
ref
OverWeightAction
13: weight:=getWeight()
14: maxWeight:=getMaxWeight()
break
weight == maxWeight
15: suppliers:= getUnHandledSuppliers()
16: branches:= getUnHandledBranches()
17: reScheduleDelivery
(branches,suppliers)

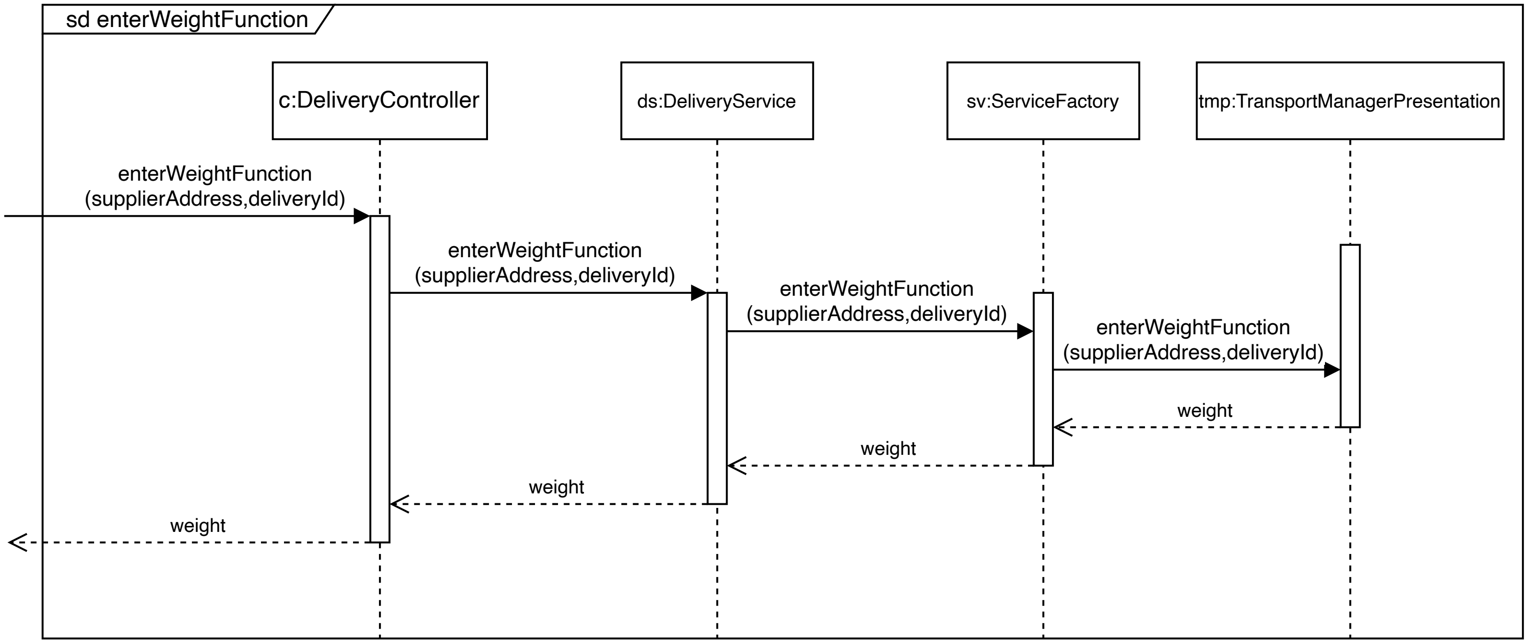

sd OverWeightAction
c:DeliveryController
D1:Delivery
1: overWeightAction
(deliveryId,action,
supplierAddress,weight)
alt
action==1
alt
action==2
2: boolean:= replaceTruck(deliveryId)
ref
replaceTruck
3: EnterOverWeightAction()
4: action
alt
action==1
action==2
5: unloadProducts
(deliveryId,weight,supplierAddress)
ref
unloadProducts
action==3
6: unloadProducts
(deliveryId,weight,supplierAddress)
ref
unloadProducts

