

RMCode: Automatic Prototyping from Requirement Model

Yilong Yang

Software Engineering Lab

<http://lab.mydreamy.net>

yylonly@gmail.com

Preliminary

Requirement Model

- Use Case Diagram
- System Sequence Diagram / Activity Diagram
- Conceptual Class Diagram
- Operation Contract

Requirement Model

- The responsibilities of the target system includes knowing responsibilities and doing responsibilities.
- **Knowing responsibilities** of the target system are specified by the **conceptual class diagram**.
- **Doing responsibilities** are specified by **activity diagrams** or **system sequence diagrams** for use cases in the **use case diagram**.
- **Contract** of doing responsibility not only contains the **signatures of operation** but also **pre and post conditions** of operation for fulfilling the responsibility

Motivation

- Requirement errors mostly lead to the failures of software development.
- Customers and end-user are not entirely sure of what is needed before trying out the target software.
- It is very desirable to have a tool to generate prototypes directly from requirements automatically.

State-of-arts

- Current UML modeling tools can only generate skeleton code, the operations still need to be manually implemented.
- Even if providing a design model to each operation, only less than 48% correct source code can be generated.

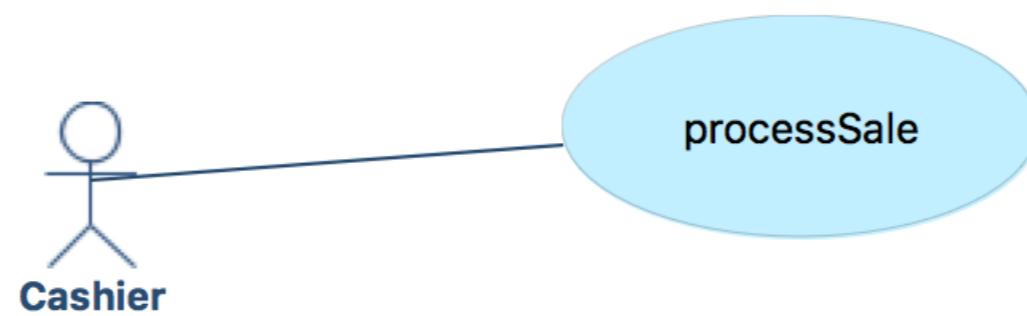
RMCode: Requirement Modeler and Code Generator

- RMCode can automatically generate object-oriented prototype from UML requirement model.
- Generated prototype can be used for validating the requirements and eliciting more requirements.

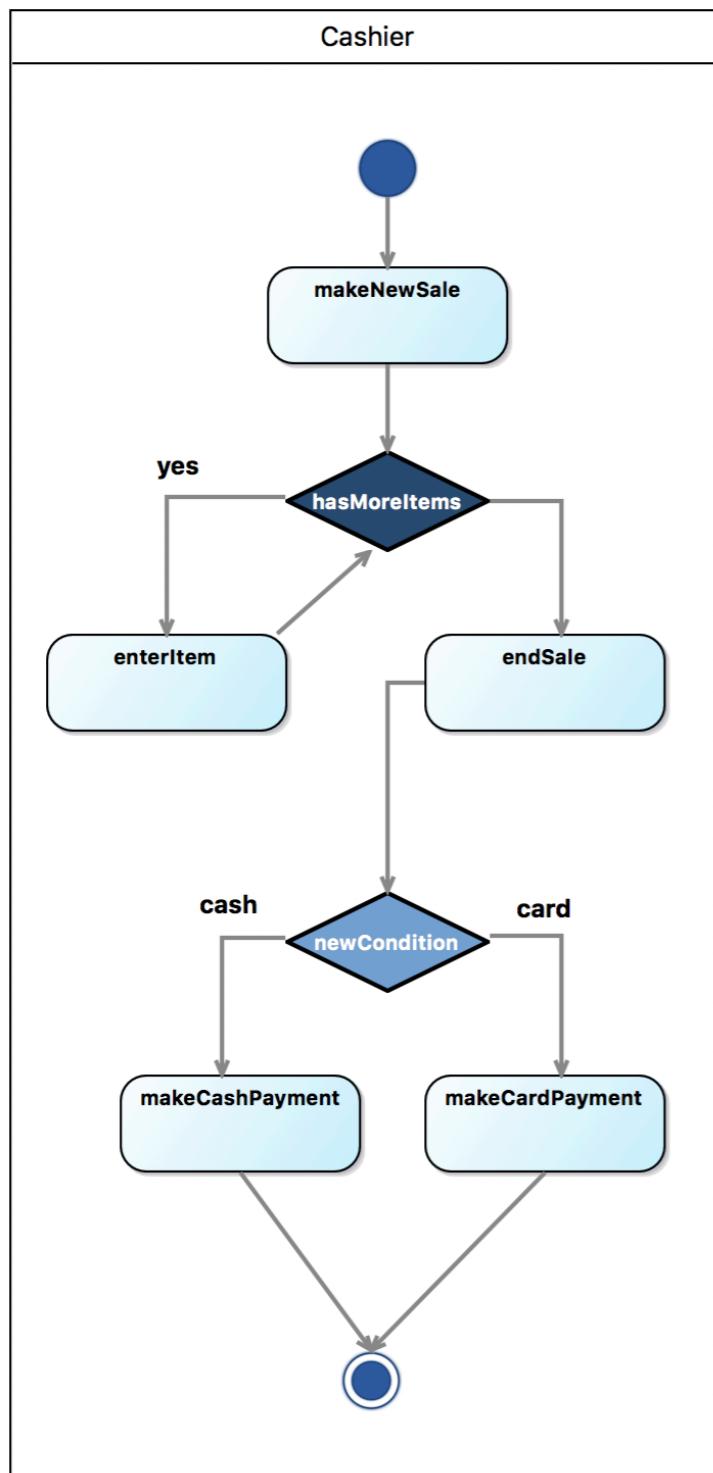
Case Study I

CoCoME (Supermarket System)

