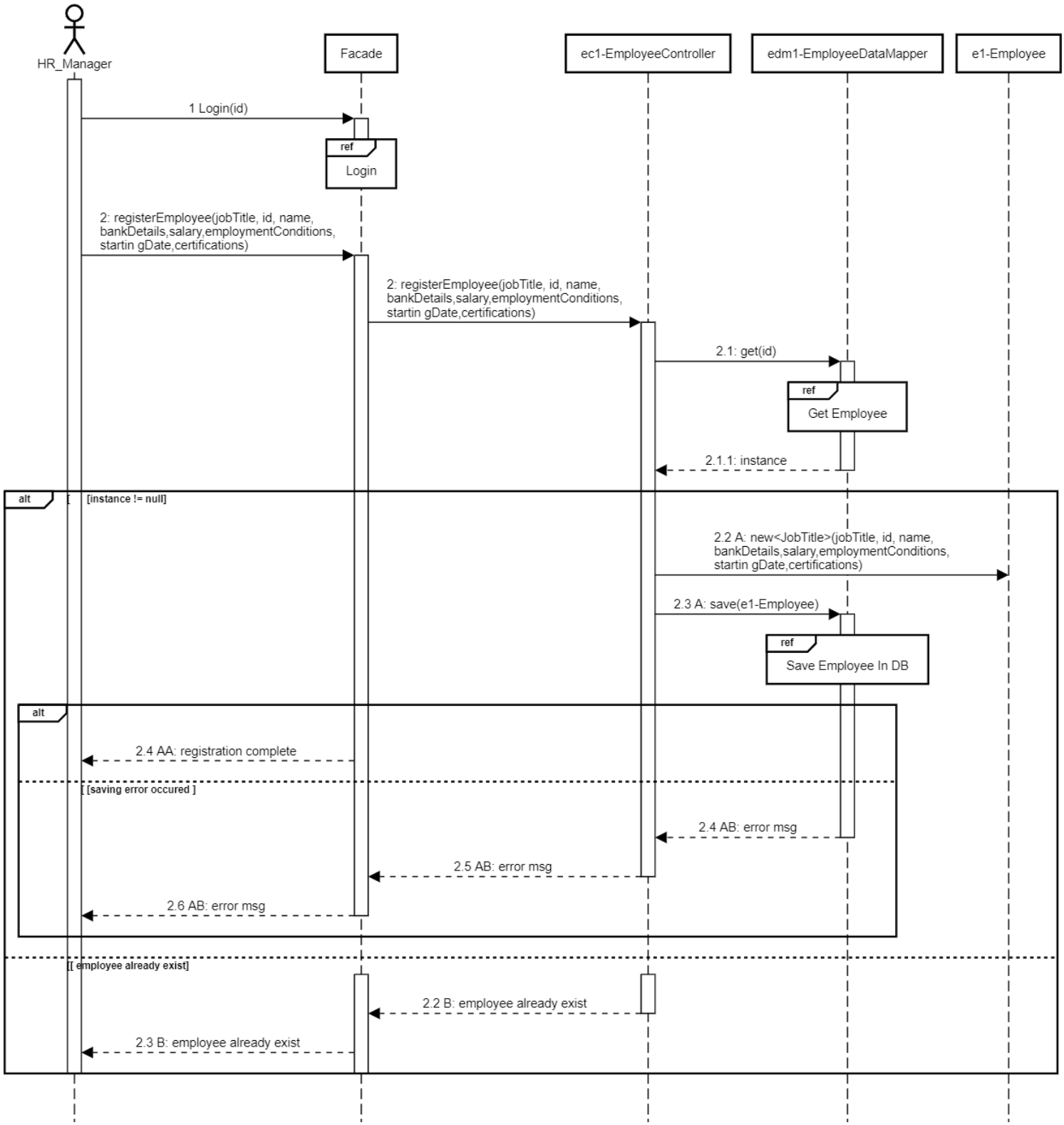
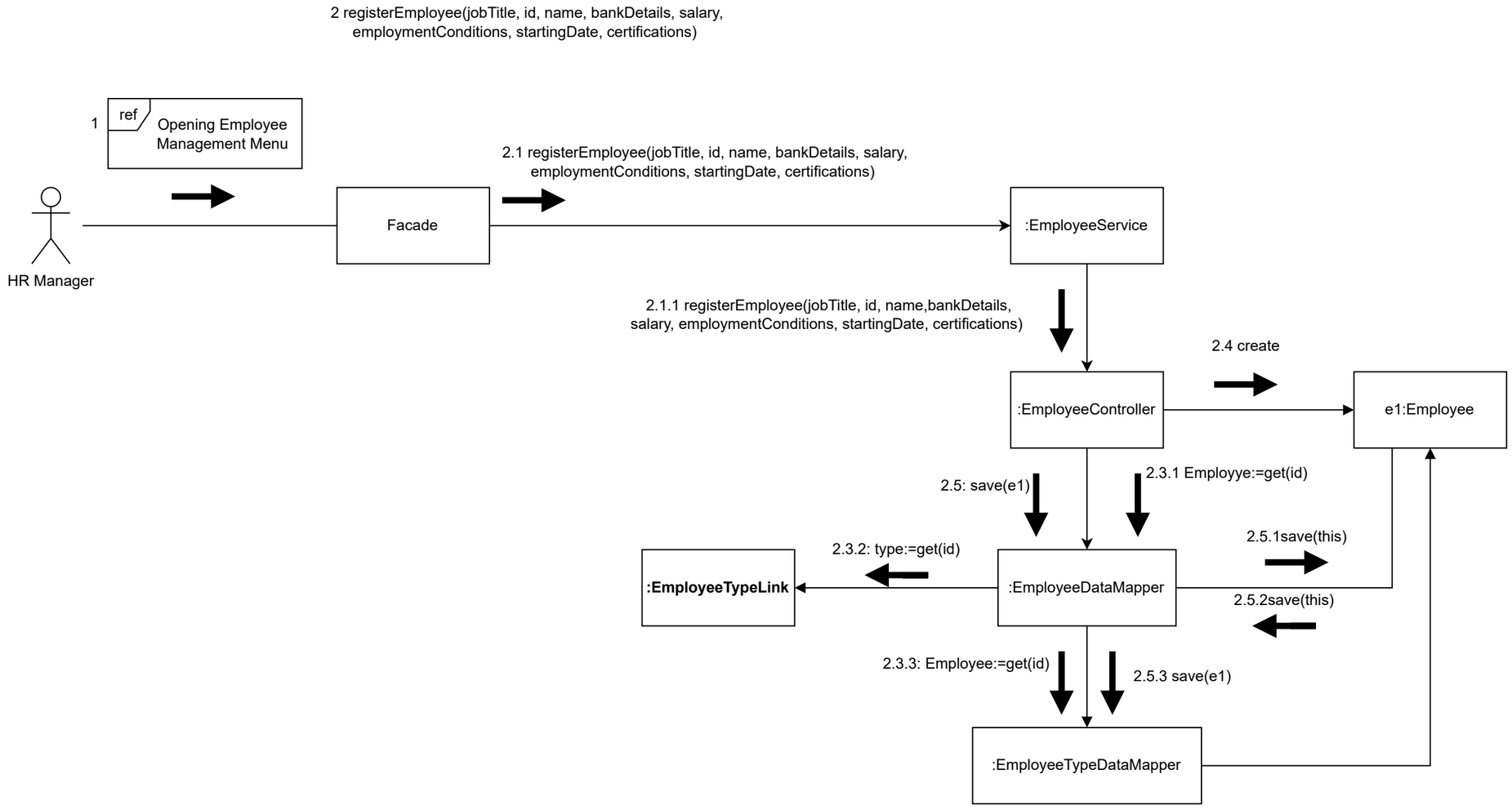
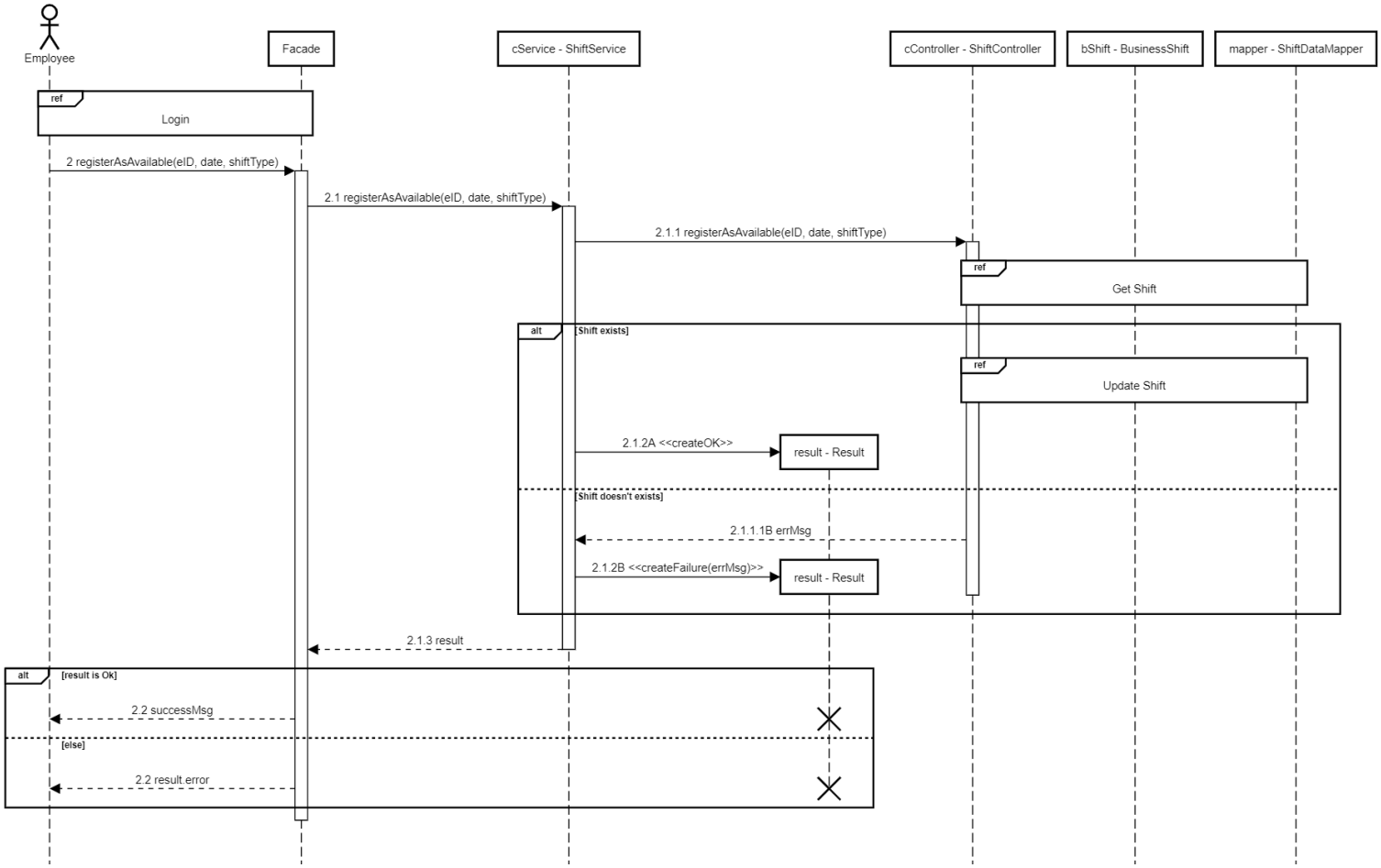
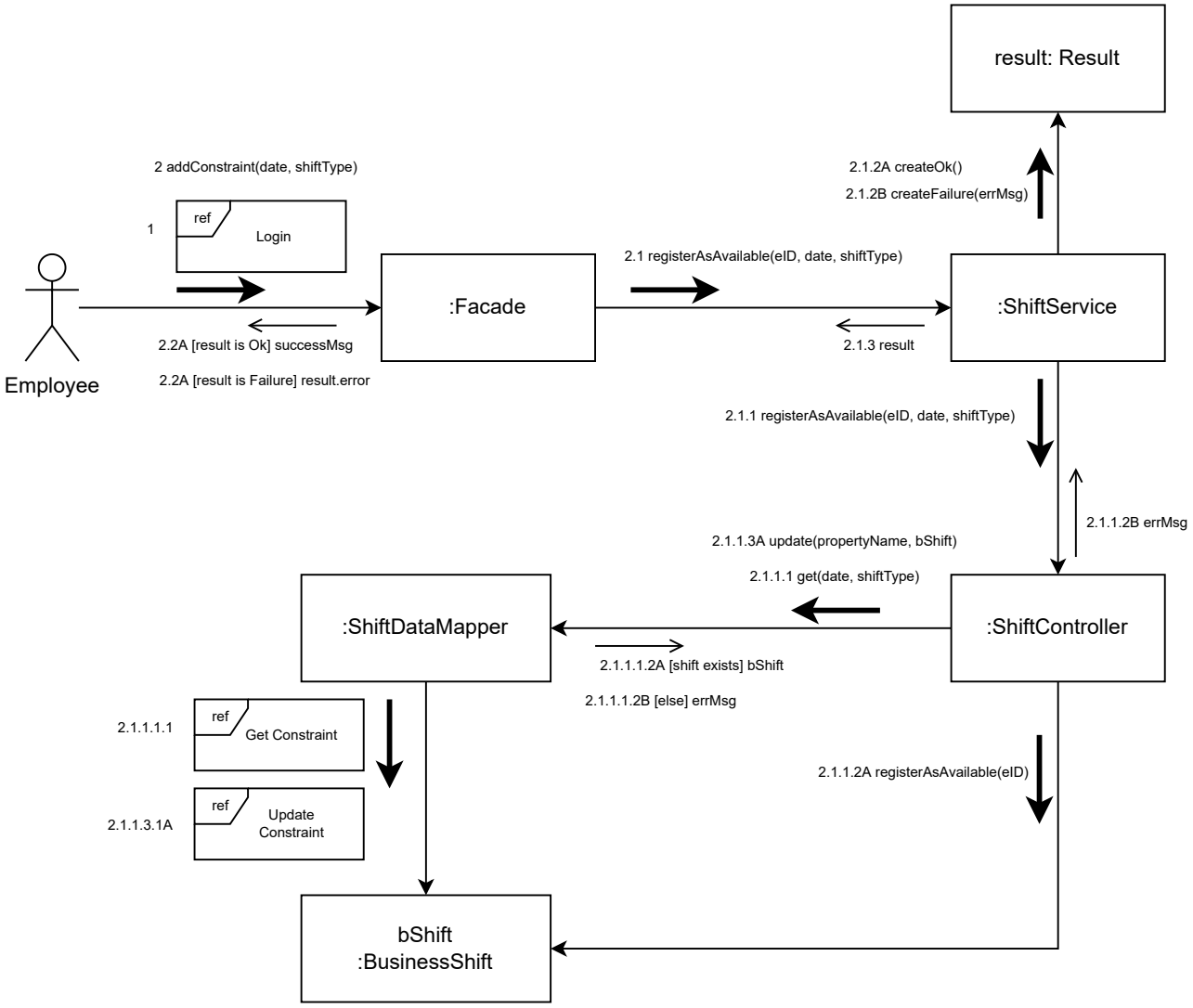
New Employee Registration



