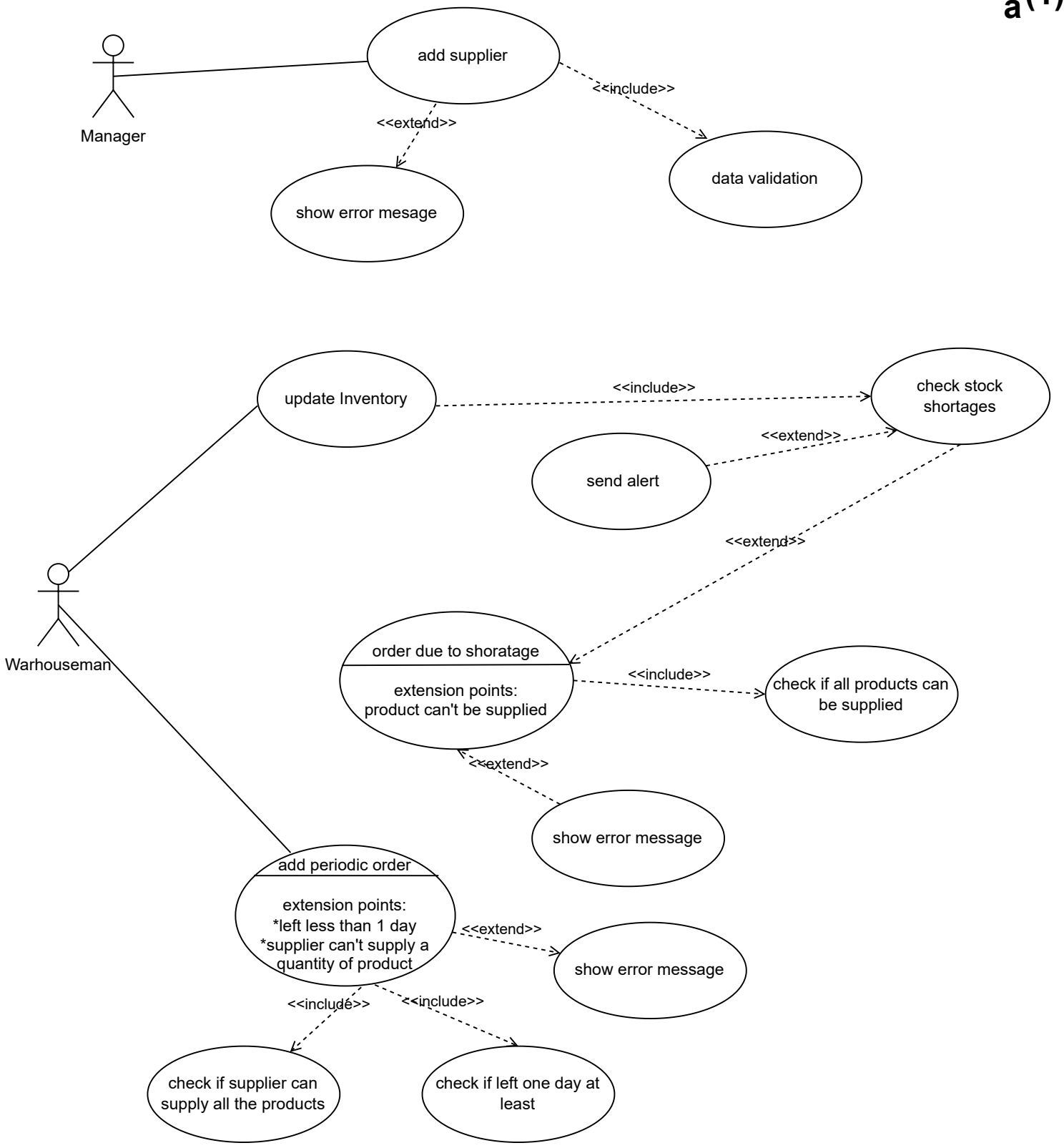