Use Case Diagram



System Interactions

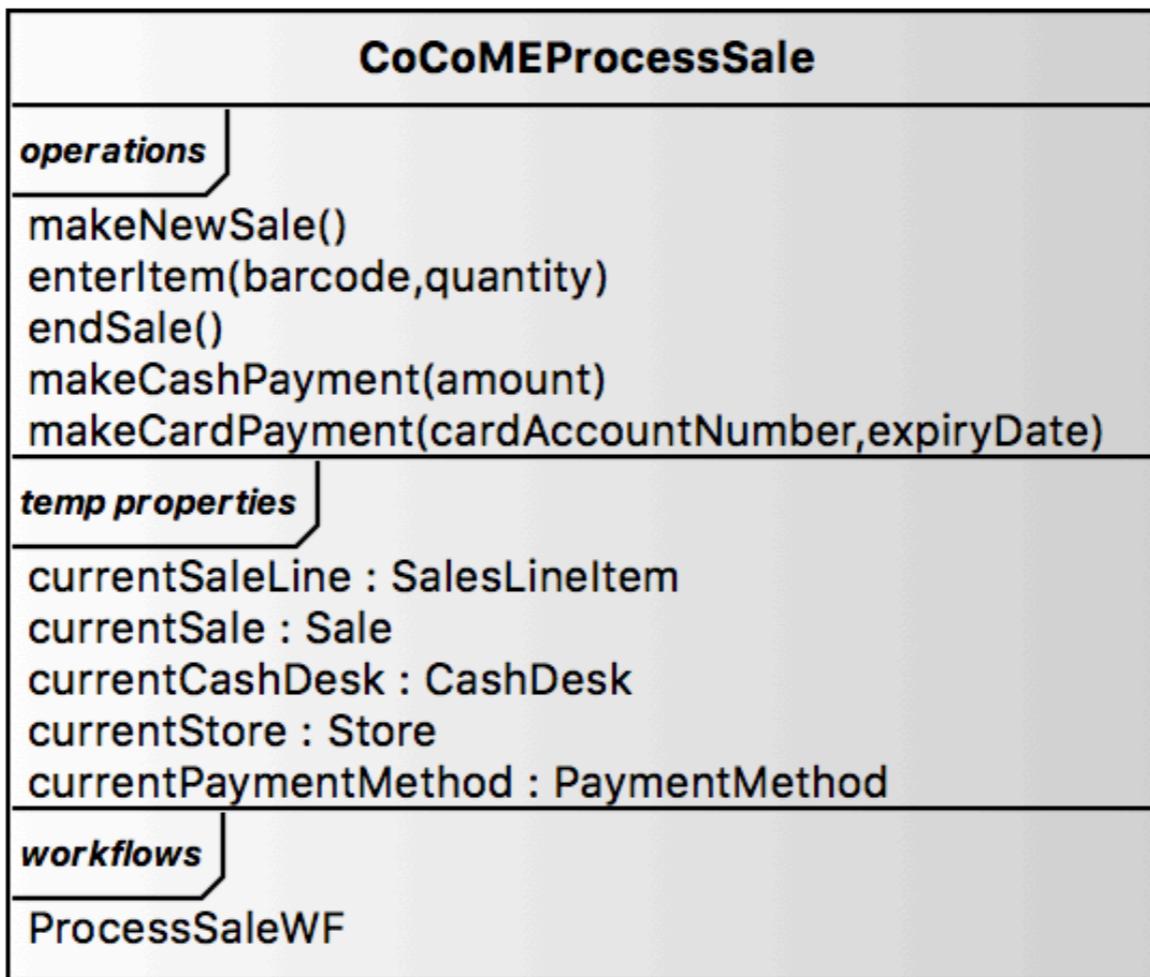


Activity Diagram

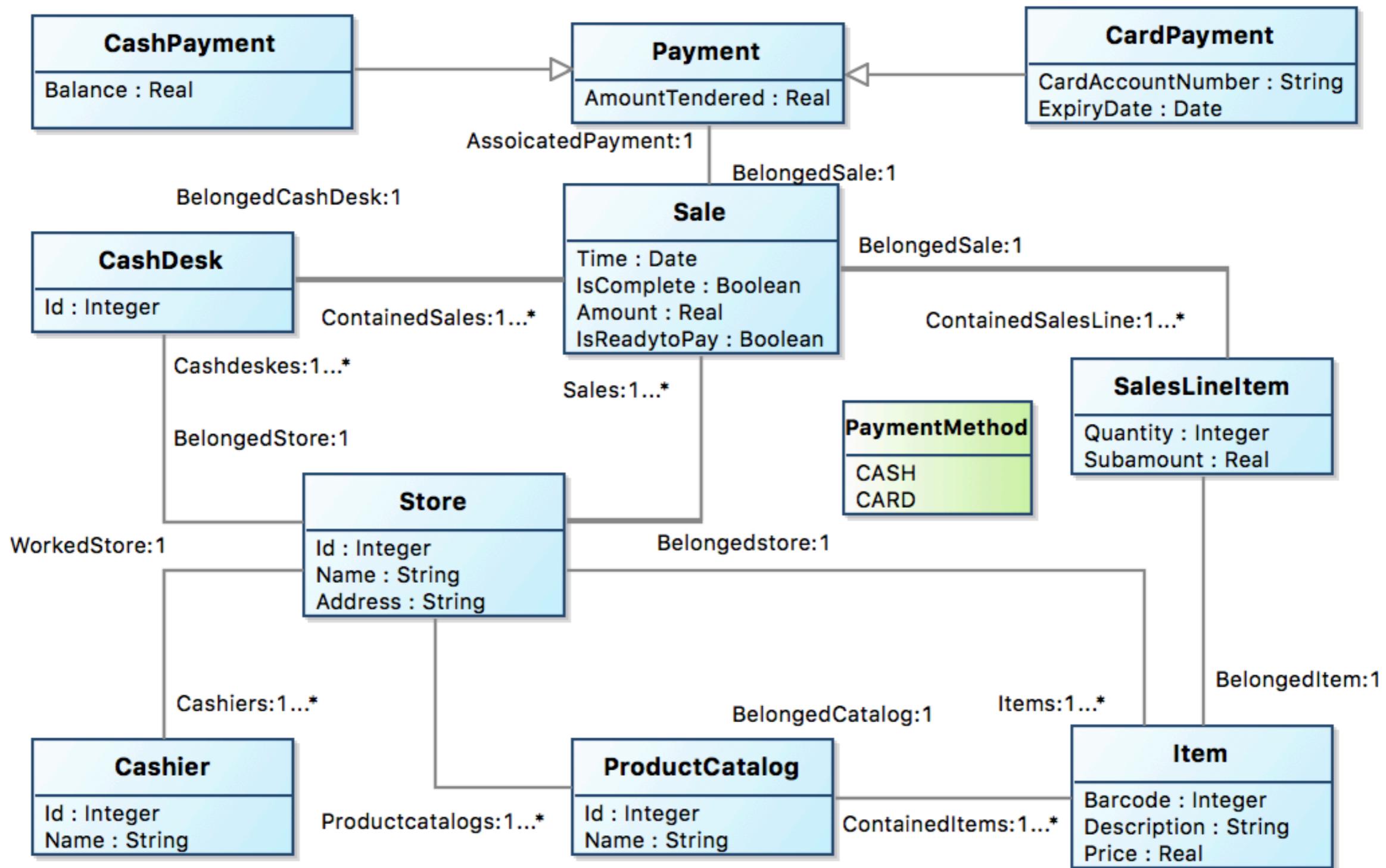


System Sequence Diagram

Doing responsibilities



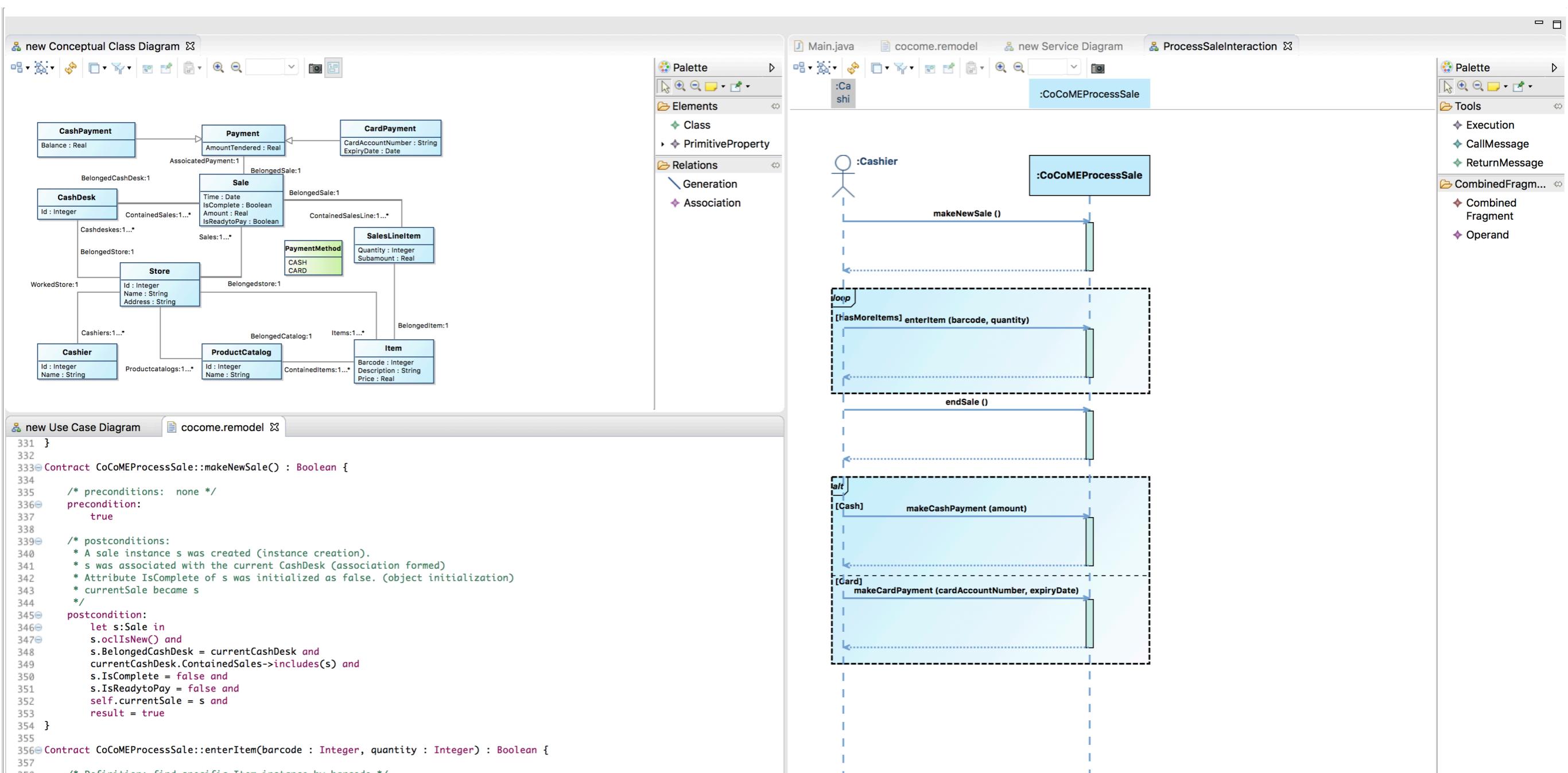
Conceptual Class Diagram



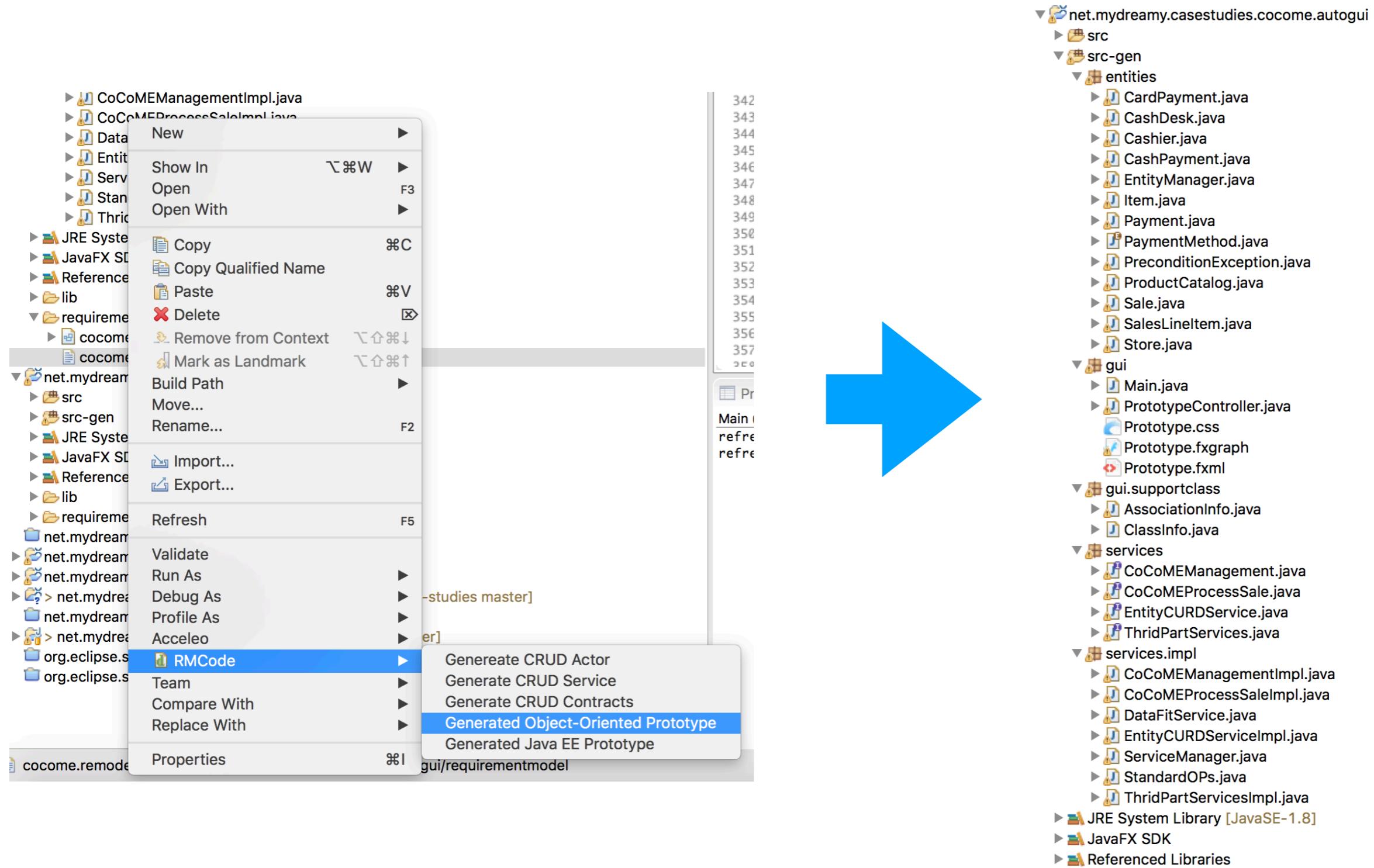
Operation Contract

```
Contract CoCoMEProcessSale::makeNewSale() : Boolean {  
  
    /* preconditions: none */  
    precondition:  
        true  
  
    /* postconditions: [] */  
    postcondition:  
        let s:Sale in  
            s.oclIsNew() and  
            s.BelongedCashDesk = currentCashDesk and  
            currentCashDesk.ContainedSales->includes(s) and  
            s.IsComplete = false and  
            s.IsReadytoPay = false and  
            self.currentSale = s and  
            result = true  
}
```

Requirement Model in RMCode



Generate Prototype



Prototype: System Operations

Prototype Cocomore

System Function		System Status	
▼ Cashier		Operation Parameters	
▼ processSale		barcode:	<input type="text"/>
makeNewSale		quantity:	<input type="text"/>
enterItem		Operation Return	
endSale			
makeCashPayment			
makeCardPayment			
System Log			
► Administrator	Execute		Reset

Definition

```
item:Item = Item.allInstances()->any(i:Item | i.Barcode = barcode)
```

Precondition

```
currentSale.IsComplete = false
```

Postcondition

```
let sli:SalesLineItem in  
sli.ocllsNew() and  
self.currentSaleLine = sli and  
sli.BelongedSale = currentSale and  
currentSale.ContainedSalesLine->includes(sli) and  
sli.Quantity = quantity and  
sli.BelongedItem = item and  
sli.BelongedLine = currentSaleLine
```

Invariants

Generated By RMCode

Prototype: System Status

Prototype Cocomore

System Function System Status

Class statistics

Class Name	# of Objects
Store	0
ProductCatalog	0
CashDesk	0
Sale	0
Cashier	0
SalesLineItem	0
Item	0

Object Statistics

All Invariants

Association statistics

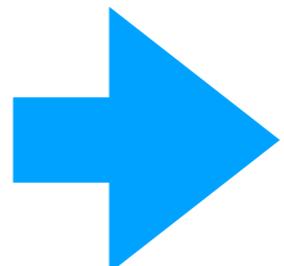
Source Class	Association Name	Target Class	Multiple	Association Number
No content in table				