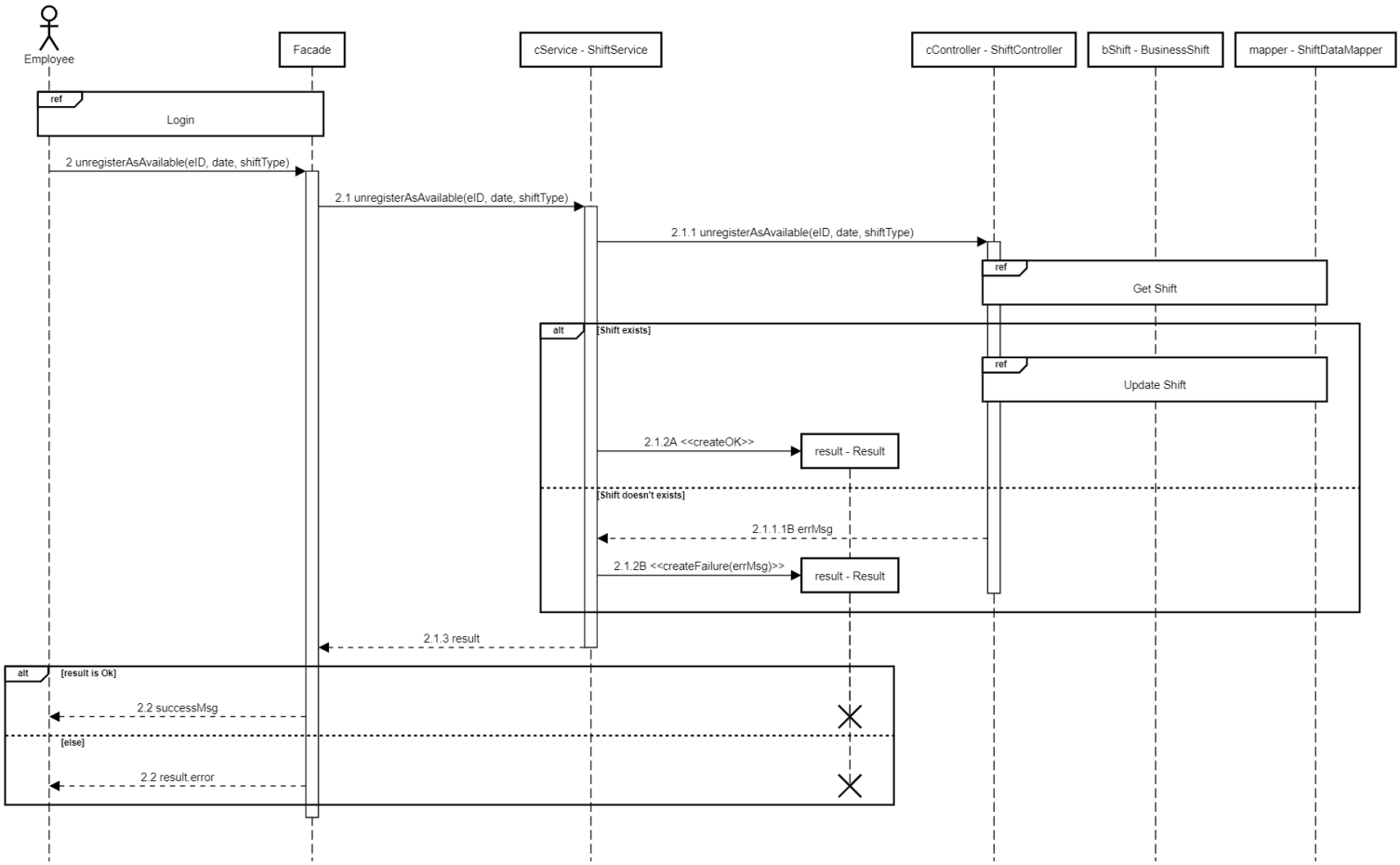

2 registerEmployee(jobTitle, id, name, bankDetails, salary,
employmentConditions, startingDate, certifications)
1
ref
Opening Employee
Management Menu
HR Manager
Facade
2.1 registerEmployee(jobTitle, id, name, bankDetails, salary,
employmentConditions, startingDate, certifications)
:EmployeeService
2.1.1 registerEmployee(jobTitle, id, name,bankDetails,
salary, employmentConditions, startingDate, certifications)
2.4 create
:EmployeeController
e1:Employee
2.5: save(e1)
2.3.1 Employye:=get(id)
2.3.2: type:=get(id)
:EmployeeTypeLink
:EmployeeDataMapper
2.5.1save(this)
2.5.2save(this)
2.3.3: Employee:=get(id)
2.5.3 save(e1)
:EmployeeTypeDataMapper