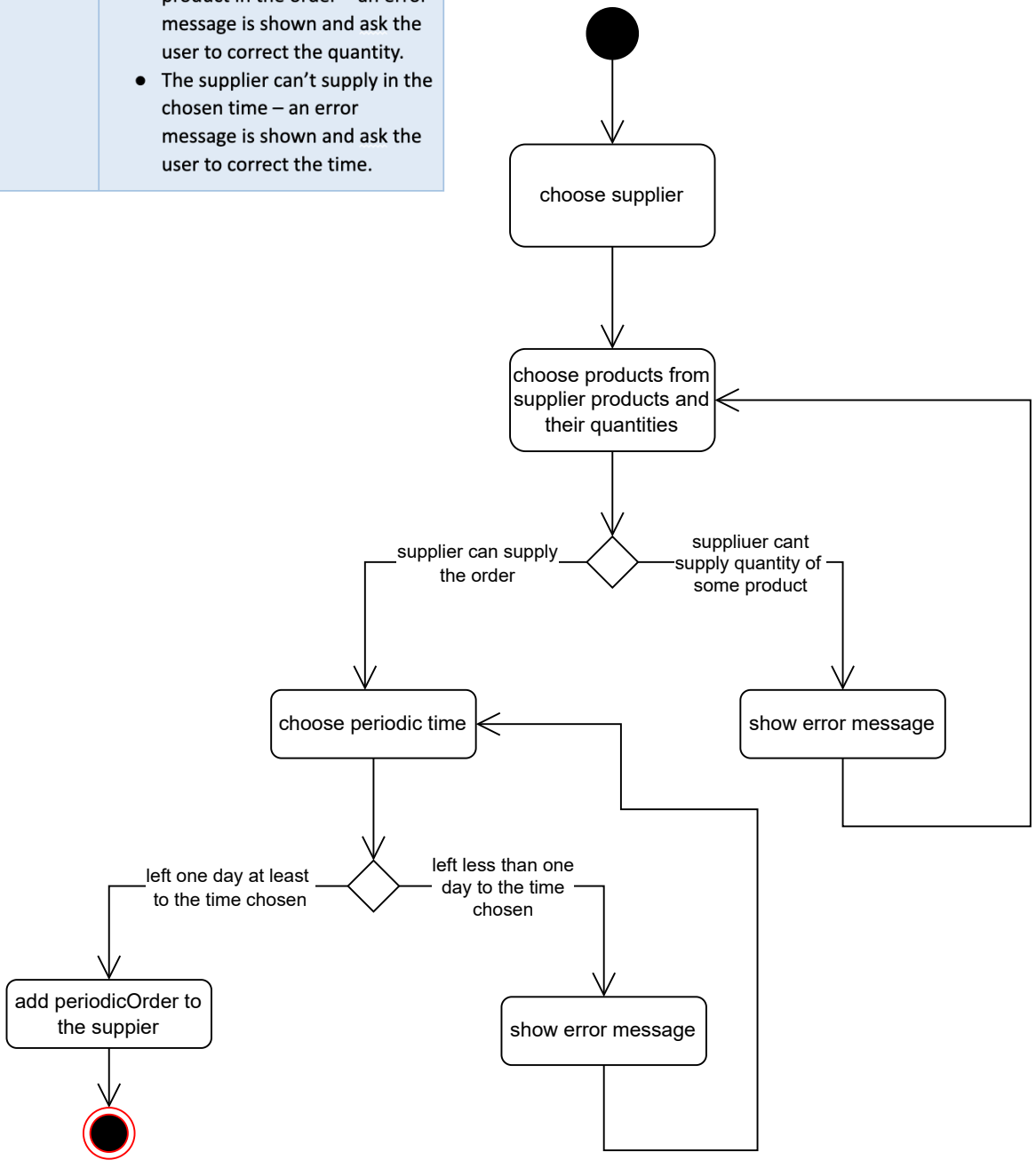
a(1)