Load Status Save Status Refresh Status Check All Invariants

Generate CRUD Operations

The screenshot shows the 'Contract CoCoME Management' interface with a context menu open over some code. The menu includes options like Undo Typing, Save, Open Declaration, and RMCode. The RMCode submenu is expanded, showing options: Generate CRUD Actor, Generate CRUD Service, Generate CRUD Contracts (which is highlighted), Generated Object-Oriented Prototype, and Generated Java EE Prototype.

```
15 //Contract CoCoMEManagement::createItem(item : item, catalogId : integer) : Boolean {  
16 //  
17 // Undo Typing  
18 // Revert File  
19 // Save  
20 //  
21 // Open Declaration F3  
22 // Open Generated File  
23 // Quick Outline ⌘O  
24 // Open With ►  
25 // Show In ▾⌘W ►  
26 //  
27 // Cut ⌘X  
28 // Copy ⌘C  
29 // Copy Qualified Name  
30 // Paste ⌘V  
31 //  
32 // Rename Element ▾⌘R  
33 // Validate  
34 // Quick Fix ⌘I  
35 // Source ►  
36 // Find References ▲⌘G  
37 // Add to Snippets...  
38 // Run As ►  
39 // Debug As ►  
40 // Profile As ►  
41 // Validate  
42 // RMCode ►  
43 // Team ►  
44 // Compare With ►  
45 // Replace With ►  
46 // Preferences...  
47 // Remove from Context ▾⇧⌘↓
```



EntityCURDService	
<i>operations</i>	
createStore(id,name,address)	
queryStore(Id)	
modifyStore(id,name,address)	
deleteStore(Id)	
createProductCatalog(id,name)	
queryProductCatalog(Id)	
modifyProductCatalog(id,name)	
deleteProductCatalog(Id)	
createCashDesk(id)	
queryCashDesk(Id)	
modifyCashDesk(id)	
deleteCashDesk(Id)	
createCashier(id,name)	
queryCashier(Id)	
modifyCashier(id,name)	
deleteCashier(Id)	
createItem(barcode,description,price)	
queryItem(Barcode)	
modifyItem(barcode,description,price)	
deleteItem(Barcode)	

Loading or adding start-up data

Prototype Cocomo

System Function		System Status	
► Cashier		Operation Parameters	
▼ Administrator		Definition	
createStore		store:Store = Store.allInstance()->new()	
queryStore		Precondition	
modifyStore		store.ocllsUndefined() = true	
deleteStore		Postcondition	
createProductCatalog		let sto:Store in sto.ocllsNew() and sto.Id = id and sto.Name = name and sto.Address = address and Store.allInstance()->includes(sto) a result = true	
queryProductCatalog		Invariants	
modifyProductCatalog			
deleteProductCatalog			
createCashDesk			
queryCashDesk			
modifyCashDesk			
deleteCashDesk			
createCashier			
queryCashier			
modifyCashier			
deleteCashier			
createItem			
queryItem			
modifyItem			
deleteItem			
		Execute	Reset

Generated By RMCode

Start-up Data: UM Store

Prototype Cocomore

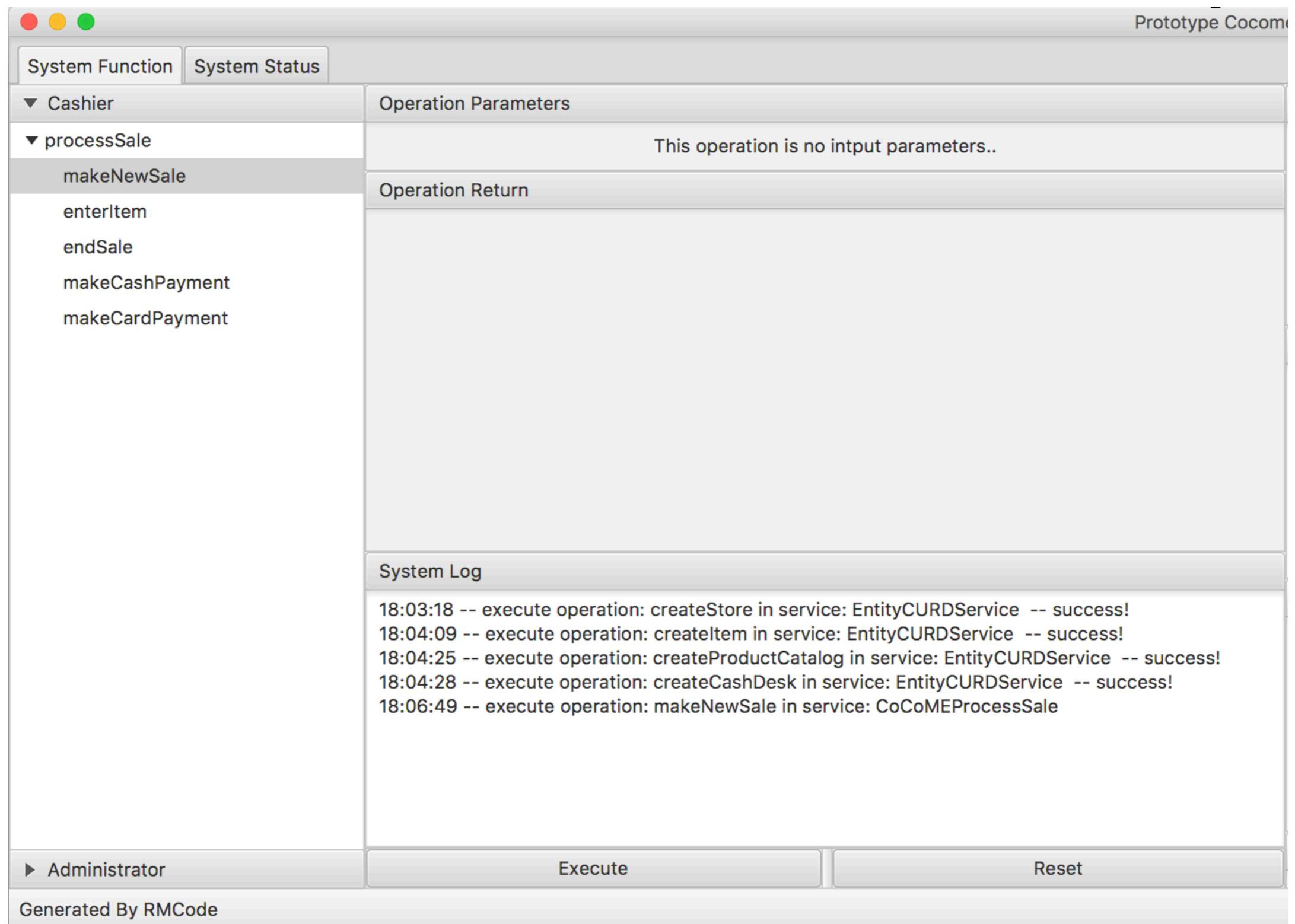
System Function System Status

Class statistics		All Objects Store:			All Invariants
Class Name	# of Objects	Id	Name	Address	
Store	1	1	UMStore	Taipa	
ProductCatalog	0				
CashDesk	0				
Sale	0				
Cashier	0				
SalesLineItem	0				
Item	0				
Payment	0				
Association statistics					
Source Class	Association Name	Target Class	Multiple	Association Number	
Store	Cashdeskes	CashDesk	true	0	
Store	Productcatalogs	ProductCatalog	true	0	
Store	Items	Item	true	0	
Store	Cashiers	Cashier	true	0	
Store	Sales	Sale	true	0	
Load Status	Save Status	Refresh Status	Check All Invariants		

Start-up data: Item Apple

System Function		System Status		Prototype Cocome			
Class statistics				All Objects Item:			All Invariants
Class Name		# of Objects		Barcode	Description	Price	
Store		1		1	Apple	10.0	
ProductCatalog		1					
CashDesk		1					
Sale		0					
Cashier		0					
SalesLineItem		0					
Item		1					
Payment		0					
< >							
Association statistics							
Source Class	Association Name	Target Class	Multiple	Association Number			
Item	BelongedCatalog	ProductCatalog	false	0			
Load Status	Save Status	Refresh Status	Check All Invariants				

Validate ProcessSale



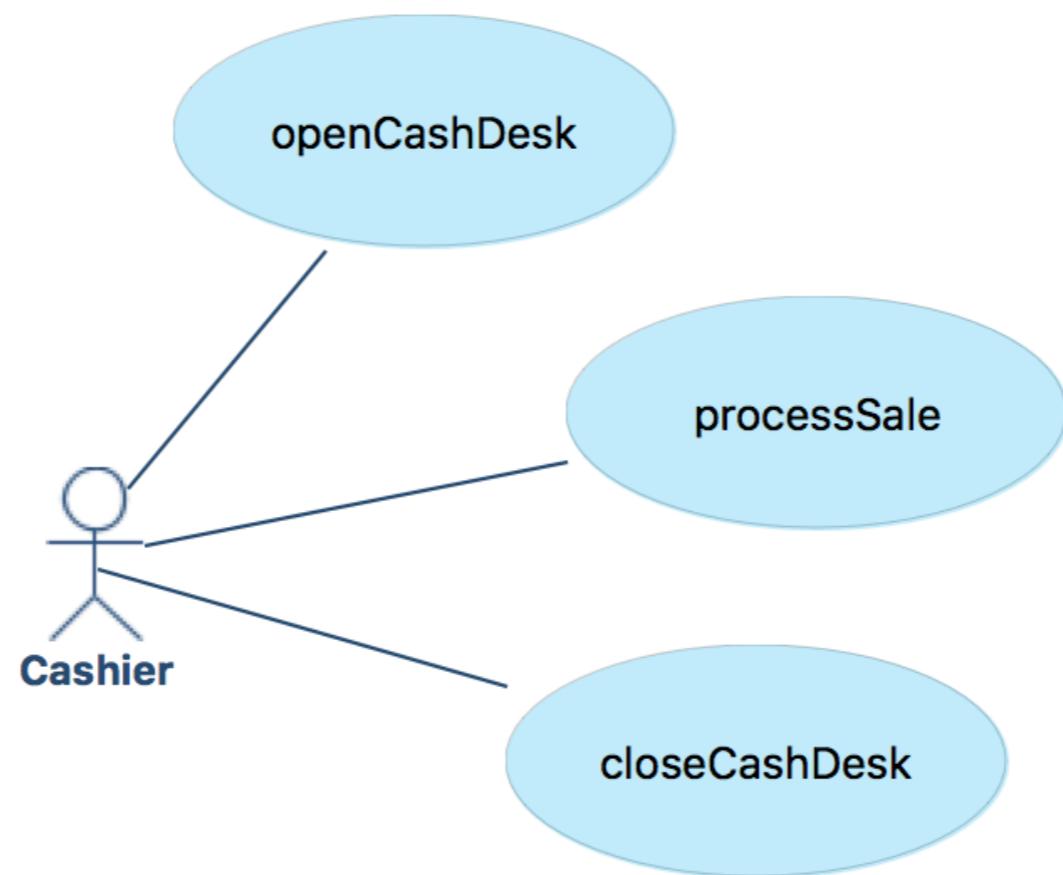
Validate ProcessSale

```
Exception in thread "JavaFX Application Thread" java.lang.RuntimeException: java.lang.reflect.InvocationTargetException
  at javafx.fxml.FXMLLoader$MethodHandler.invoke(FXMLLoader.java:1770)
  at javafx.fxml.FXMLLoader$ControllerMethodEventHandler.handle(FXMLLoader.java:1653)
  at com.sun.javafx.event.CompositeEventHandler.dispatchBubblingEvent(CompositeEventHandler.java:86)
  at com.sun.javafx.event.EventHandlerManager.dispatchBubblingEvent(EventHandlerManager.java:238)
  at com.sun.javafx.event.EventHandlerManager.dispatchBubblingEvent(EventHandlerManager.java:191)
  at com.sun.javafx.event.CompositeEventDispatcher.dispatchBubblingEvent(CompositeEventDispatcher.java:59)
  at com.sun.javafx.event.BasicEventDispatcher.dispatchEvent(BasicEventDispatcher.java:58)
  at com.sun.javafx.event.EventDispatchChainImpl.dispatchEvent(EventDispatchChainImpl.java:114)
  at com.sun.javafx.event.BasicEventDispatcher.dispatchEvent(BasicEventDispatcher.java:56)
  at com.sun.javafx.event.EventDispatchChainImpl.dispatchEvent(EventDispatchChainImpl.java:114)
  at com.sun.javafx.event.BasicEventDispatcher.dispatchEvent(BasicEventDispatcher.java:56)
```