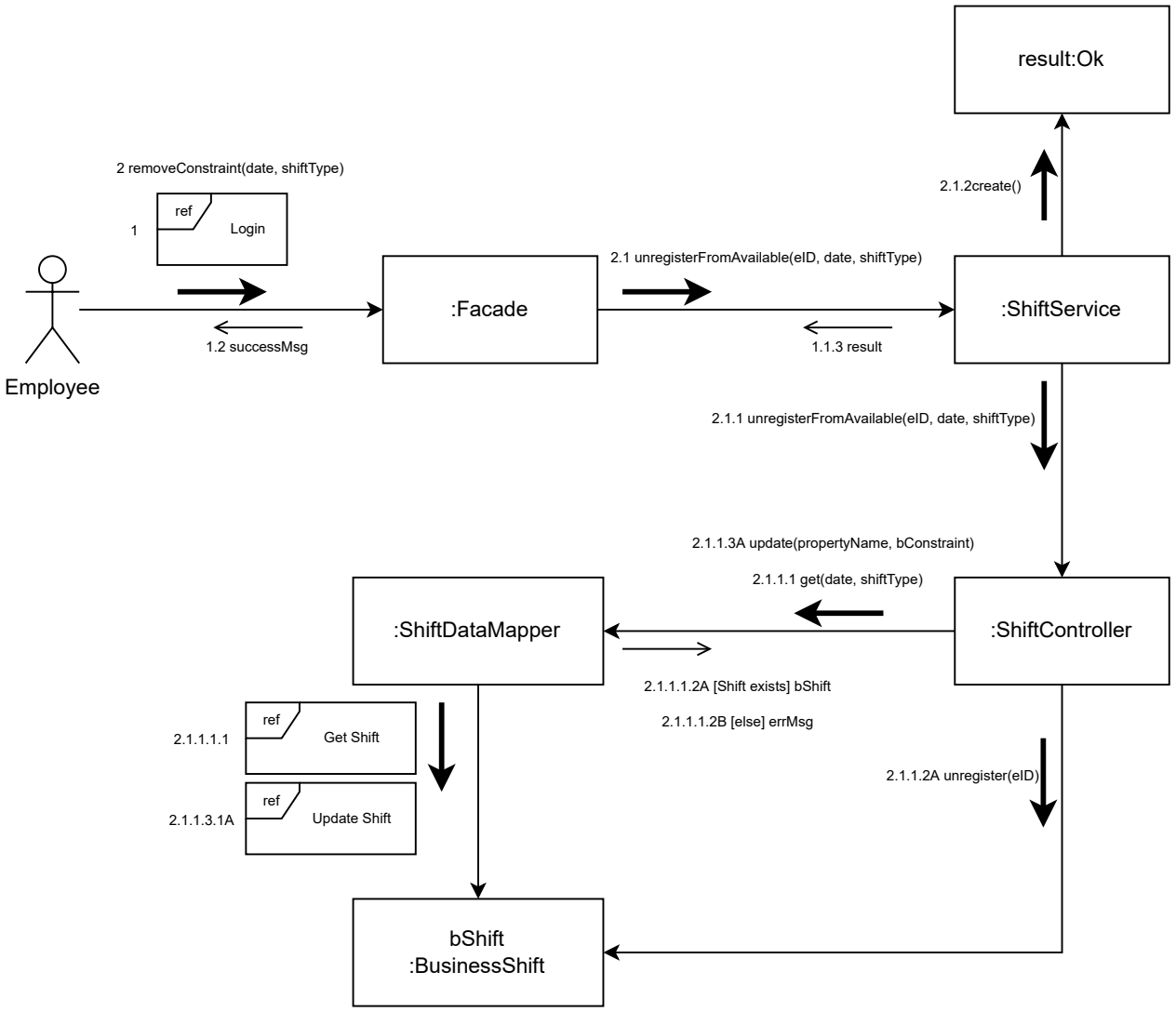
Adding Constraint



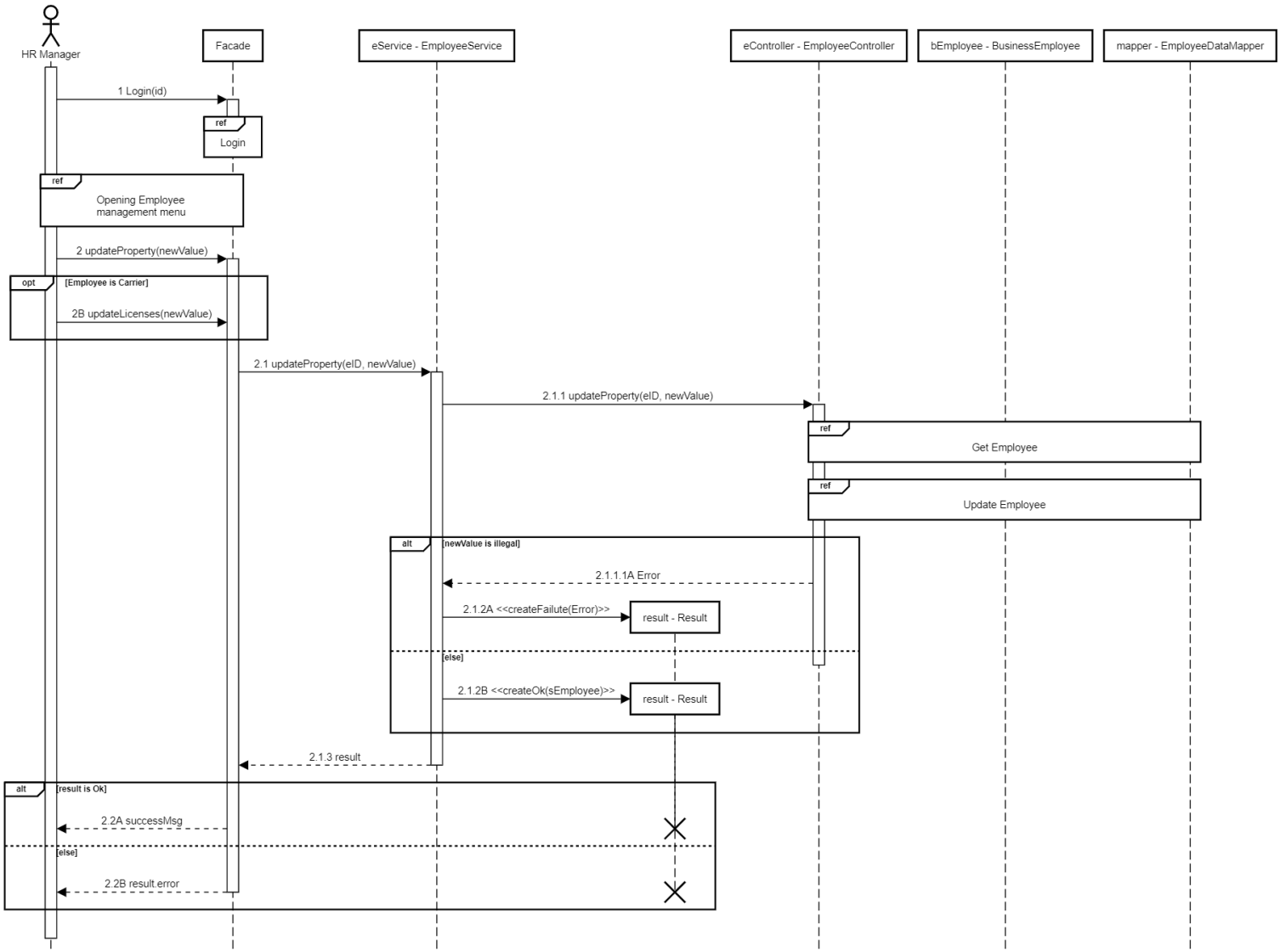

result: Result
2 addConstraint(date, shiftType)
2.1.2A createOk()
2.1.2B createFailure(errMsg)
1
ref
Login
2.1 registerAsAvailable(eID, date, shiftType)
:Facade
:ShiftService
Employee
2.2A [result is Ok] successMsg
2.2A [result is Failure] result.error
2.1.3 result
2.1.1 registerAsAvailable(eID, date, shiftType)
2.1.1.2B errMsg
2.1.1.3A update(propertyName, bShift)
2.1.1.1 get(date, shiftType)
:ShiftDataMapper
:ShiftController
2.1.1.1.2A [shift exists] bShift
2.1.1.1.2B [else] errMsg
2.1.1.1.1
ref
Get Constraint
2.1.1.2A registerAsAvailable(eID)
2.1.1.3.1A
ref
Update Constraint
bShift
:BusinessShift