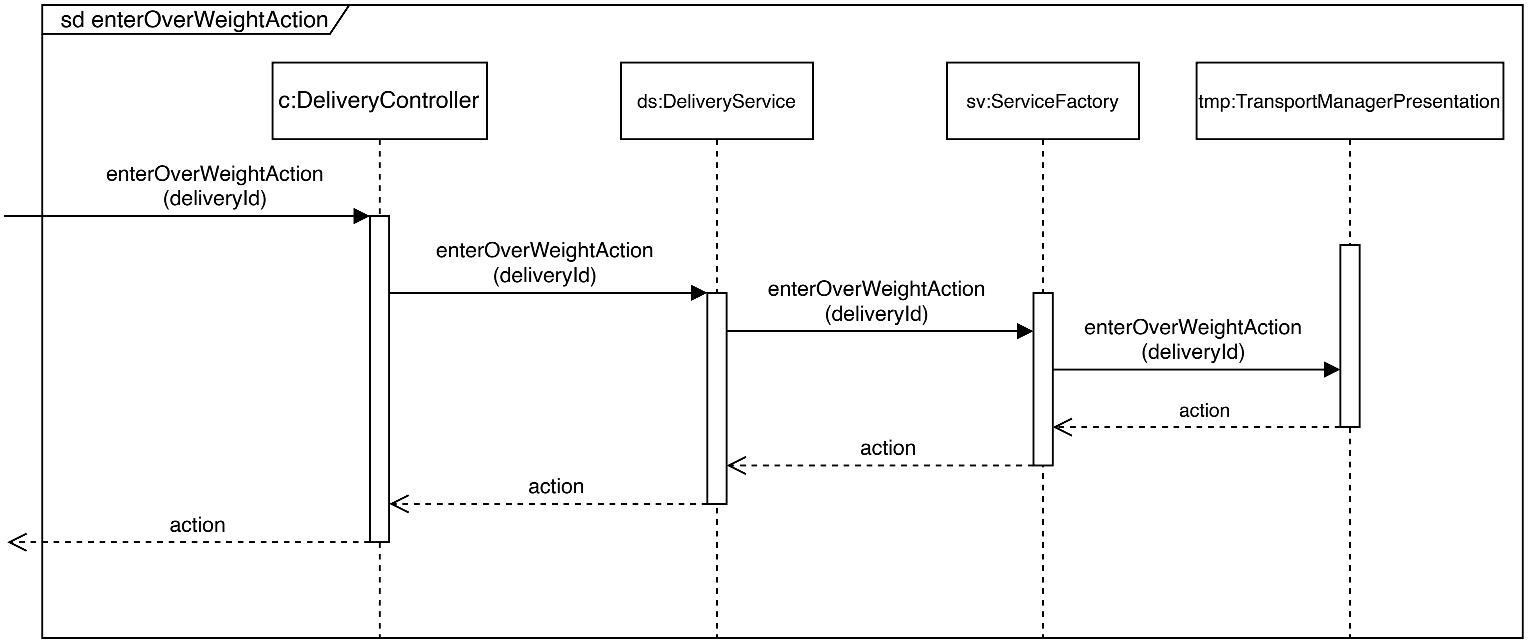

sd replaceTruck
c:DeliveryController
D1:Delivery
T:Truck
t:Truck
dao:DAO
deliveriesCache:
Deliverys
SQLite
DataSource
1: replaceTruck(deliveryId)
2: D1:=find(deliveryId)
2.1: D1:=
getDelivery
(deliveryId)
3: truck_T_number:= getTruckNumber()
4: T:= find
(truck_T_number)
4.1: D1:=
getTruck
(truck_T_number)
5: deliveryDate:= getDate()
6: T_coolingLevel:=getCoolingLevel()
7: d_truckWeight:=getTruckWeight()
8: T_licenseType:=getLicenseType()
9: T_weight:=getWeight()
10: trucks:=findAll()
10.1: trucks:= get all trucks
loop
foreach truck t in trucks
11: t_coolingLevel:=getCoolingLevel()
12: t_maxWeight:=getMaxWeight()
13: t_licenseType:=getLicenseType()
14: trucksInShift:=
shiftToTrucks.get(date)
15: t_weight:=getWeight()
opt
[t_maxWeight >= t_weight + in_weight
+ (d_truckWeight - T_weight) &&
!trucksInShift.contains(t) &&
t_coolingLevel == T_coolingLevel &&
t_licenseType >= T_licenseType]
16: truck_t_number:=getTruckNumber()
17: setTruckNumber(t_truck_numbe)
17.1: update(t_truck_number)
17.2:
setTruckNumber
(t_truck_numbe)
17.3:
updateDelivery
in DB
18: date2trucks
[deliveryDate].remove(T)
19: date2trucks
[deliveryDate].add(t)
20: setTruckWeight(t_weight + in_weight +
(d_truckWeight - T_weight))
20.1: update(t_weight + in_weight +
(d_truckWeight - T_weight)
20.2:
setTruckWeight
(t_weight +
in_weight +
(d_truckWeight
- T_weight))
20.3:
updateDelivery
in DB
21: True
22: false


sd unloadProducts
c:DeliveryController
D1:Delivery
T:Truck
F1:file
dao:DAO
deliveriesCache: Deliverys
suppliersCache: Suppliers
filesCache: Files
trucksCache: Trucks
SQLite DataSource
1: unloadProducts (deliveryId,weight, supplierAddress)
2: D1:=find(deliveryId)
2.1: D1:= getDelivery (deliveryId)