Postcondition

```
let s:Sale in
s.ocllsNew() and
s.BelongedCashDesk = currentCashDesk and
currentCashDesk.ContainedSales->includes(s) and
s.IsComplete = false and
s.IsReadytoPay = false and
self.currentSale = s and
result = true
```

Refined Requirement Model



OpenCashDesk

Prototype Cocomo

System Function	System Status
Cashier <ul style="list-style-type: none"> processSale makeNewSale enterItem endSale makeCashPayment makeCardPayment openCashDesk closeCashDesk 	<p>Operation Parameters</p> <p>cashDeskID: <input type="text" value="1"/></p> <p>Operation Return</p> <p>true</p> <p>System Log</p> <pre>18:15:41 -- execute operation: createStore in service: EntityCURDService -- success 18:15:51 -- execute operation: createProductCatalog in service: EntityCURDService -- success 18:16:25 -- execute operation: createCashDesk in service: EntityCURDService -- success 18:17:31 -- execute operation: openStore in service: CoCoMEProcessSale -- success 18:17:37 -- execute operation: openCashDesk in service: CoCoMEProcessSale -- success</pre> <p>Definition</p> <pre>cd:CashDesk = CashDesk.allInstances()->any(s:CashDesk s.id = cashDeskID)</pre> <p>Precondition</p> <pre>cd.oclIsUndefined() = false and cd.isOpen = false and currentStore.oclIsUndefined() = false and currentStore.isOpen = true</pre> <p>Postcondition</p> <pre>self.currentCashDesk = cd and cd.isOpen = true and result = true</pre> <p>Invariants</p> <p>CashDesk_UniqueCashDeskId</p>
Administrator	<input type="button" value="Execute"/> <input type="button" value="Reset"/>
StoreManager	

Generated By RMCode

CashDesk1 Status:

All Objects CashDesk:		
Id	Name	IsOpened
1	CashDesk1	true

ProcessSale - makeNewSale

Prototype Cocomo

System Function	System Status	
▼ Cashier	Operation Parameters	Definition
▼ processSale	This operation is no intput parameters..	
makeNewSale	Operation Return	Precondition
enterItem		currentCashDesk.oclIsUndefined() = currentCashDesk.IsOpened = true and (currentSale.oclIsUndefined() = true (currentSale.oclIsUndefined() = currentSale.IsComplete = true)
endSale		
makeCashPayment		
makeCardPayment		
openCashDesk		
closeCashDesk		
	System Log	Postcondition
	18:15:41 -- execute operation: createStore in service: EntityCURDService -- success! 18:15:51 -- execute operation: createProductCatalog in service: EntityCURDService -- success! 18:16:25 -- execute operation: createCashDesk in service: EntityCURDService -- success! 18:17:31 -- execute operation: openStore in service: CoCoMEProcessSale -- success! 18:17:37 -- execute operation: openCashDesk in service: CoCoMEProcessSale -- success! 18:18:28 -- execute operation: makeNewSale in service: CoCoMEProcessSale -- success!	let s:Sale in s.oclIsNew() and s.BelongedCashDesk = currentCashDesk currentCashDesk.ContainerSales->in s.IsComplete = false and s.IsPaid = false and
		Invariants
		Sale_AmountGreatAndEqualZero
▶ Administrator		
▶ StoreManager	Execute	Reset
Generated By RMCode		

Class statistics		
Class Name	# of Objects	
Store	1	
ProductCatalog	1	
CashDesk	1	
Sale	1	
Cashier	0	

ProcessSale - enterItem

Prototype Cocomo

System Function	System Status	
▼ Cashier	Operation Parameters	Definition
▼ processSale	barcode: 1	item:Item = Item.allInstances()->any(i:Item i.Barcode
makeNewSale	quantity: 2	
enterItem	Operation Return	Precondition
endSale	true	currentSale.oclIsUndefined() = false and currentSale.IsComplete = false
makeCashPayment		
makeCardPayment		
openCashDesk		
closeCashDesk		
	System Log	Postcondition
	18:15:41 -- execute operation: createStore in service: EntityCURDService -- success! 18:15:51 -- execute operation: createProductCatalog in service: EntityCURDService -- success! 18:16:25 -- execute operation: createCashDesk in service: EntityCURDService -- success! 18:17:31 -- execute operation: openStore in service: CoCoMEProcessSale -- success! 18:17:37 -- execute operation: openCashDesk in service: CoCoMEProcessSale -- success! 18:18:28 -- execute operation: makeNewSale in service: CoCoMEProcessSale -- success! 18:18:59 -- execute operation: createItem in service: EntityCURDService -- success! 18:19:16 -- execute operation: enterItem in service: CoCoMEProcessSale -- success!	let sli:SalesLineItem in sli.oclIsNew() and self.currentSaleLine = sli and sli.BelongedSale = currentSale and currentSale.ContainerSalesLine->includes(sli) and
		Invariants
		Item_UniqueBarcode
		Item_PriceGreaterThanOrEqualToZero
		Item_StockNumberGreaterThanOrEqualToZero
▶ Administrator		
▶ StoreManager	Execute	
Generated By RMCode	Reset	

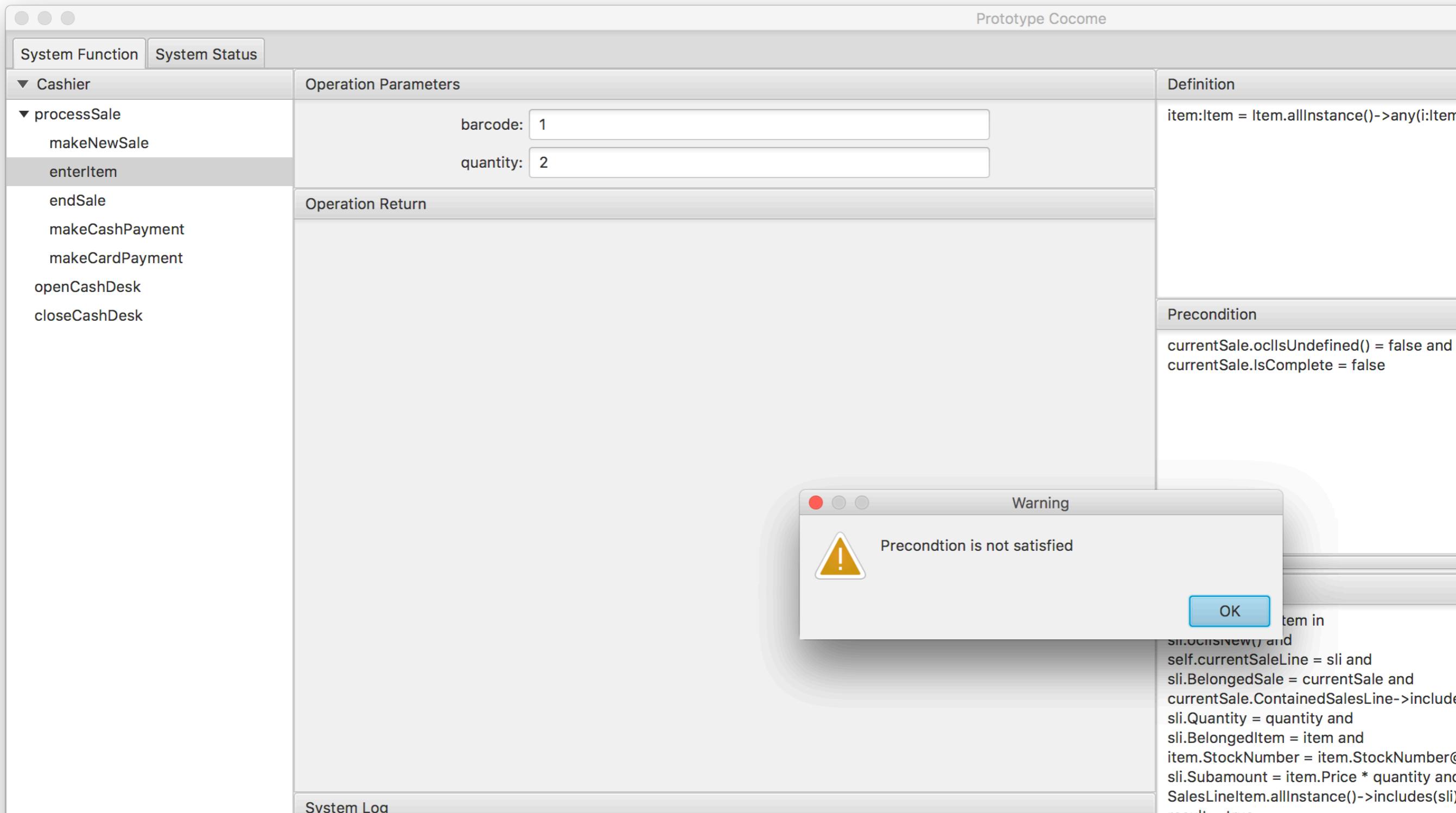
Input Variables:

Operation Parameters
barcode: 1
quantity: 2

ProcessSale - enterItem

System Function	System Status	All Objects SalesLineItem:		
Class statistics		# of Objects	Quantity	Subamount
	Store	1	2	20.0
	ProductCatalog	1		
	CashDesk	1		
	Sale	1		
	Cashier	0		
	SalesLineItem	1		
	Item	1		
<				>
Association statistics				
Source Class	Association Name	Target Class	Multiple	Association Number
SalesLineItem	BelongedSale	Sale	false	1
SalesLineItem	BelongedItem	Item	false	1

**execute enterItem before
makeNewSale**



ProcessSale - endSale

Prototype Cocome

System Function	System Status	Definition
▼ Cashier	Operation Parameters	
▼ processSale	This operation is no intput parameters..	
makeNewSale		
enterItem		
endSale		
makeCashPayment		
makeCardPayment		
openCashDesk		
closeCashDesk		
Operation Return	20.0	Precondition
System Log	<pre>18:15:41 -- execute operation: createStore in service: EntityCURDService -- success! 18:15:51 -- execute operation: createProductCatalog in service: EntityCURDService -- success! 18:16:25 -- execute operation: createCashDesk in service: EntityCURDService -- success! 18:17:31 -- execute operation: openStore in service: CoCoMEProcessSale -- success! 18:17:37 -- execute operation: openCashDesk in service: CoCoMEProcessSale -- success! 18:18:28 -- execute operation: makeNewSale in service: CoCoMEProcessSale -- success! 18:18:59 -- execute operation: createltem in service: EntityCURDService -- success! 18:19:16 -- execute operation: enterItem in service: CoCoMEProcessSale -- success! 18:22:39 -- execute operation: endSale in service: CoCoMEProcessSale -- success!</pre>	currentSale.ocllst currentSale.IsCon currentSale.IsRea
Administrator		Postcondition
StoreManager	Execute	currentS currentSale. and currentSale.IsRea result = currentSa
	Reset	Invariants