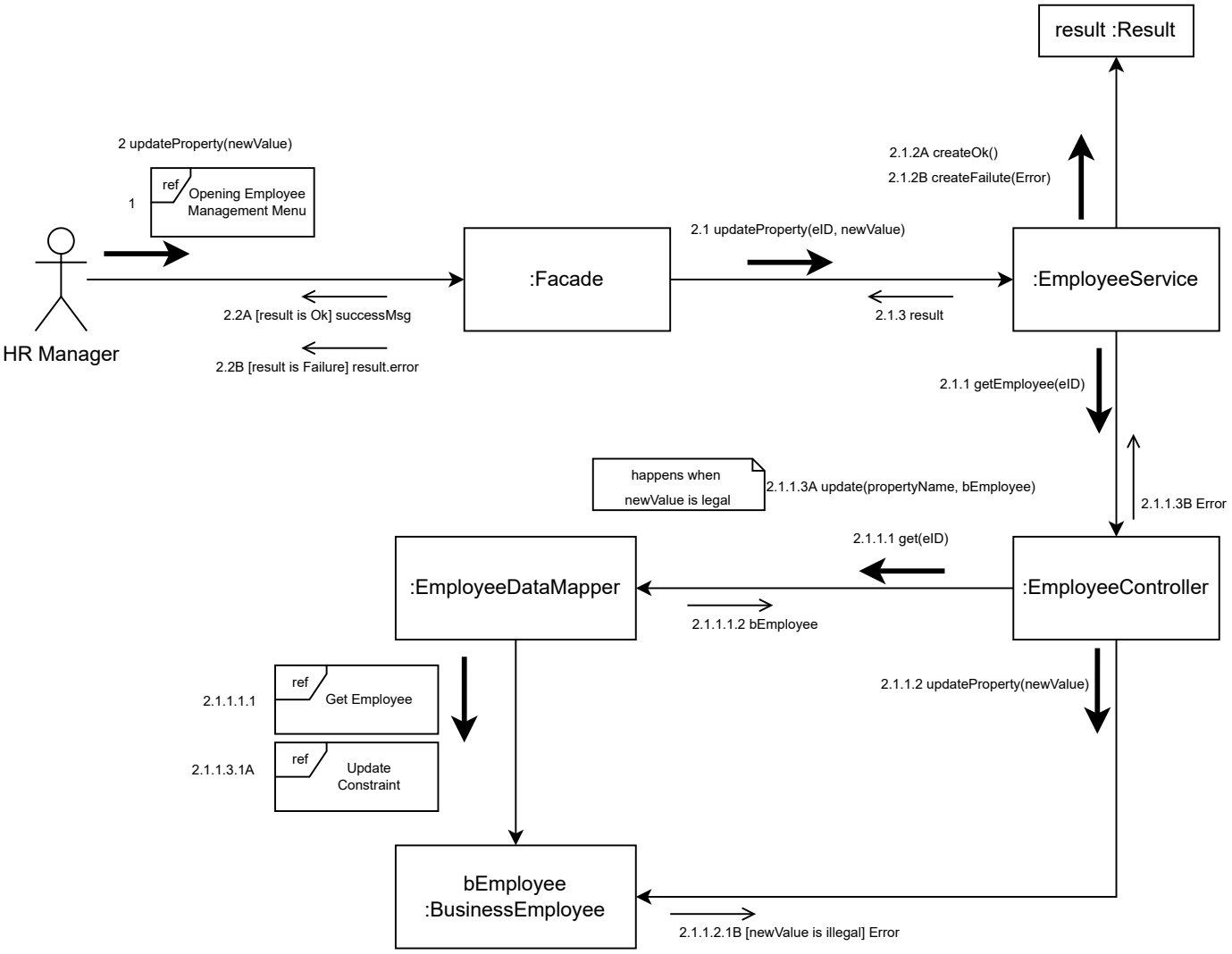
Adding Constraint

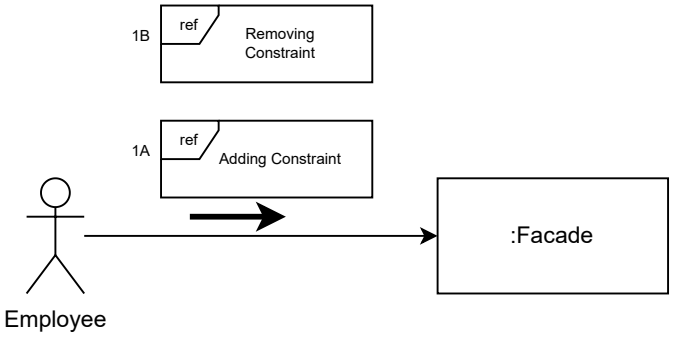



result:Ok
2 removeConstraint(date, shiftType)
ref
1
Login
2.1.2create()
2.1 unregisterFromAvailable(eID, date, shiftType)
:Facade
:ShiftService
Employee
1.2 successMsg
1.1.3 result
2.1.1 unregisterFromAvailable(eID, date, shiftType)
2.1.1.3A update(propertyName, bConstraint)
2.1.1.1 get(date, shiftType)
:ShiftDataMapper
:ShiftController
2.1.1.1.2A [Shift exists] bShift
2.1.1.1.2B [else] errMsg
ref
2.1.1.1.1
Get Shift
2.1.1.2A unregister(eID)
ref
2.1.1.3.1A
Update Shift
bShift
:BusinessShift

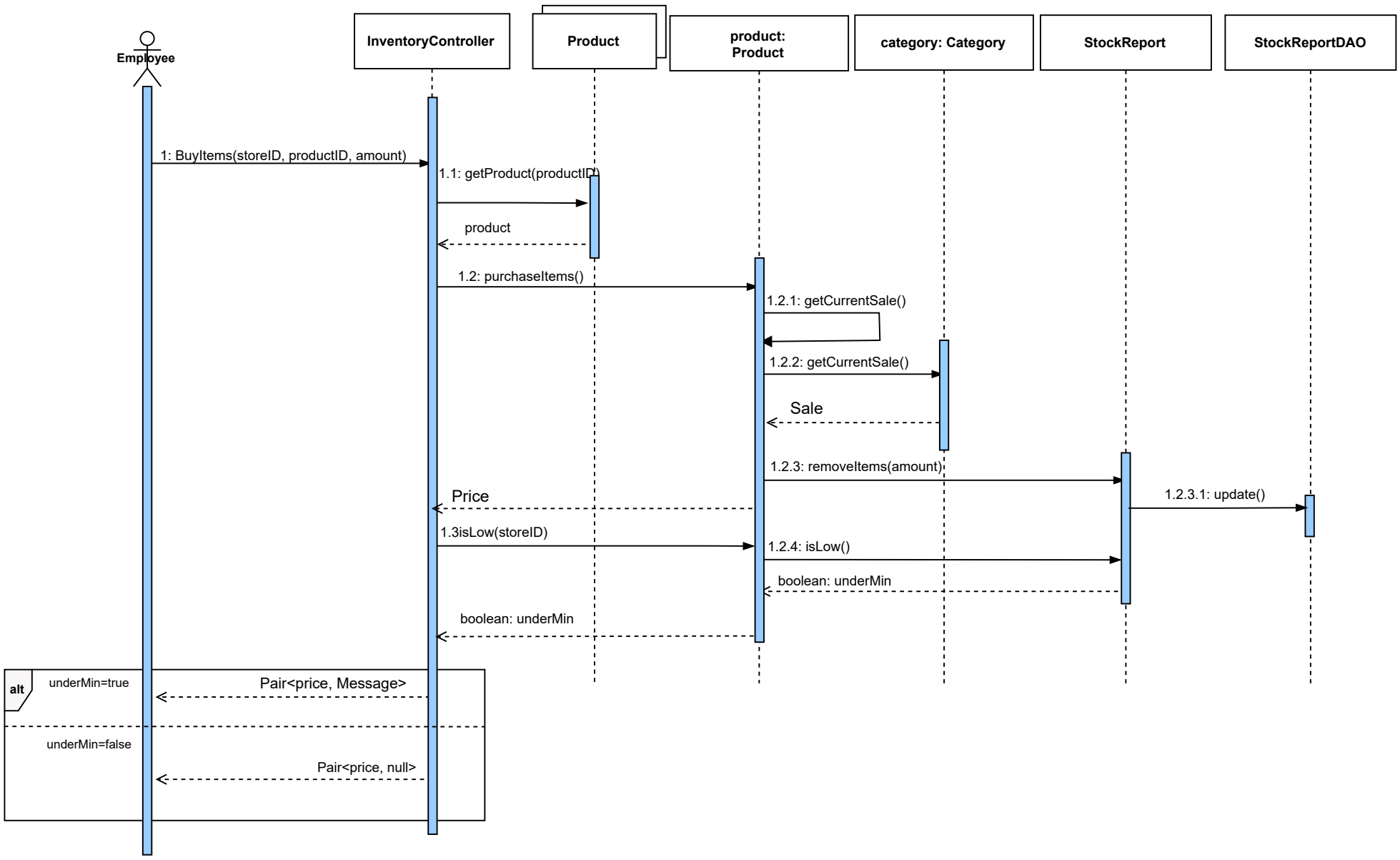
Updating Employee Details

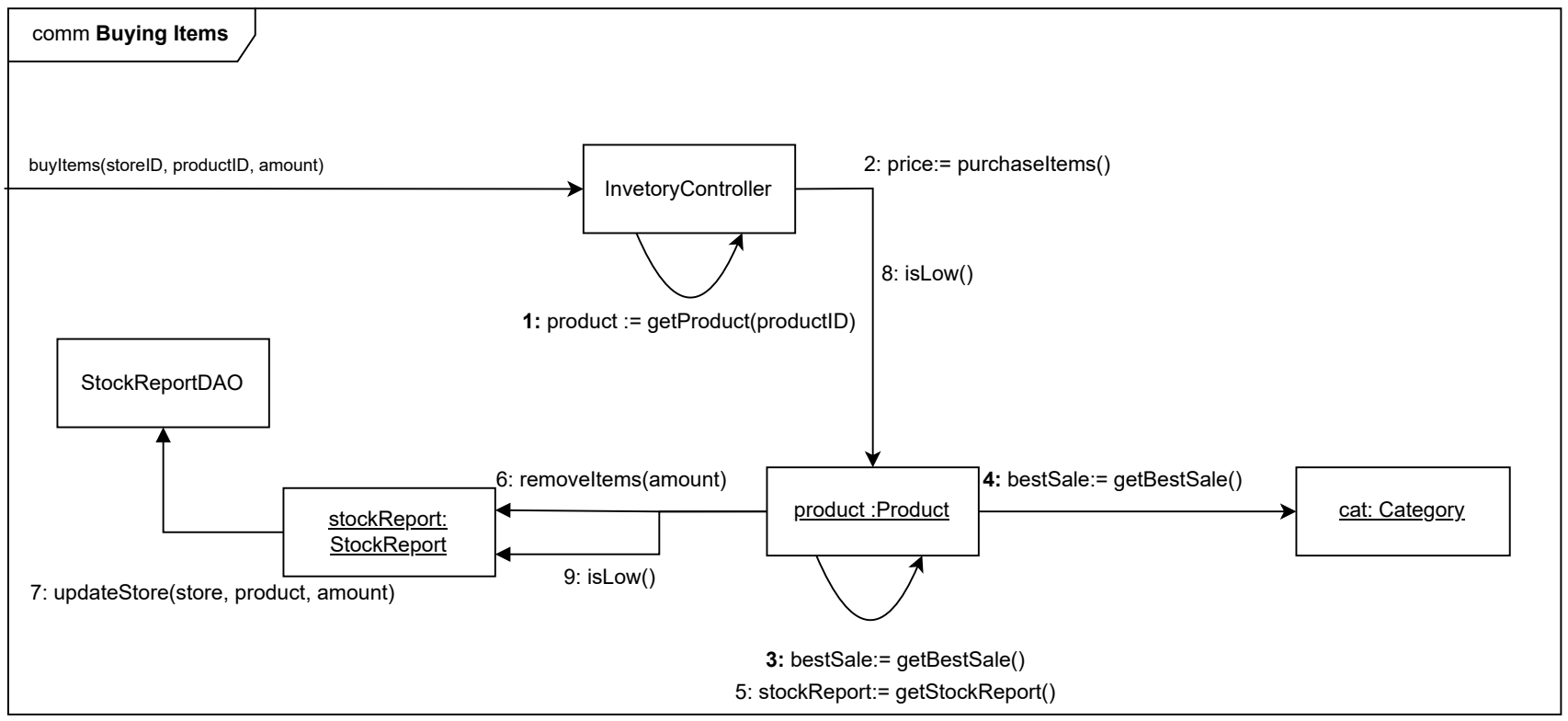



result :Result
2 updateProperty(newValue)
1
ref
Opening Employee Management Menu
2.1.2A createOk()
2.1.2B createFailute(Error)
2.1 updateProperty(eID, newValue)
:Facade
:EmployeeService
HR Manager
2.2A [result is Ok] successMsg
2.2B [result is Failure] result.error
2.1.3 result
2.1.1 getEmployee(eID)
happens when newValue is legal
2.1.1.3A update(propertyName, bEmployee)
2.1.1.3B Error
2.1.1.1 get(eID)
:EmployeeDataMapper
:EmployeeController
2.1.1.1.2 bEmployee
2.1.1.1.1
ref
Get Employee
2.1.1.2 updateProperty(newValue)
2.1.1.3.1A
ref
Update Constraint
bEmployee
:BusinessEmployee
2.1.1.2.1B [newValue is illegal] Error