Manager
add supplier
<<include>>
<<extend>>
show error mesage
data validation
update Inventory
<<include>>
check stock shortages
<<extend>>
send alert
<<extend>>
Warhouseman
order due to shoratage
extension points:
product can't be supplied
<<include>>
check if all products can be supplied
<<extend>>
show error message
add periodic order
extension points:
*left less than 1 day
*supplier can't supply a quantity of product
<<extend>>
show error message
<<include>>
<<include>>
check if supplier can supply all the products
check if left one day at least

Use case name	Periodic order
Textual description	The warehouseman selects the periodic time, and a supplier and chooses what products and their quantities to order from him
Actor	warehouseman
Pre conditions	1. The Application is initialized properly
Post conditions	A periodic order is set with a specific supplier.
Main success scenario	1. Warehouseman chooses a supplier to periodically order from. 2. Warehouseman chooses the products from the products the supplier supplies and their quantities to order. 3. Warehouseman chooses a periodic time for the order 4. A periodic order is sent to the supplier
Alternatives/Extensions	• The supplier can't supply the quantity required of a specific product in the order – an error message is shown and ask the user to correct the quantity. • The supplier can't supply in the chosen time – an error message is shown and ask the user to correct the time.

Activity diagram



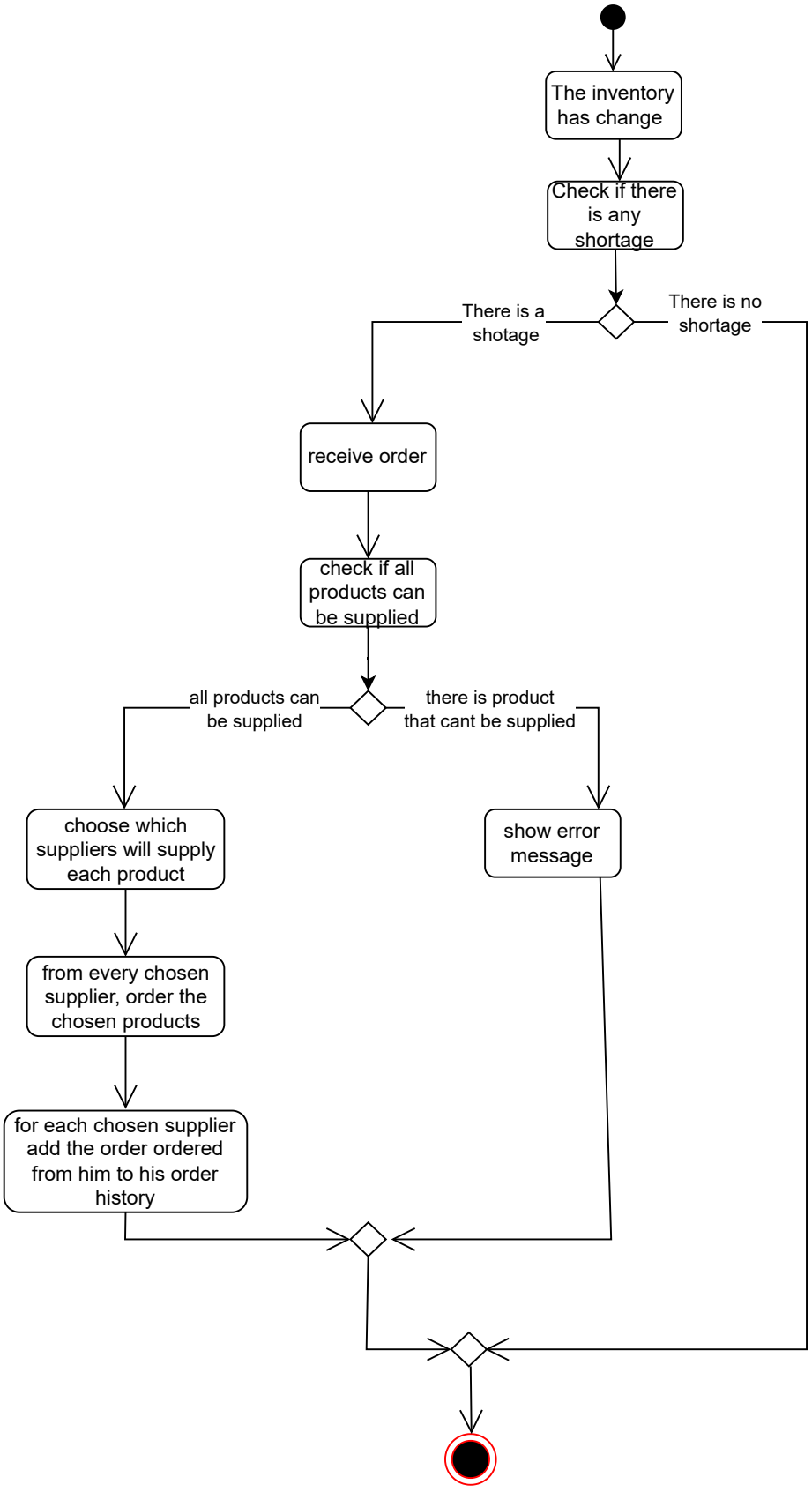
Use case name	update Inventory
Textual description	The warehouseman reports on inventory changes.
Actor	Warehouseman
Preconditions	Product needs to be initialized in the system.
Post conditions	Product changes update in the system on the relevant filed.
Main success scenario	1. The warehouseman reports on inventory changes. 2. The system updates the chosen products.
Alternatives/Extensions	• The product does not exist in the system.