Generated By RMCode

ProcessSale - endSale

Class statistics		All Objects Sale:			
Class Name	# of Objects	Time	IsComplete	Amount	IsReadytoPay
Store	1		false	20.0	true
ProductCatalog	1				
CashDesk	1				
Sale	1				
Cashier	0				
SalesLineItem	1				
Item	1				

ProcessSale - makeCashPayment

Prototype Cocomo

System Function System Status

▼ Cashier

- ▼ processSale
 - makeNewSale
 - enterItem
 - endSale
 - makeCashPayment**
 - makeCardPayment
 - openCashDesk
 - closeCashDesk

Operation Parameters

amount:

Operation Return

true

System Log

```
18:16:25 -- execute operation: createCashDesk in service: EntityCURDService -- success!
18:17:31 -- execute operation: openStore in service: CoCoMEProcessSale -- success!
18:17:37 -- execute operation: openCashDesk in service: CoCoMEProcessSale -- success!
18:18:28 -- execute operation: makeNewSale in service: CoCoMEProcessSale -- success!
18:18:59 -- execute operation: createItem in service: EntityCURDService -- success!
18:19:16 -- execute operation: enterItem in service: CoCoMEProcessSale -- success!
18:22:39 -- execute operation: endSale in service: CoCoMEProcessSale -- success!
18:25:03 -- execute operation: makeCashPayment in service: CoCoMEProcessSale -- success!
```

< >

Execute Reset

Definition

Precondition

currentSale.oclIsU
currentSale.IsCom
currentSale.IsRead
amount >= curren

Postcondition

let cp:CashPayme
cp.oclIsNew() and
cp.AmountTender
cp.BelongedSale =
currentSale.Associ
currentSale.Belong
currentStore.Sales

Invariants

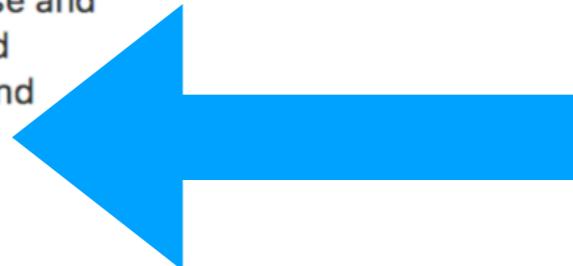
ProcessSale - makeCashPayment

Prototype Coco

System Function		System Status	
Class statistics		All Objects Sale:	
Class Name	# of Objects	Time	IsComplete
Store	1	2017-10-05	true
ProductCatalog	1		
CashDesk	1		
Sale	1		
Cashier	0		
SalesLineItem	1		
Item	1		

Requirement Validation

```
Precondition
currentSale.ocllsUndefined() = false and
currentSale.IsComplete = false and
currentSale.IsReadytoPay = true and
amount >= currentSale.Amount
```



If we miss this condition

```
Postcondition
let cp:CashPayment in
cp.ocllsNew() and
cp.AmountTendered = amount and
cp.BelongedSale = currentSale and
currentSale.AssoicatedPayment = cp and
currentSale.Belongedstore = currentStore and
currentStore.Sales->includes(currentSale) and
currentSale.IsComplete = true and
currentSale.Time = Now and
cp.Balance = amount - currentSale.Amount and
CashPayment.allInstance()->includes(cp) and
result = true
```

```
Invariants
CashPayment_BalanceGreatAndEqualZero
Balance >= 0
```

Requirement Validation

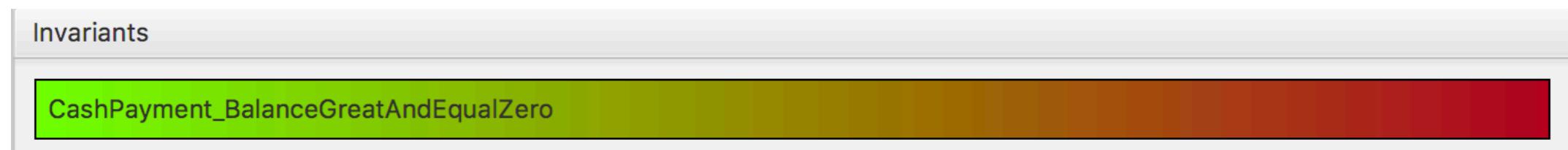
System Function	System Status	All Objects Sale:				
Class statistics		# of Objects	Time	IsComplete	Amount	IsReadytoPay
	Store	1	2017-10-06	true	100.0	true
	ProductCatalog	1				
	CashDesk	1				
	Sale	1				

**Sale Status After executing of enterItem
(we need to pay 100 MOP)**

Requirement Validation

Class statistics			All Objects CashPayment:	
Class Name	# of Objects		AmountTendered	Balance
Store	1		80.0	-20.0
ProductCatalog	1			
CashDesk	1			
Sale	1			
Cashier	0			
SalesLineItem	1			
Item	1			
Payment	0			
CashPayment	1			