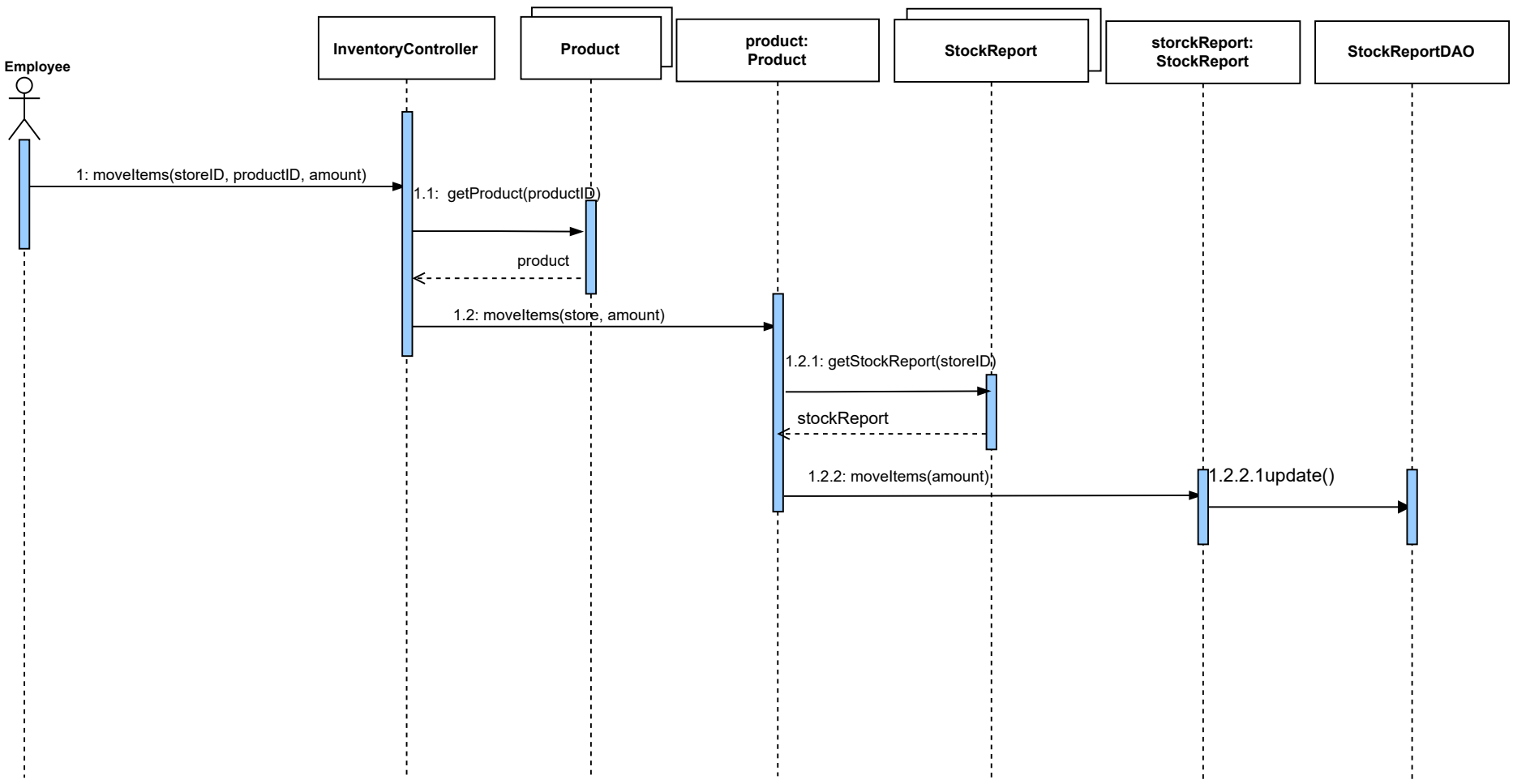

1B
ref
Removing
Constraint
1A
ref
Adding Constraint
:Facade
Employee

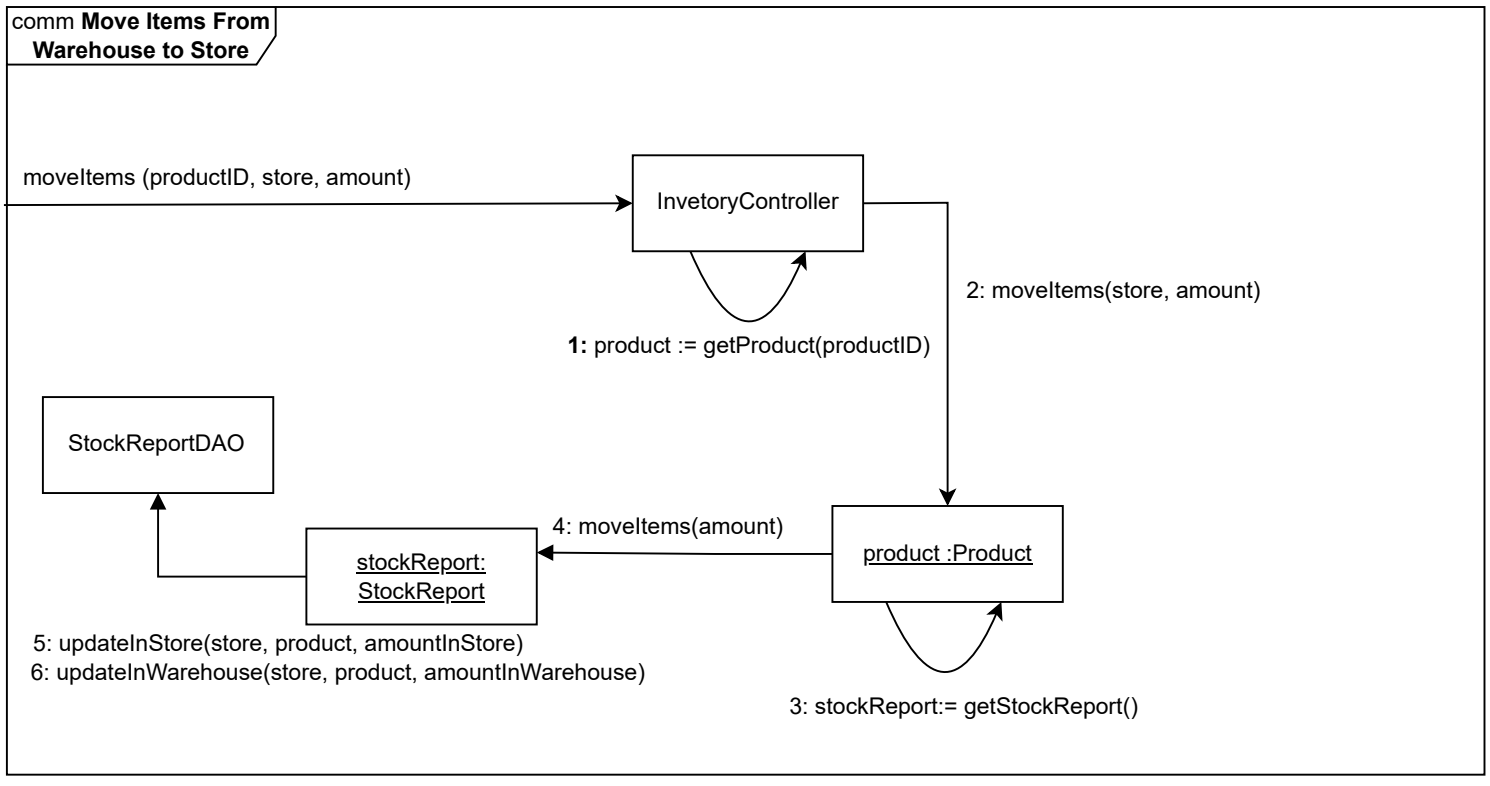
Buying Items



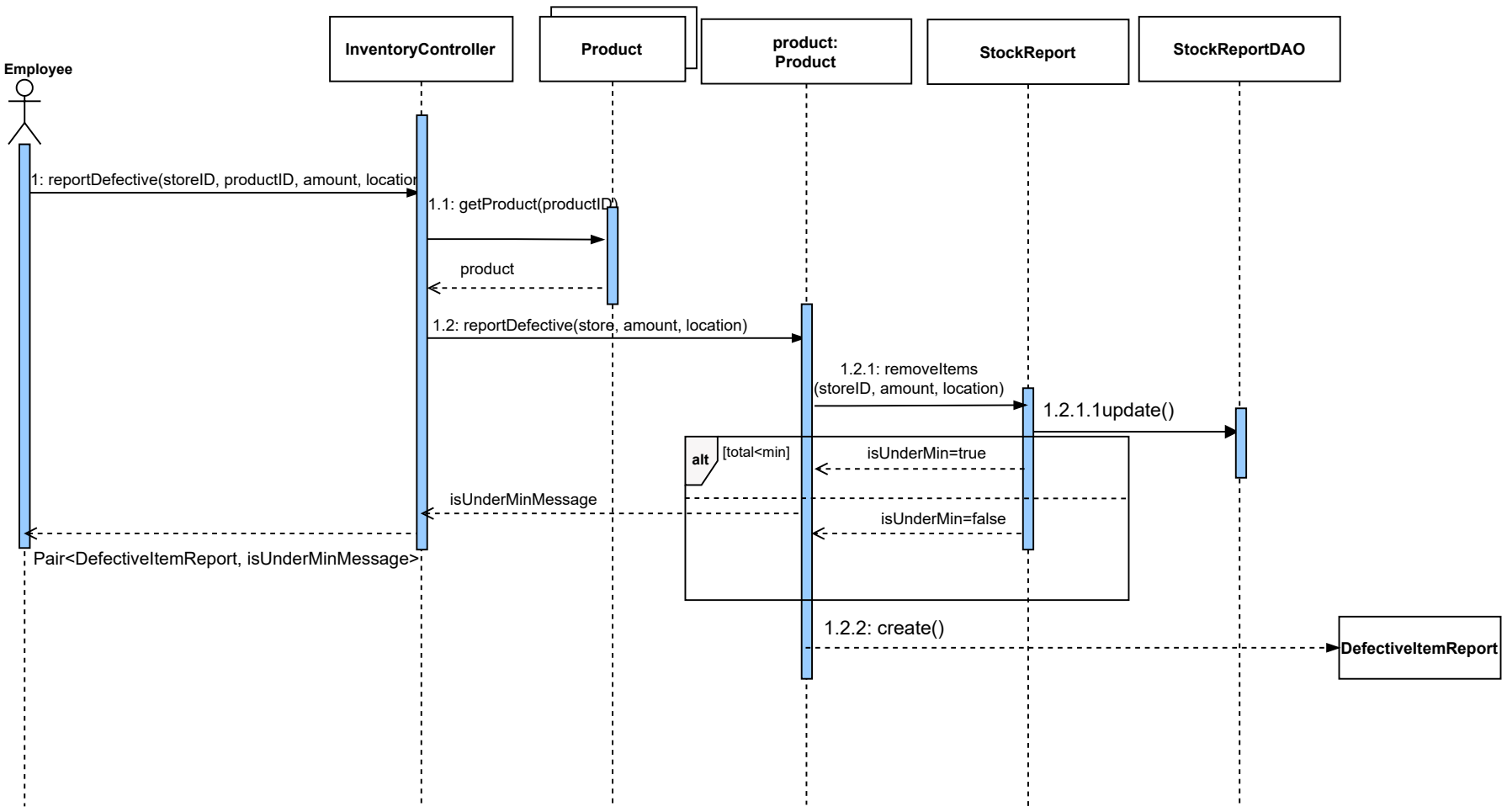

comm **Buying Items**
buyItems(storeID, productID, amount)
InvetoryController
2: price:= purchaseItems()
8: isLow()
1: product := getProduct(productID)
StockReportDAO
6: removeItems(amount)
4: bestSale:= getBestSale()
stockReport: StockReport
product :Product
cat: Category
9: isLow()
7: updateStore(store, product, amount)
3: bestSale:= getBestSale()