Use case name	check stock shortages
Textual description	The system checks if there are shortages on specific product
Actor	Inventory system
Preconditions	Change accrued in the inventory
Post conditions	The system run “order due to shortage” operation.
Main success scenario	1. The system check in the inventory if the number of products in the store and the warehouse is less than the minimum. 2. A shortage alert appears in the system and update about order request that been sent
Alternatives/Extensions	There is no shortage, and no order is sent.

Use case name	order due to shortage
Textual description	The system sends an order of the shortages products to the specific supplier that it chooses.
Actor	Inventory system
Preconditions	Change accrued in the inventory
Post conditions	An order is sent by the inventory system to the supplier system.
Main success scenario	The inventory system sends the shortages products as an order to the supplier system.
Alternatives/Extensions	• There is no shortage, and no order is sent.

ACTIVITY DIAGRAM

Use case - F



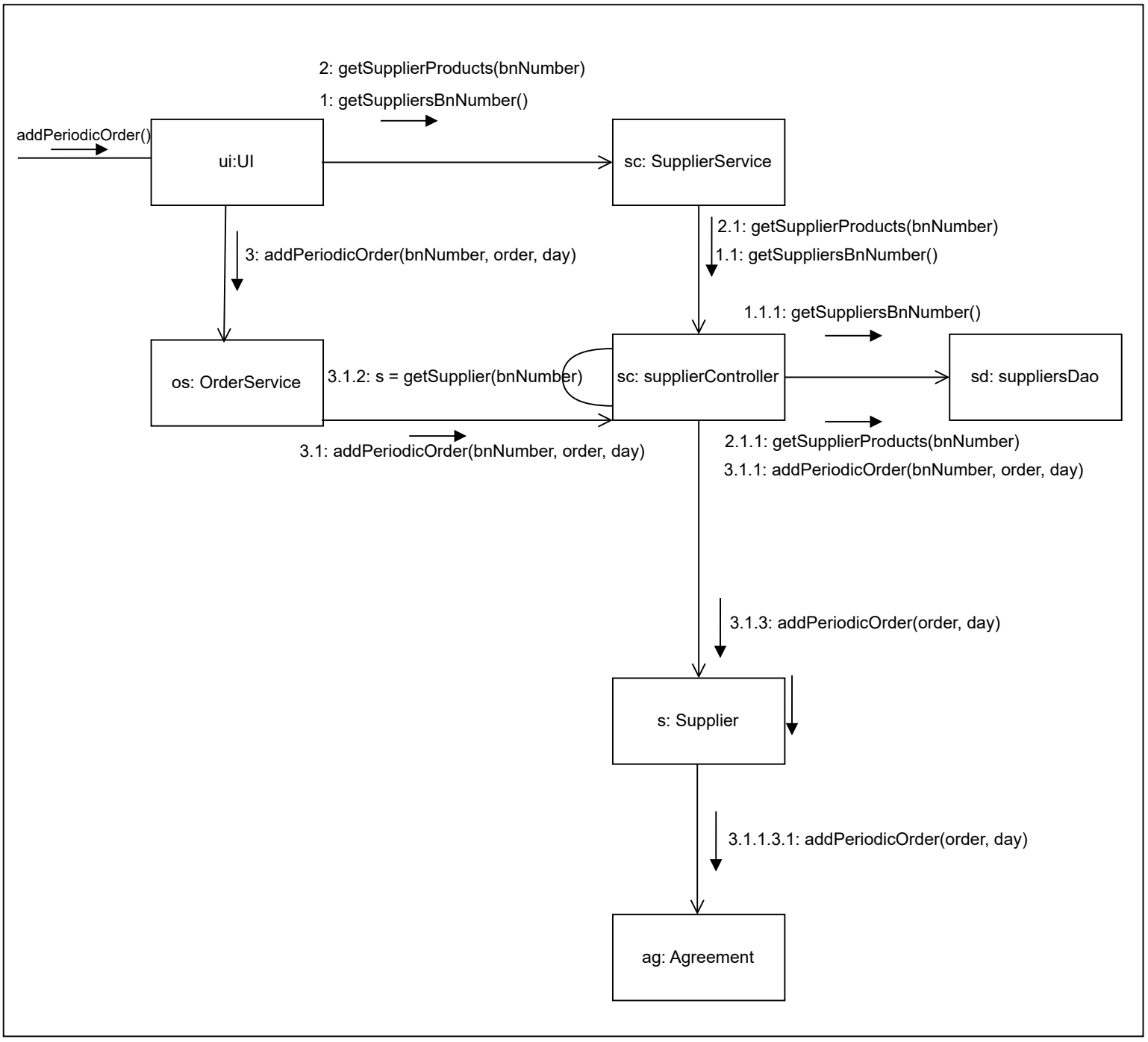
Operation: addPeriodicOrder(Supplier s, products:Map<Integer, Integer>, day)
References : Cases: add Periodic order.
Pre conditions : the system initialized properly.
Post condition :