System Status After executing of CashPayment with only 80 payment

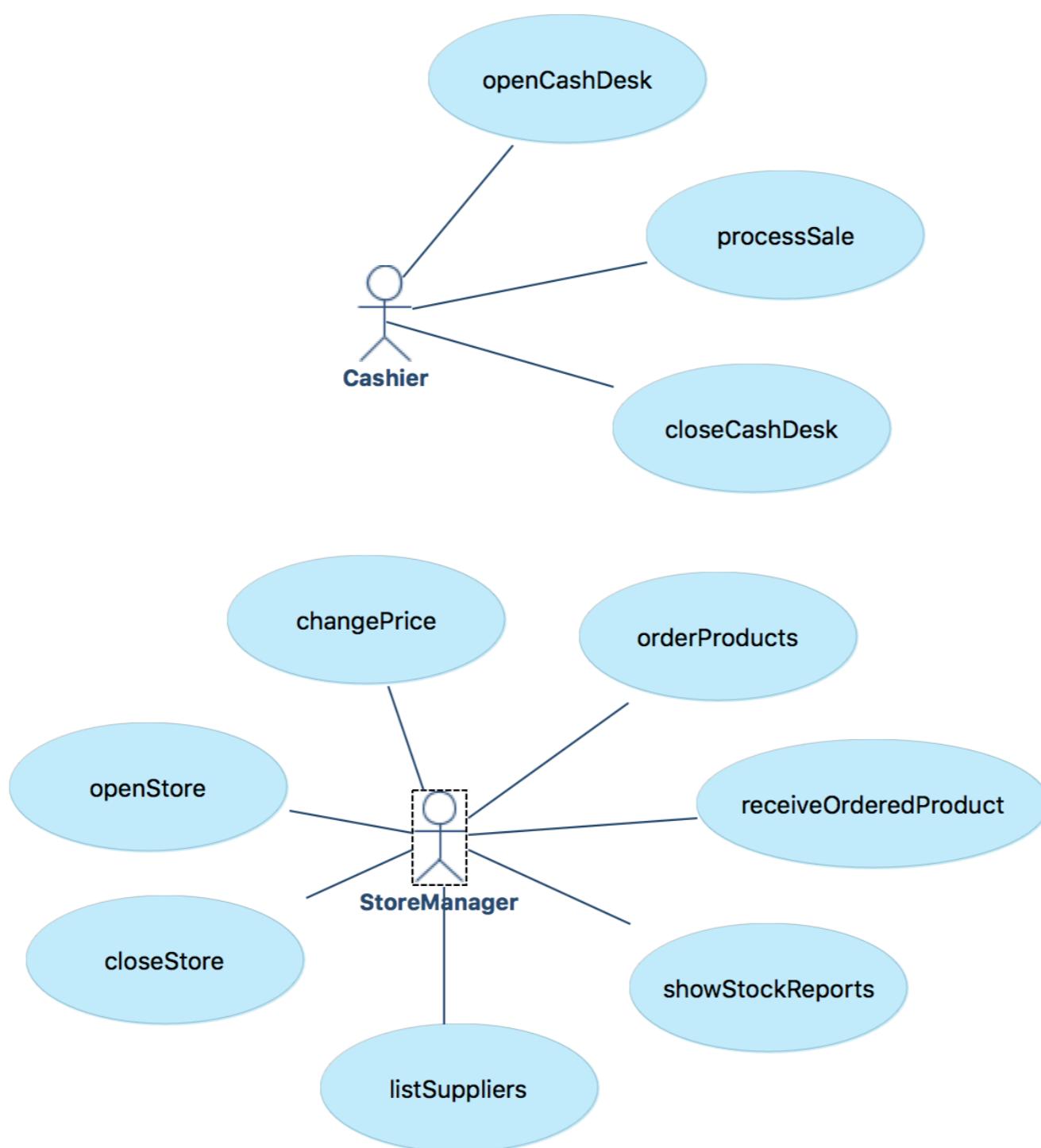


CashPayment Invariant is not holding at this moment

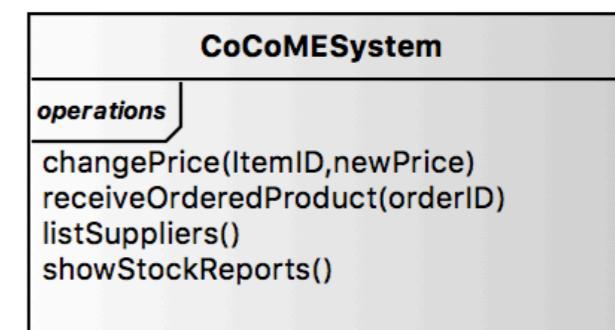
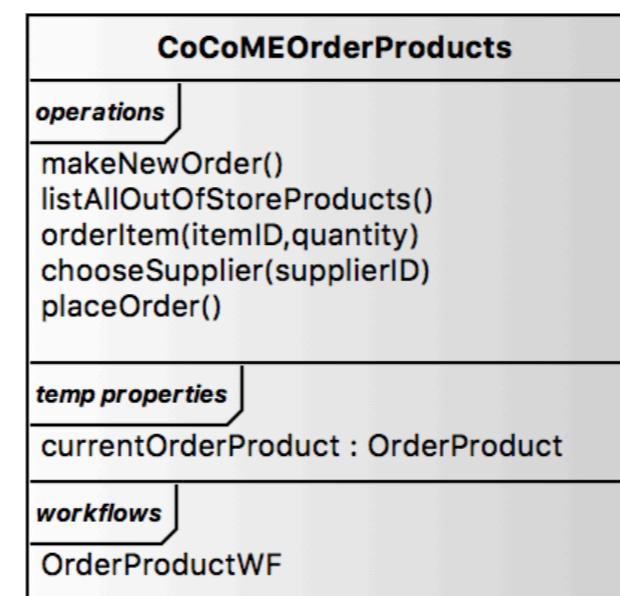
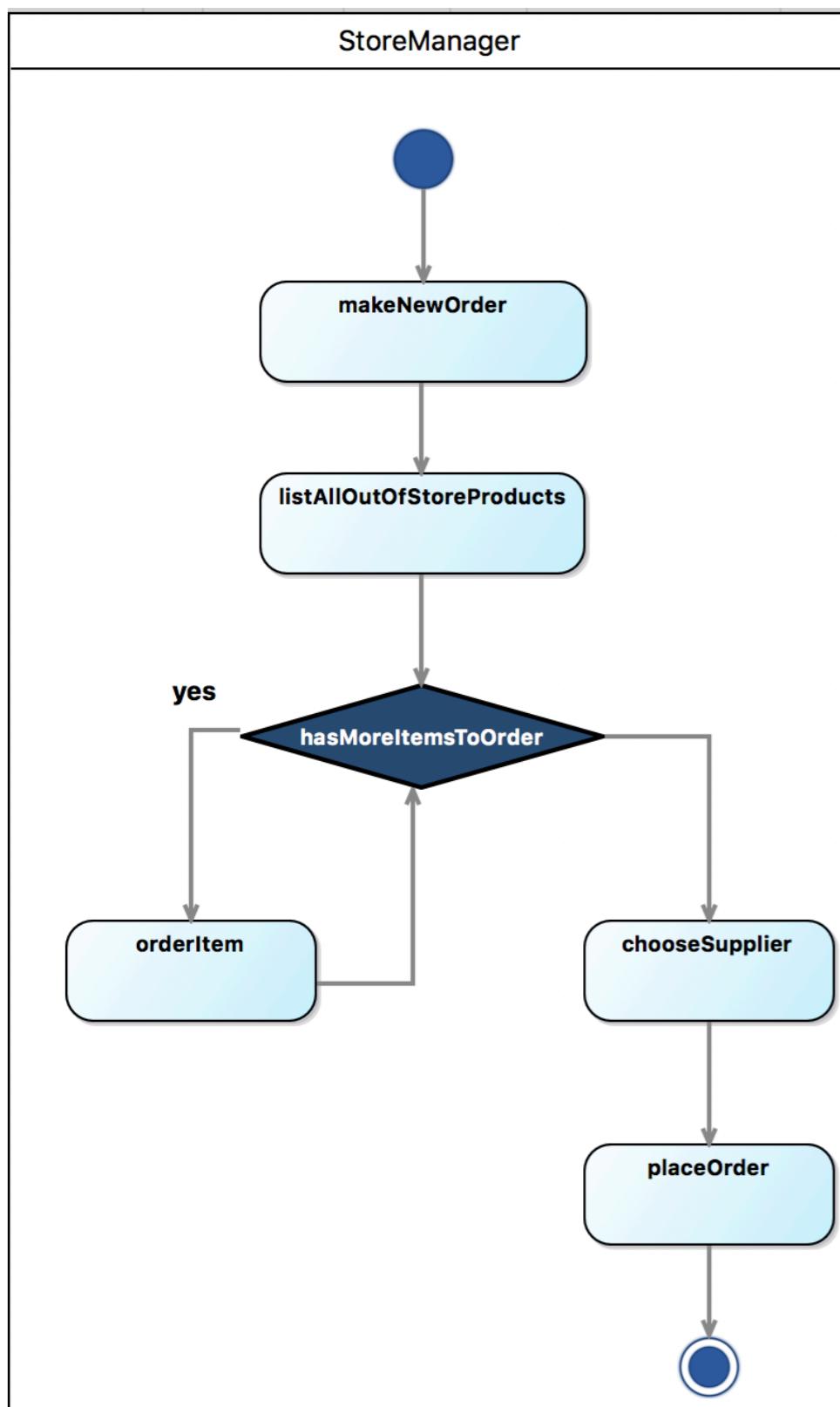
Next Step

- We specified main functionality about Cashier about checking out.
- We did not touch storage management yet....

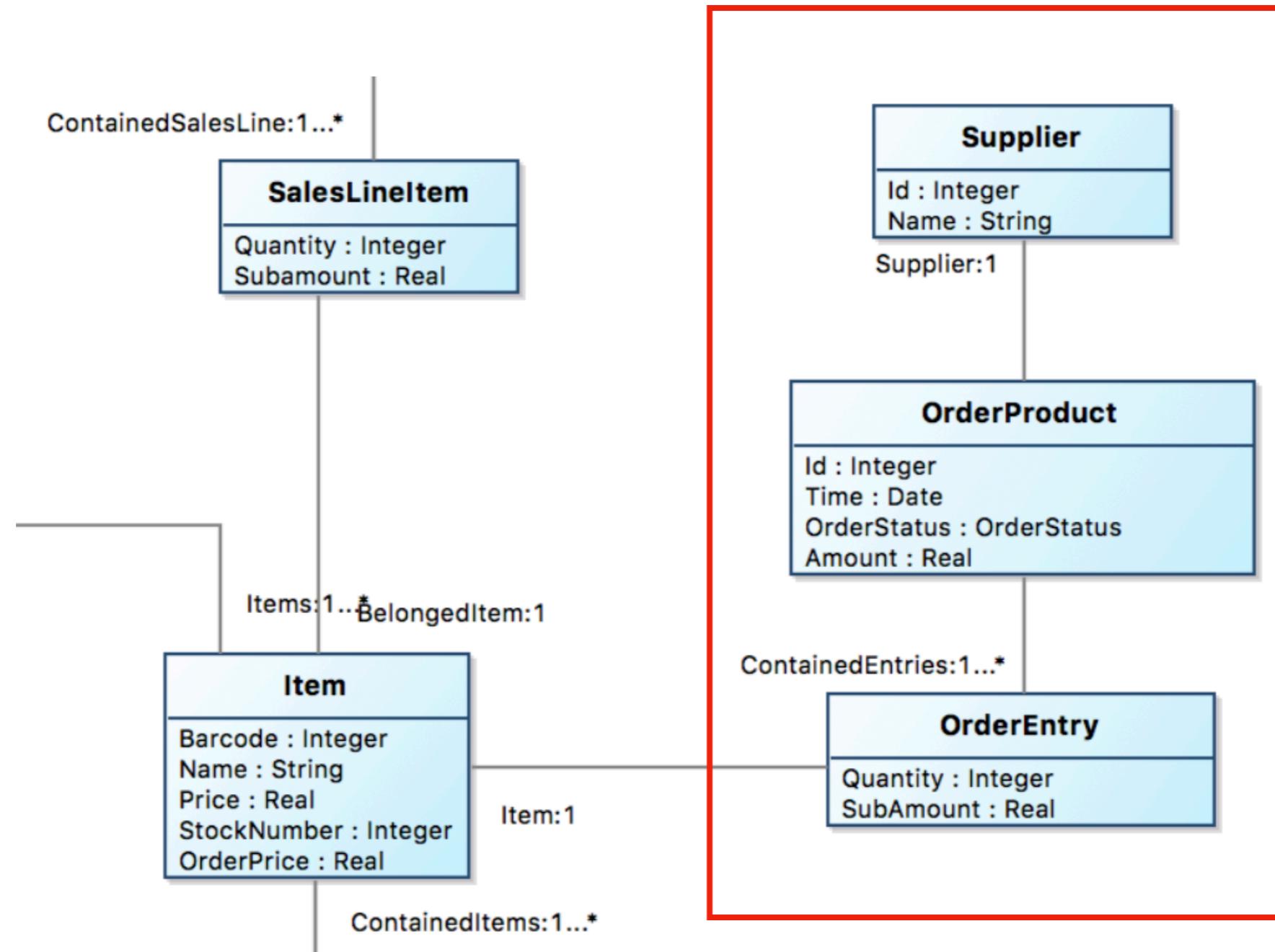
Refined Requirement Model



UseCase: OrderProducts



Refining Conceptual Class Diagram



showStockReports

The screenshot shows a software application window with a menu bar and a central content area. The menu bar includes standard OS X-style icons for close, minimize, and zoom. Below the menu is a toolbar with two buttons: "System Function" and "System Status".

The main content area is divided into two sections:

- Left Panel (System Function):** A tree view of system functions:
 - Cashier
 - Administrator
 - StoreManager
 - orderProducts
 - makeNewOrder
 - listAllOutOfStoreProducts
 - orderItem
 - chooseSupplier
 - placeOrder
 - receiveOrderedProduct
 - showStockReports
 - changePrice
 - listSuppliers
 - openStore
 - closeStore
- Right Panel (Operation Parameters and Return):**
 - Operation Parameters:** A message: "This operation is no intput parameters.."
 - Operation Return:** A table showing stock reports.

Barcode	Name	Price	StockNumber	OrderPrice	
1	Apple	10.0	1000	8.0	
2	Banana	8.0	1000	5.0	
3	Egg	20.0	1000	15.0	
4	Bacon	40.0	1000	30.0	
5	Surface Laptop	10000.0	2	9000.0	

showStockReports

The screenshot shows a software application window with a title bar containing red, yellow, and green buttons. Below the title bar is a navigation bar with two tabs: "System Function" (highlighted in yellow) and "System Status".

The left sidebar lists various system functions:

- Cashier
- Administrator
- StoreManager
 - orderProducts
 - makeNewOrder
 - listAllOutOfStoreProducts
 - orderItem
 - chooseSupplier
 - placeOrder
 - receiveOrderedProduct
 - showStockReports
 - changePrice
 - listSuppliers
 - openStore
 - closeStore

The main content area has two sections:

 - Operation Parameters:** A message states "This operation is no intput parameters..".
 - Operation Return:** A table displays the following data:

Barcode	Name	Price	StockNumber	OrderPrice
1	Apple	10.0	1000	8.0
2	Banana	8.0	1000	5.0
3	Egg	20.0	1000	15.0
4	Bacon	40.0	1000	30.0
5	Surface Laptop	10000.0	2	9000.0

changePrice

System Function	System Status
▶ Cashier	
▶ Administrator	
▼ StoreManager	
▼ orderProducts	
makeNewOrder	
listAllOutOfStoreProducts	
orderItem	
chooseSupplier	
placeOrder	
receiveOrderedProduct	
showStockReports	
changePrice	
listSuppliers	
openStore	
closeStore	

Operation Parameters

barcode:

newPrice:

Operation Return

changePrice

All Objects Item:				
Barcode	Name	Price	StockNumber	OrderPrice
1	Apple	10.0	1000	8.0
2	Banana	8.0	1000	5.0
3	Egg	20.0	1000	15.0
4	Bacon	40.0	1000	30.0
5	Surface Laptop	9000.0	2	9000.0

After checking out this two surface laptop

All Objects Item:				
Barcode	Name	Price	StockNumber	OrderPrice
1	Apple	10.0	1000	8.0
2	Banana	8.0	1000	5.0
3	Egg	20.0	1000	15.0
4	Bacon	40.0	1000	30.0
5	Surface Laptop	9000.0	0	9000.0

makeNewOrder

Prototype

System Function System Status

► Cashier

► Administrator

▼ StoreManager

▼ orderProducts

makeNewOrder

listAllOutOfStoreProducts

orderItem

chooseSupplier

placeOrder

receiveOrderedProduct

showStockReports

changePrice

listSuppliers

openStore

closeStore

Operation Parameters

orderid: 1

Operation Return

true

This screenshot shows a software interface for managing orders. The left sidebar lists various system functions and roles, with 'makeNewOrder' currently selected under 'orderProducts'. The main panel is divided into two sections: 'Operation Parameters' and 'Operation Return'. In the 'Operation Parameters' section, there is a single input field with the value 'orderid: 1'. In the 'Operation Return' section, the value 'true' is displayed. The top right corner of the interface has the word 'Prototype'.