5: stockReport:= getStockReport()

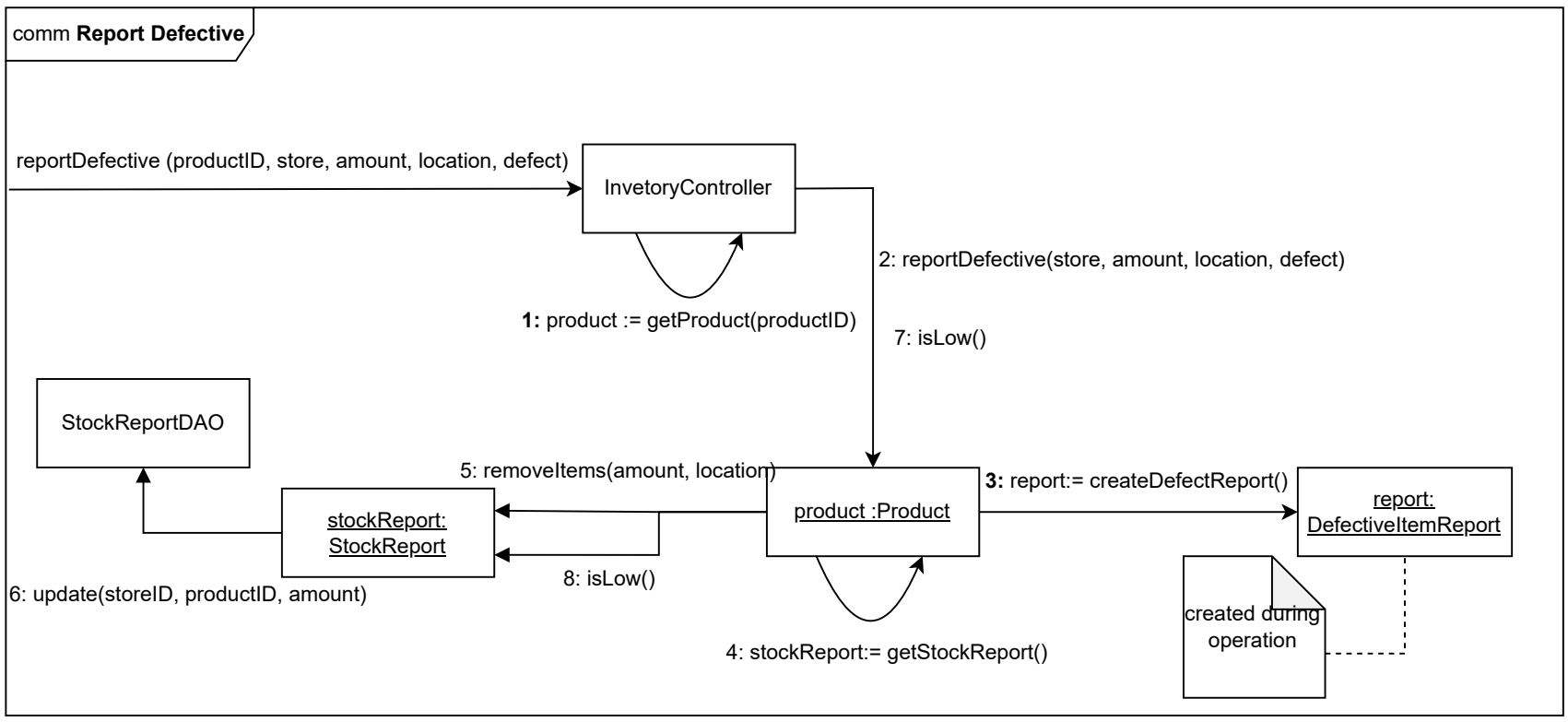
Move Items from warehouse to store



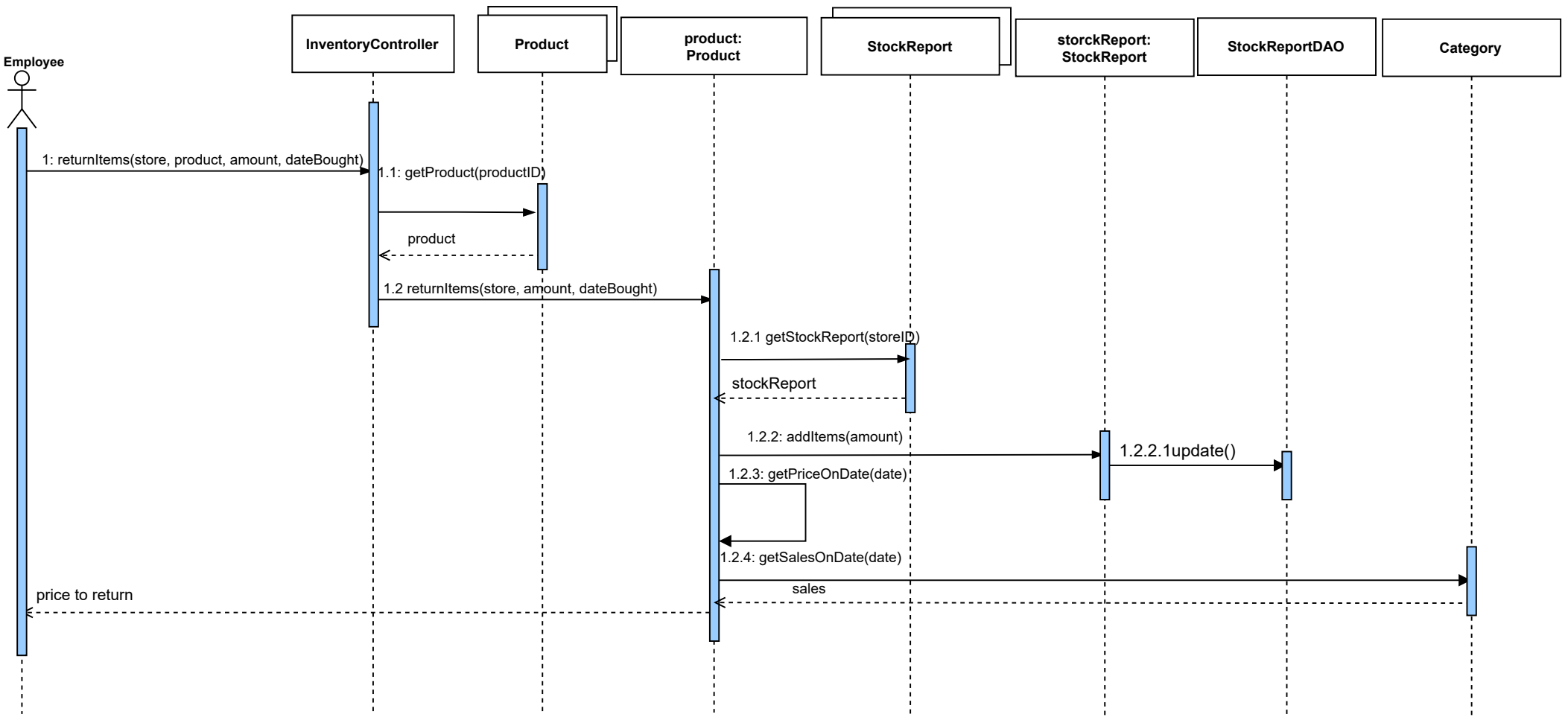

comm **Move Items From Warehouse to Store**
moveItems (productID, store, amount)
InvetoryController
2: moveItems(store, amount)
1: product := getProduct(productID)
StockReportDAO
4: moveItems(amount)
stockReport: StockReport
product :Product
5: updateInStore(store, product, amountInStore)
6: updateInWarehouse(store, product, amountInWarehouse)
3: stockReport:= getStockReport()