- a PeriodicSale instance sp was created
- the supplier mentioned was assosiated with sp
- sp.order became products
- sp.day became day

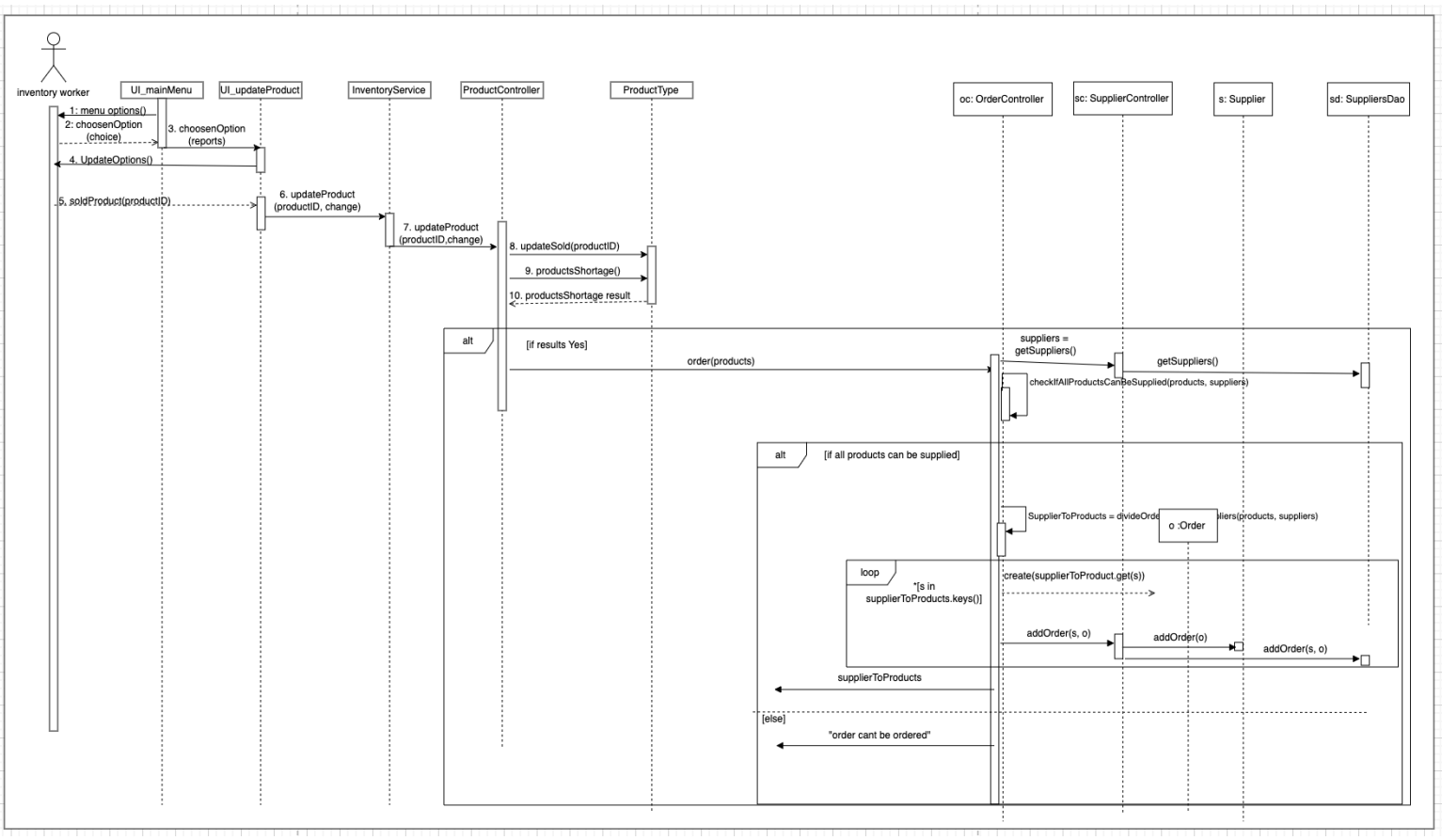
Operation: order(products: Map<Integer,Integer>)
References : Cases: Order.
Pre conditions : There is a shortage in products to order.
Post condition :

- for every supplier chosen, an order is sent and transport is ordered if needed
- for every supplier chosen, an Order instance was created and associated with the supplier

Use case - E **b**


addPeriodicOrder()
ui:UI
2: getSupplierProducts(bnNumber)
1: getSuppliersBnNumber()
sc: SupplierService
3: addPeriodicOrder(bnNumber, order, day)
2.1: getSupplierProducts(bnNumber)
1.1: getSuppliersBnNumber()
1.1.1: getSuppliersBnNumber()
os: OrderService
3.1.2: s = getSupplier(bnNumber)
sc: supplierController
sd: suppliersDao
3.1: addPeriodicOrder(bnNumber, order, day)
2.1.1: getSupplierProducts(bnNumber)
3.1.1: addPeriodicOrder(bnNumber, order, day)
3.1.3: addPeriodicOrder(order, day)
s: Supplier
3.1.1.3.1: addPeriodicOrder(order, day)
ag: Agreement

Use case - F


inventory worker
UI_mainMenu
UI_updateProduct
InventoryService
ProductController
ProductType
oc: OrderController
sc: SupplierController
s: Supplier
sd: SuppliersDao
1: menu options()
2: choosenOption (choice)
3. choosenOption (reports)
4. UpdateOptions()
5. soldProduct(productID)
6. updateProduct (productID, change)
7. updateProduct (productID,change)
8. updateSold(productID)
9. productsShortage()
10. productsShortage result
alt
[if results Yes]
order(products)
suppliers = getSuppliers()
getSuppliers()
checkIfAllProductsCanBeSupplied(products, suppliers)
alt
[if all products can be supplied]
SupplierToProducts = divideOrde
liers(products, suppliers)
o :Order
loop
"[s in supplierToProducts.keys()]
create(supplierToProduct.get(s))
addOrder(s, o)
addOrder(o)
addOrder(s, o)
supplierToProducts
[else]
"order cant be ordered"