Listing out of stock products

Operation Parameters				
This operation is no intput parameters..				
Operation Return				
Barcode	Name	Price	StockNumber	OrderPrice
No content in table				
Postcondition				
result = Item.allInstance()->select(item:Item item.Price = 0)				

Bug? Let's check post-condition

result = Item.allInstance()->select(item:Item | item.Price = 0)

Should be item.StockNumber = 0

Fixing error requirement, re-generation, and running prototype

Operation Parameters											
This operation is no intput parameters..											
Operation Return											
<table><thead><tr><th>Barcode</th><th>Name</th><th>Price</th><th>StockNumber</th><th>OrderPrice</th></tr></thead><tbody><tr><td></td><td></td><td></td><td></td><td></td></tr></tbody></table>		Barcode	Name	Price	StockNumber	OrderPrice					
Barcode	Name	Price	StockNumber	OrderPrice							
No content in table											
Postcondition											
result = Item.allInstance()->select(item:Item item.Price = 0)											

Bug? Let's check post-condition

Postcondition

result = Item.allInstance()->select(item:Item | item.Price = 0)

Should be item.StockNumber = 0

Fixing error requirement, re-generation, and running prototype

► Cashier	Operation Parameters				
► Administrator	This operation is no intput parameters..				
▼ StoreManager	Operation Return				
▼ orderProducts					
makeNewOrder	Barcode	Name	Price	StockNumber	OrderPrice
listAllOutOfStoreProducts	5	Surface Laptop	9000.0	0	9000.0
orderItem					
chooseSupplier					
placeOrder					

Executing orderItem

System Function System Status

- Cashier
- Administrator
- ▼ StoreManager
- ▼ orderProducts
 - makeNewOrder
 - listAllOutOfStoreProducts
 - orderItem
 - chooseSupplier
 - placeOrder

Operation Parameters

barcode:

quantity:

Operation Return

All Objects Item:

Barcode	Name	Price	StockNumber	OrderPrice
1	Apple	10.0	1000	8.0
2	Banana	8.0	1000	5.0
3	Egg	20.0	1000	15.0
4	Bacon	40.0	1000	30.0
5	Surface Laptop	9000.0	0	9000.0

List suppliers and choose one

Operation Parameters	
Cashier	
Administrator	
StoreManager	
orderProducts	
makeNewOrder	
listAllOutOfStoreProducts	
orderItem	
chooseSupplier	
placeOrder	
receiveOrderedProduct	
showStockReports	
changePrice	
listSuppliers	

Operation Parameters	
supplierID:	1
Operation Return	
Cashier	
Administrator	
StoreManager	
orderProducts	
makeNewOrder	
listAllOutOfStoreProducts	
orderItem	
chooseSupplier	
placeOrder	

Place Order

► Cashier	Operation Parameters
► Administrator	This operation is no intput parameters..
▼ StoreManager	Operation Return
▼ orderProducts	
makeNewOrder	
listAllOutOfStoreProducts	
orderItem	
chooseSupplier	
placeOrder	
receiveOrderedProduct	
showStockReports	
changePrice	
listSuppliers	
openStore	
closeStore	

All Objects Item:

Barcode	Name	Price	StockNumber	OrderPrice
1	Apple	10.0	1000	8.0
2	Banana	8.0	1000	5.0
3	Egg	20.0	1000	15.0
4	Bacon	40.0	1000	30.0
5	Surface Laptop	9000.0	0	9000.0

Place Order

► Cashier	Operation Parameters
► Administrator	This operation is no intput parameters..
▼ StoreManager	Operation Return
▼ orderProducts	
makeNewOrder	
listAllOutOfStoreProducts	
orderItem	
chooseSupplier	
placeOrder	
receiveOrderedProduct	
showStockReports	
changePrice	
listSuppliers	
openStore	
closeStore	
	true

All Objects Item:				
Barcode	Name	Price	StockNumber	OrderPrice
1	Apple	10.0	1000	8.0
2	Banana	8.0	1000	5.0
3	Egg	20.0	1000	15.0
4	Bacon	40.0	1000	30.0
5	Surface Laptop	9000.0	0	9000.0

All Objects OrderProduct:			
Id	Time	OrderStatus	Amount
3	2017-10-25	REQUESTED	90000.0

Receive Ordered Products

► Cashier

► Administrator

▼ StoreManager

▼ orderProducts

- makeNewOrder
- listAllOutOfStoreProducts
- orderItem
- chooseSupplier
- placeOrder

receiveOrderedProduct

showStockReports

changePrice

listSuppliers

openStore

closeStore

Operation Parameters

orderID: 3

Operation Return

► Cashier

► Administrator

▼ StoreManager

▼ orderProducts

- makeNewOrder
- listAllOutOfStoreProducts
- orderItem
- chooseSupplier
- placeOrder

receiveOrderedProduct

showStockReports

Operation Parameters

This operation is no input parameters.

Operation Return

Barcode	Name	Price	StockNumber	OrderPrice
1	Apple	10.0	1000	8.0
2	Banana	8.0	1000	5.0
3	Egg	20.0	1000	15.0
4	Bacon	40.0	1000	30.0
5	Surface Laptop	9000.0	10	9000.0

CoCoME (Supermarket System)

- UML requirement model
- Refine UML requirement model in RMCode
- Validate and fix requirements