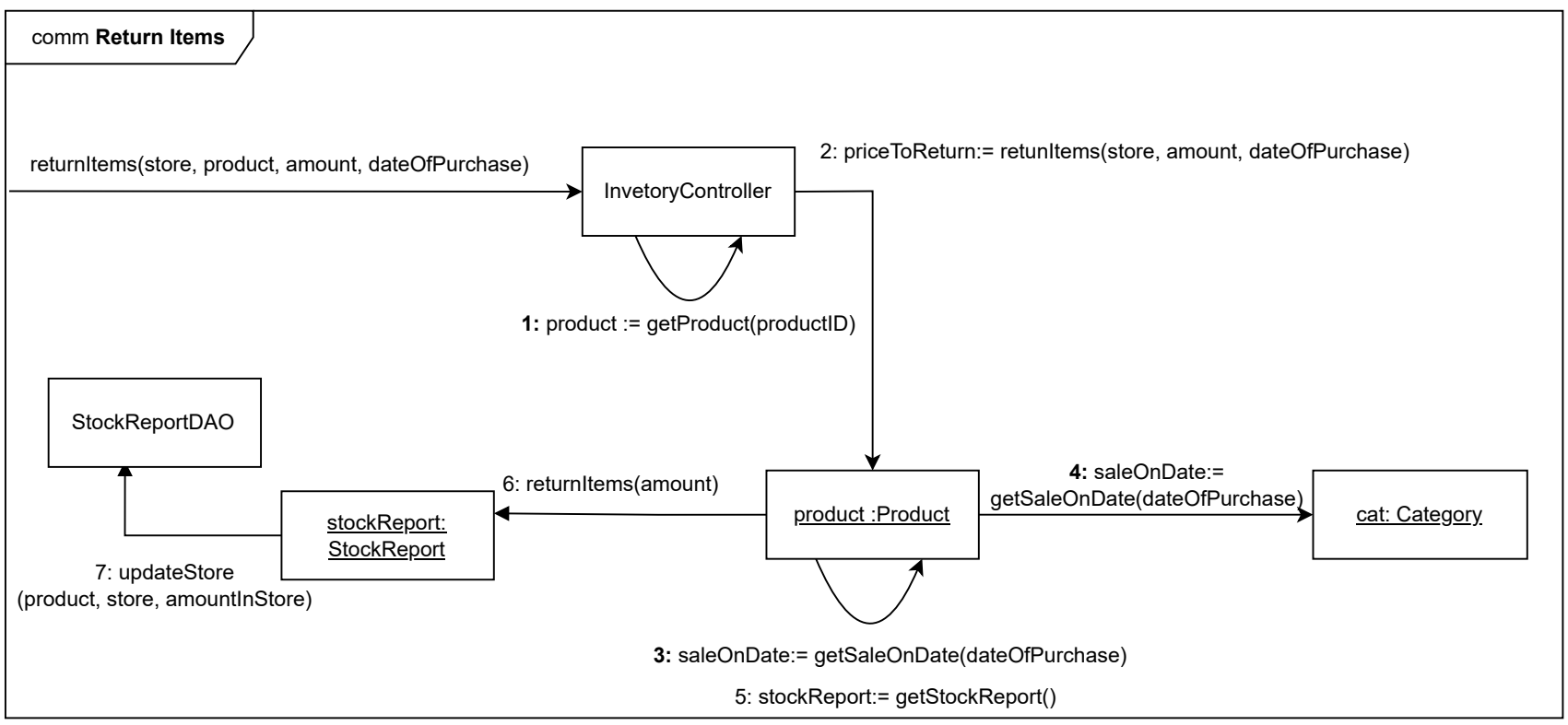
Report Defective

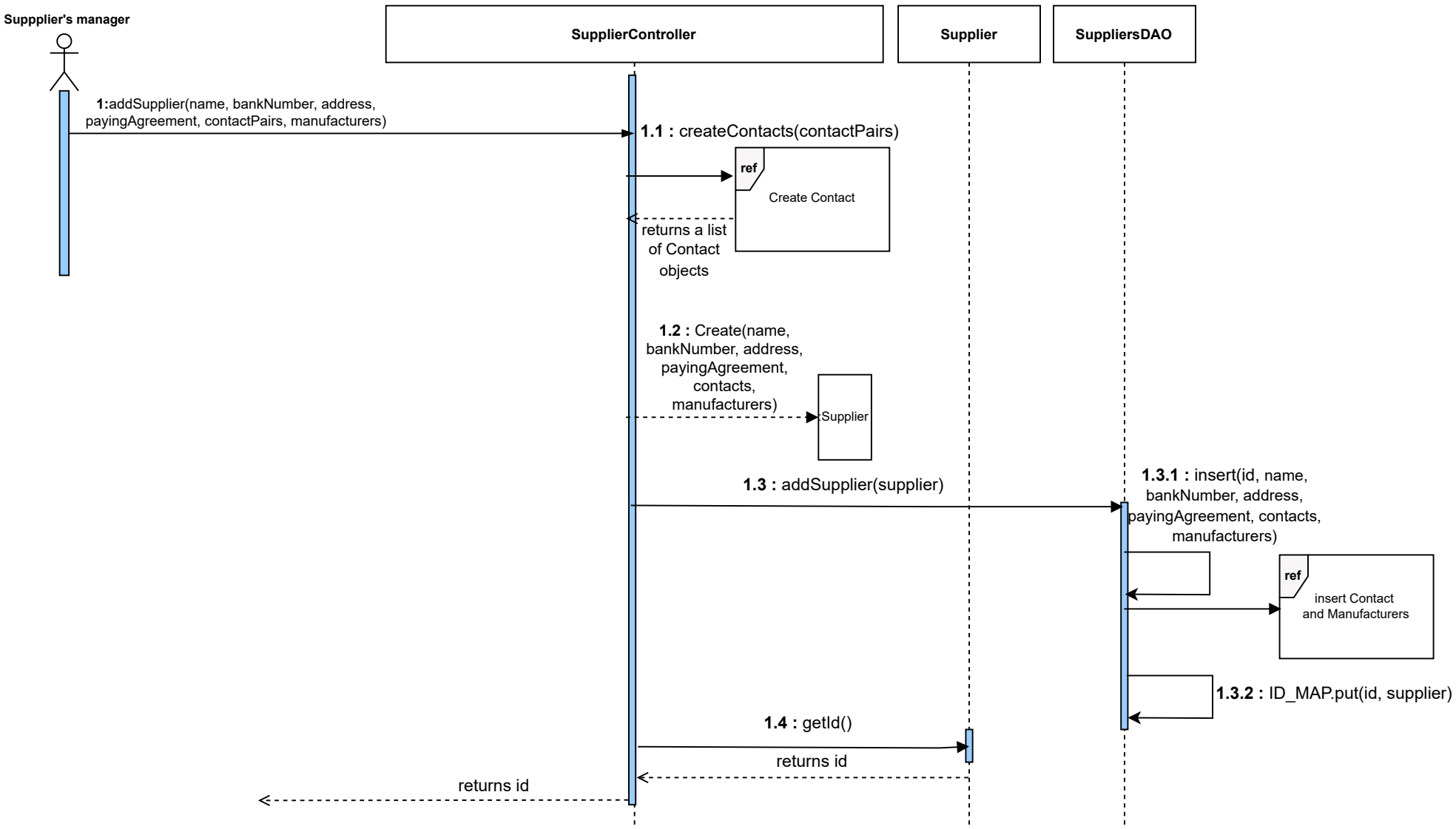


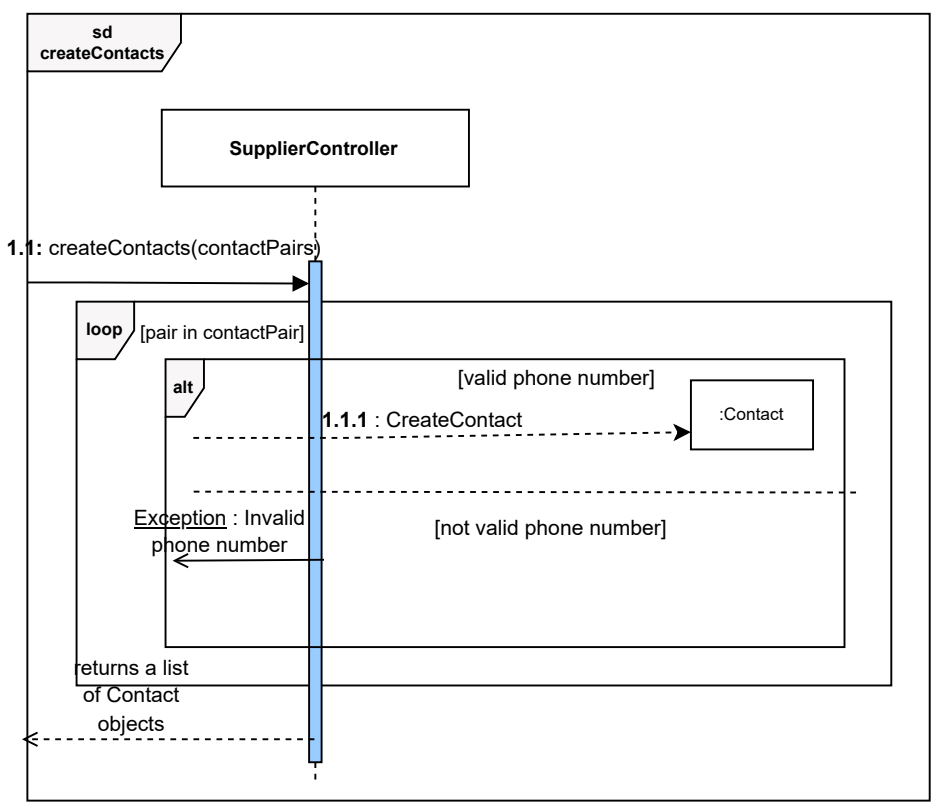

comm **Report Defective**
reportDefective (productID, store, amount, location, defect)
InvetoryController
2: reportDefective(store, amount, location, defect)
1: product := getProduct(productID)
7: isLow()
StockReportDAO
5: removeItems(amount, location)
3: report:= createDefectReport()
stockReport: StockReport
product :Product
report: DefectiveItemReport
8: isLow()
6: update(storeID, productID, amount)
created during operation
4: stockReport:= getStockReport()