3: supplier:=find(supplierAddress)
3.1: supplier:= getSupplier (supplierAddress)
3: d_truckWeight:=getTruckWeight()
4: truck_T_number:= getTruckNumber()
5: T:= find(truck_T_number)
5.1: D1:= getTruck (truck_T_number)
6: T_maxWeight:=getMaxWeight()
7: F1:=getSupplierFile(supplier)
7.1: F1:=find(supplier)
7.2: F1:=find supplier file
8: products:=getProducts()
7.3: add F1 to filesCache
9: F2:=new File(filesCounter)
F2:file
10: F2:=insert(fileDTO)
10.1: add file F2 to DB
loop
foreach product in products
11: amount:=getAmount(product)
12: updateAmount(product,amount - (amount * ((in_weight + d_truckWeight - T_maxWeight) / in_weight)))
12.1: update (FileDTO)
13: addProduct(product, (in_weight + d_truckWeight - T_maxWeight) / in_weight)
12.2: update files cache
12.2: update file in DB
13.1: update (FileDTO)
13.2: update file in DB
14: supplierHandled(supplier)
15: setTruckWeight(T_maxWeight)
15.1: update(deliveryDTO)
15.2: setTruckWeight (T_maxWeight)
15.3: update Delivery in DB

the next diagrams represents process of receiving input from the user using callbacks

לא היה ברור מדרישות העבודה האם צריך לציין על תהליכים אלו


sd enterWeightFunction
c:DeliveryController
ds:DeliveryService
sv:ServiceFactory
tmp:TransportManagerPresentation
enterWeightFunction (supplierAddress,deliveryId)
enterWeightFunction (supplierAddress,deliveryId)
enterWeightFunction (supplierAddress,deliveryId)
enterWeightFunction (supplierAddress,deliveryId)
weight
weight
weight
weight


sd enterOverWeightAction
c:DeliveryController
ds:DeliveryService
sv:ServiceFactory
tmp:TransportManagerPresentation
enterOverWeightAction (deliveryId)
enterOverWeightAction (deliveryId)
enterOverWeightAction (deliveryId)
enterOverWeightAction (deliveryId)
action
action
action
action