ReturnItems

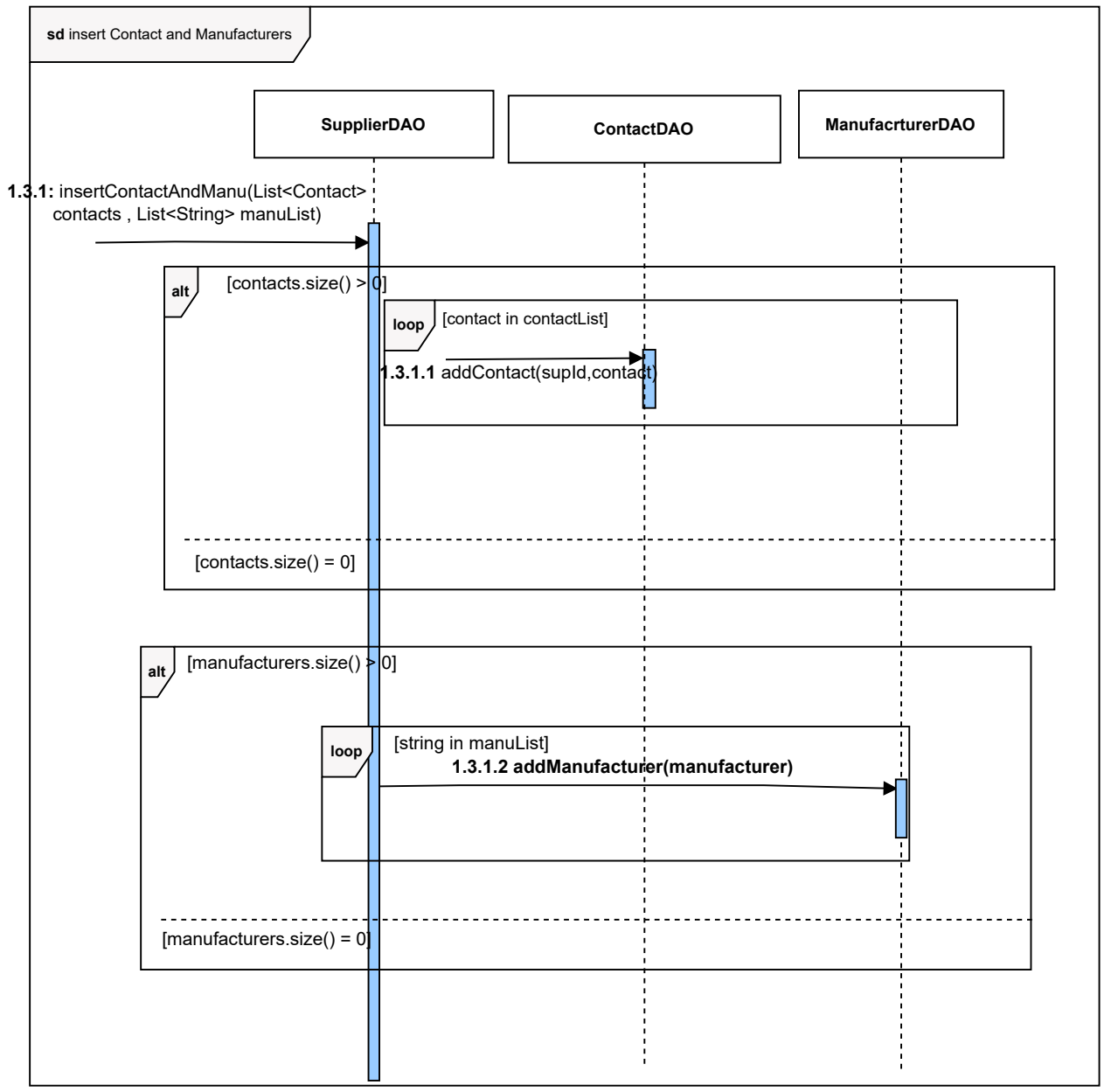


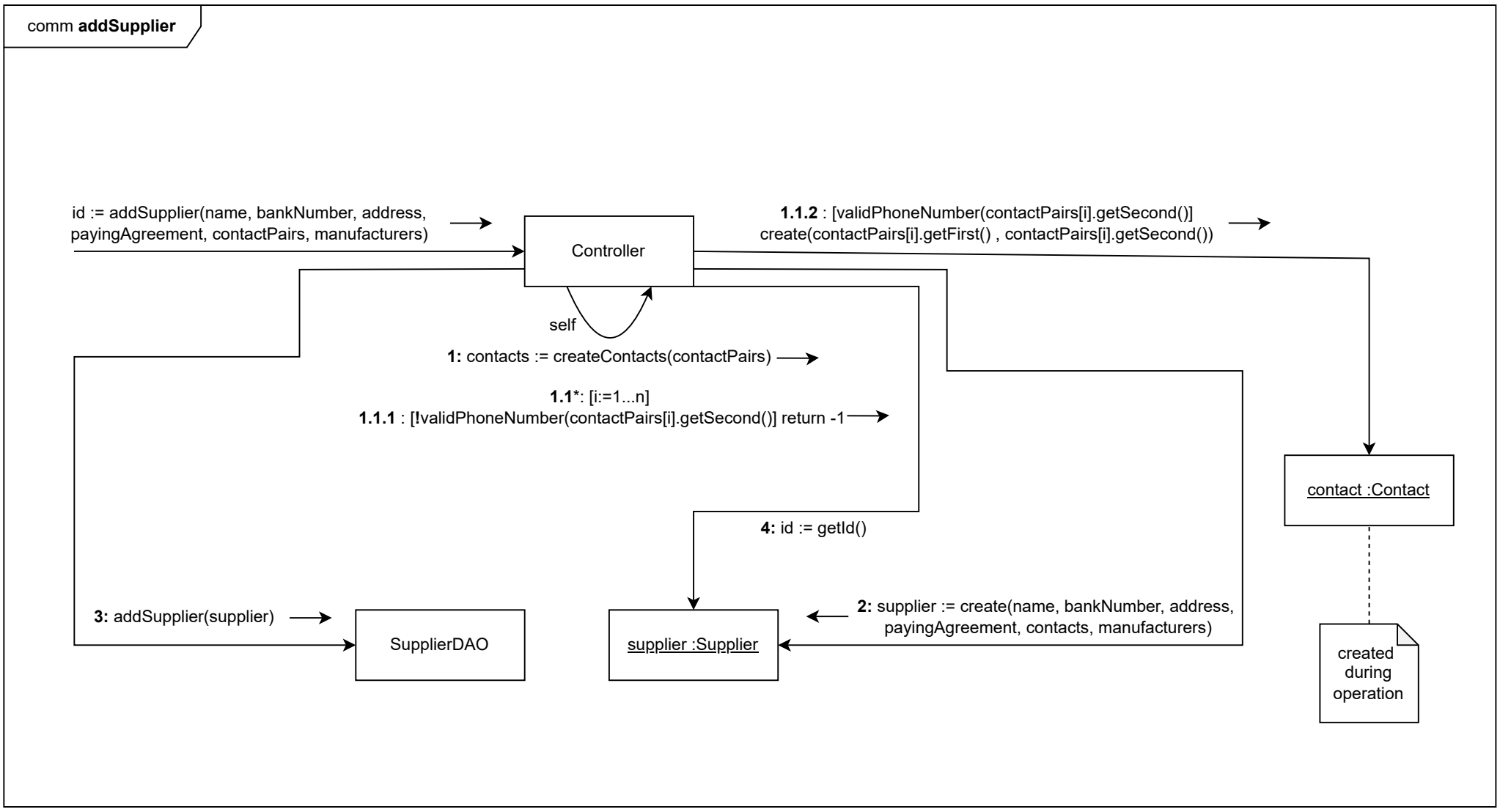

comm **Return Items**
returnItems(store, product, amount, dateOfPurchase)
InvetoryController
2: priceToReturn:= retunItems(store, amount, dateOfPurchase)
1: product := getProduct(productID)
StockReportDAO
6: returnItems(amount)
4: saleOnDate:=
getSaleOnDate(dateOfPurchase)
stockReport:
StockReport
product :Product
cat: Category
7: updateStore
(product, store, amountInStore)
3: saleOnDate:= getSaleOnDate(dateOfPurchase)
5: stockReport:= getStockReport()





When adding a Supplier to the system, the user doesn't have to add agreement, and in fact adding agreement is a different and seperated use case, so we didn't write this here


sd insert Contact and Manufacturers
SupplierDAO
ContactDAO
ManufacrturerDAO
1.3.1: insertContactAndManu(List<Contact>
contacts , List<String> manuList)
alt
[contacts.size() > 0]
loop
[contact in contactList]
1.3.1.1 addContact(supId,contact)
[contacts.size() = 0]
alt
[manufacturers.size() > 0]
loop
[string in manuList]
1.3.1.2 addManufacturer(manufacturer)
[manufacturers.size() = 0]


comm addSupplier
id := addSupplier(name, bankNumber, address,
payingAgreement, contactPairs, manufacturers)
Controller
1.1.2 : [validPhoneNumber(contactPairs[i].getSecond())]
create(contactPairs[i].getFirst() , contactPairs[i].getSecond())
self
1: contacts := createContacts(contactPairs)
1.1*: [i:=1...n]
1.1.1 : [!validPhoneNumber(contactPairs[i].getSecond()] return -1
contact :Contact
4: id := getId()
3: addSupplier(supplier)
SupplierDAO
supplier :Supplier
2: supplier := create(name, bankNumber, address,
payingAgreement, contacts, manufacturers)
created
during
operation