Case Study II

UM Library Management System

Use Case Diagram

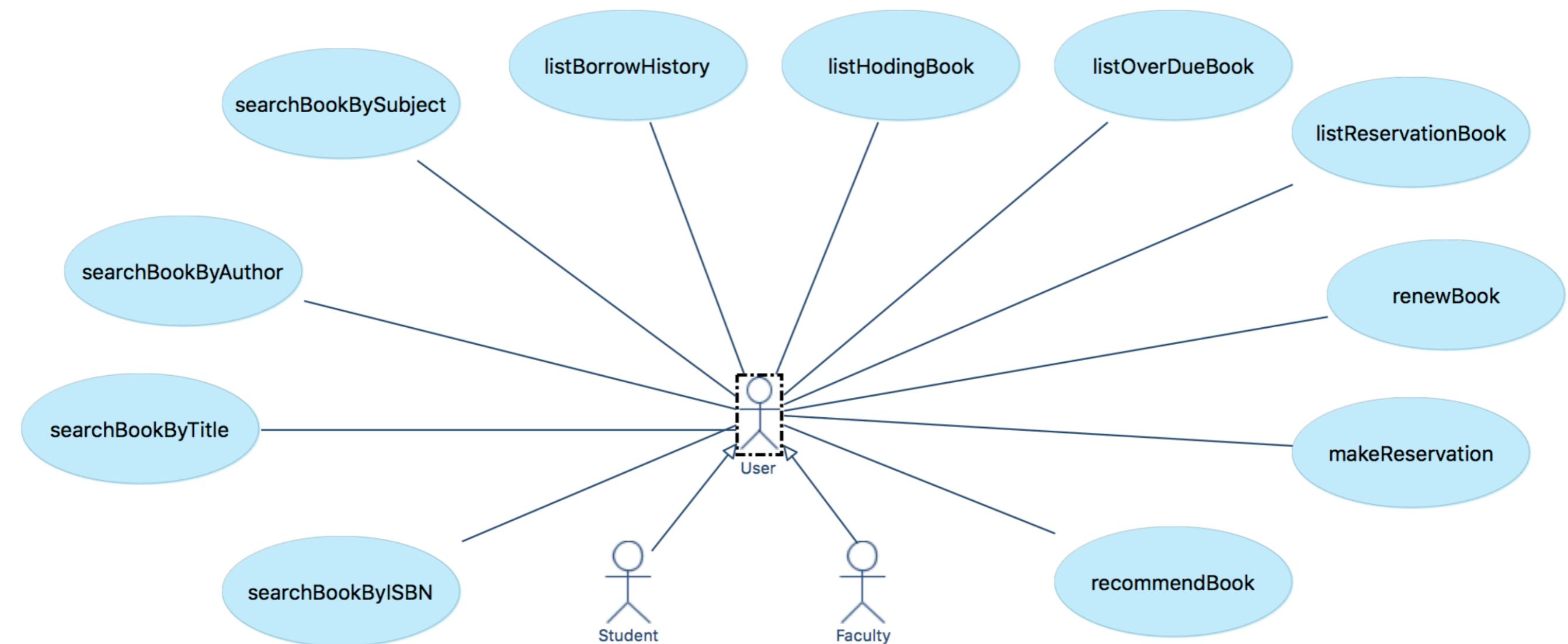


Figure 2. Use Cases of Actor Librarian

Use Case Diagram

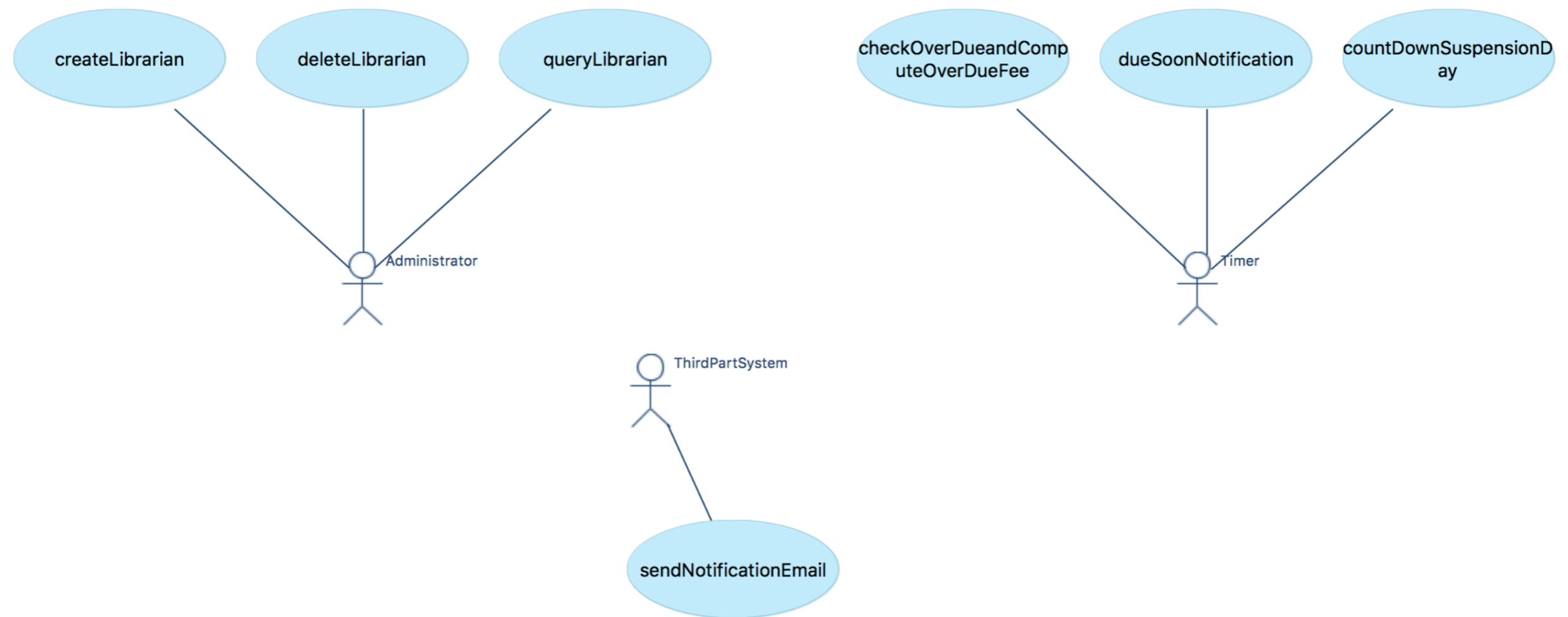


Figure 3. Use Cases of Actor Timer, Administrator, and ThridPartSystem

LibraryManagementSystem

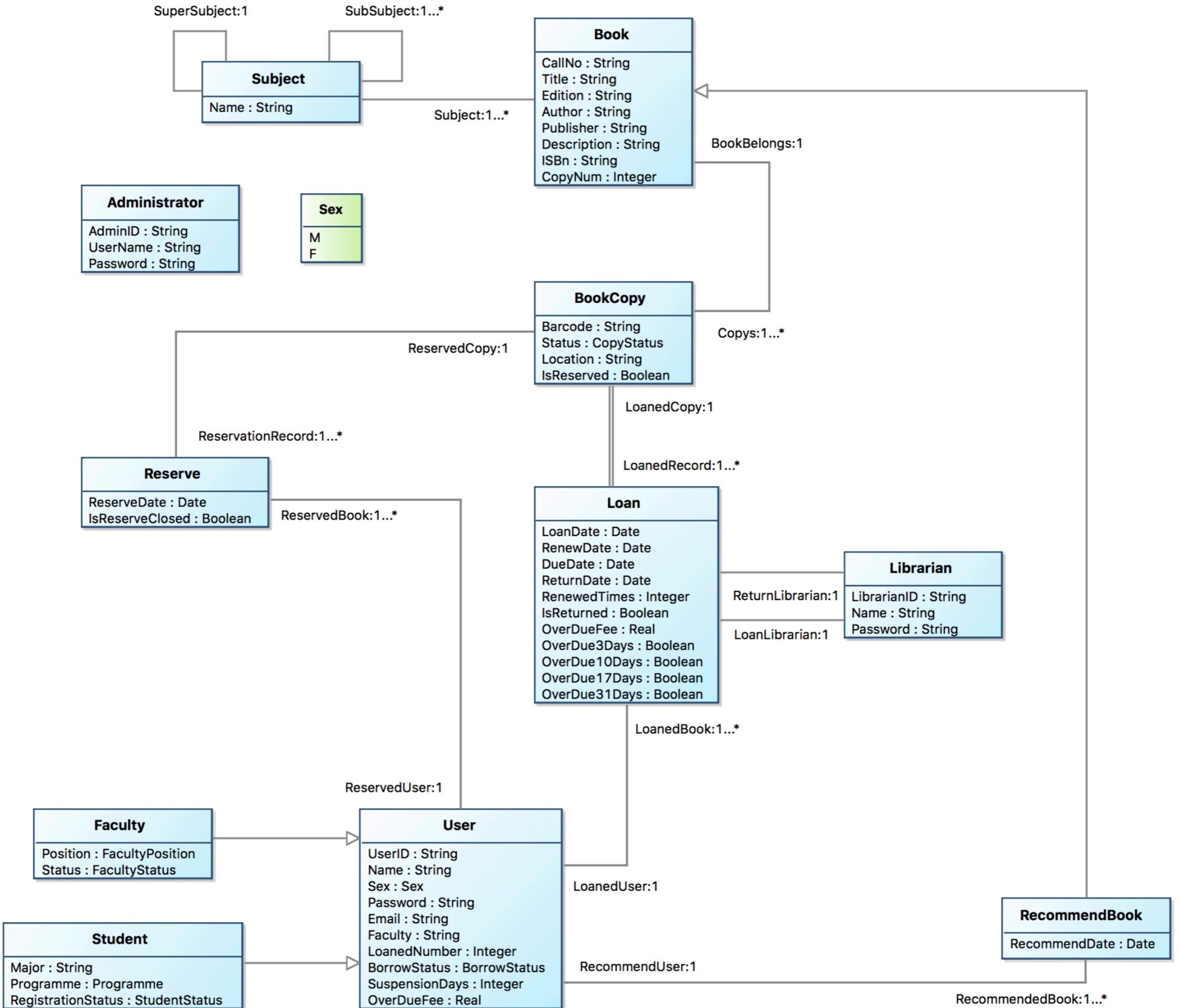
operations

searchBookByBarCode(barcode)
searchBookByTitle(title)
searchBookByAuthor(authorname)
searchBookByISBN(ISBNnumber)
searchBookBySubject(subject)
addBook(book)
deleteBook(barcode)
addSubject()
listAllSubject()
deleteSubject()
recommendBook(userid,book)
queryBookCopy(barcode)
addBookCopy(callNo,copy)
deleteBookCopy(barcode)
makeReservation(uid,barcode)
cancelReservation(uid,barcode)
borrowBook(uid,barcode)
renewBook(uid,barcode)
returnBook(barcode)
payOverDueFee(uid,fee,change)
listBorrowHistory(userid)
listHoldingBook(userid)
listOverDueBook(userid)
listReservationBook(userid)
listRecommendBook(userid)
checkOverDueandComputeOverDueFee()
dueSoonNotification()
countDownSuspensionDay()
createStudent()
modifyStudent()
createFaculty()
modifyFaculty()
deleteUser(uid)
queryUser(uid)
createLibrarian(librarian)
deleteLibrarian(librarianid)
queryLibrarian(librarianid)
createLibrarianByDetails()

ThirdPartServices

operations

sendNotificationEmail(user)



The contract of *borrowBook*

```
Contract LibraryManagementSystem::borrowBook(uid:String, barcode:String) : Boolean {  
  
    definition:  
        user:User = User.allInstances()->any(u:User | u.UserID = uid),  
        stu:Student = Student.allInstances()->any(s:Student | s.UserID = uid),  
        fac:Faculty = Faculty.allInstances()->any(f:Faculty | f.UserID = uid),  
        copy:BookCopy = BookCopy.allInstances()->any(bc:BookCopy | bc.Barcode = barcode),  
        res:Reserve = Reserve.allInstances()->any(r:Reserve | r.ReservedCopy = copy and r.ReservedUser = user and r.IsReserveClosed = false)
```

```
precondition:  
    user.oclIsUndefined() = false and  
    copy.oclIsUndefined() = false and  
    user.BorrowStatus = BorrowStatus::NORMAL and  
    user.SuspensionDays = 0 and  
    if  
        user.oclIsTypeOf(Student)  
    then  
        if  
            stu.Programme = Programme::BACHELOR  
        then  
            stu.LoanedNumber < 20  
        else  
            if  
                stu.Programme = Programme::MASTER  
            then  
                stu.LoanedNumber < 40  
            else  
                stu.LoanedNumber < 60  
            endif  
        endif  
    else  
        fac.LoanedNumber < 60  
    endif and  
(copy.Status = CopyStatus::AVAILABLE or  
 (copy.Status = CopyStatus::ONHOLDSHELF and  
 copy.IsReserved = true and  
 res.oclIsUndefined() = false and  
 res.IsReserveClosed = false  
 )  
)
```

```
postcondition:
    let loan:Loan in
        loan.oclIsNew() and
        loan.LoanedUser = user and
        loan.LoanedCopy = copy and
        loan.IsReturned = false and
        loan.LoanDate = Today and
        user.LoanedNumber = user.LoanedNumber@pre + 1 and
        user.LoanedBook->includes(loan) and
        copy.LoanedRecord->includes(loan) and
        if
            user.oclIsTypeOf(Student)
        then
            loan.DueDate = Today.After(30)
        else
            loan.DueDate = Today.After(60)
        endif and
        if
            copy.Status@pre = CopyStatus::ONHOLDSHELF
        then
            copy.IsReserved = false and
            res.IsReserveClosed = true
        endif and
        copy.Status = CopyStatus::LOANED and
        loan.OverDue3Days = false and
        loan.OverDue10Days = false and
        loan.OverDue17Days = false and
        loan.OverDue31Days = false and
        Loan.allInstance()->includes(loan) and
        result = true
```

The contract of *renewBook*

```
Contract LibraryManagementSystem::renewBook(uid:String, barcode:String) : Boolean {  
  
    definition:  
        user:User = User.allInstances()->any(u:User | u.UserID = uid),  
        stu:Student = Student.allInstances()->any(s:Student | s.UserID = uid),  
        fac:Faculty = Faculty.allInstances()->any(f:Faculty | f.UserID = uid),  
        copy:BookCopy = BookCopy.allInstances()->any(bc:BookCopy | bc.Barcode = barcode and bc.Status = CopyStatus::LOANED),  
        loan:Loan = Loan.allInstances()->any(l:Loan | l.LoanedUser = user and l.LoanedCopy = copy)
```

The contract of *renewBook*

precondition:

```
user.BorrowStatus = BorrowStatus::NORMAL and
user.oclIsUndefined() = false and
copy.oclIsUndefined() = false and
loan.oclIsUndefined() = false and
copy.IsReserved = false and
loan.DueDate.isAfter(Today) and
if
    user.oclIsTypeOf(Student)
then
    loan.RenewedTimes < 3
else
    loan.RenewedTimes < 6
endif and
loan.OverDueFee = 0
```

The contract of *renewBook*

```
postcondition:  
    loan.RenewedTimes = loan.RenewedTimes@pre + 1 and  
    loan.RenewDate = Today and  
    if  
        user.oclIsTypeOf(Student)  
    then  
        if  
            stu.Programme = Programme::BACHELOR  
        then  
            loan.DueDate = loan.DueDate@pre.After(20)  
        else  
            if  
                stu.Programme = Programme::MASTER  
            then  
                loan.DueDate = loan.DueDate@pre.After(40)  
            else  
                loan.DueDate = loan.DueDate@pre.After(60)  
            endif  
        endif  
    endif  
else  
    loan.DueDate = loan.DueDate@pre.After(60)  
endif and  
result = true
```

The contract of *returnBook*

```
Contract LibraryManagementSystem::returnBook(barcode:String) : Boolean {  
  
    definition:  
        copy:BookCopy = BookCopy.allInstances()->any(bc:BookCopy | bc.Barcode = barcode and bc.Status = CopyStatus::LOANED),  
        loan:Loan = Loan.allInstances()->any(l:Loan | l.LoanedCopy = copy and l.IsReturned = false),  
        loans:Set(Loan) = Loan.allInstances()->select(l:Loan | l.LoanedUser = loan.LoanedUser and l.IsReturned = false and l.DueDate.isAfter(Today)),  
        res:Reserve = copy.ReservationRecord->any(r:Reserve | r.ReservedCopy = copy)
```

The contract of *returnBook*

precondition:

```
copy.oclIsUndefined() = false and  
loan.oclIsUndefined() = false
```

postcondition:

```
loan.LoanedUser.LoanedNumber = loan.LoanedUser.LoanedNumber@pre - 1 and  
loan.IsReturned = true and  
loan.ReturnDate = Today and  
if  
    copy.IsReserved = true  
then  
    copy.Status = CopyStatus::ONHOLDSHELF and  
    sendNotificationEmail(res.ReservedUser.Email)  
else  
    copy.Status = CopyStatus::AVAILABLE  
endif and  
result = true
```

The contract of *dueSoonNotification*

```
Contract LibraryManagementSystem::dueSoonNotification() {  
  
    precondition:  
        true  
  
    postcondition:  
        let users:Set(User) = User.allInstance()->select(user:User | user.LoanedBook->exists(loan:Loan |  
            loan.IsReturned = false and Today.After(3).isAfter(loan.DueDate)  
        )) in  
        users->forAll(u:User |  
            sendNotificationEmail(u.Email))  
}
```

The contract of *makeReservation*

```
|Contract LibraryManagementSystem::makeReservation(uid:String, barcode:String) : Boolean {  
|  
|  definition:  
|    user:User = User.allInstances()->any(u:User | u.UserID = uid),  
|    copy:BookCopy = BookCopy.allInstances()->any(bc:BookCopy | bc.Barcode = barcode)  
|  
|  precondition:  
|    user.oclIsUndefined() = false and  
|    copy.oclIsUndefined() = false and  
|    copy.Status = CopyStatus::LOANED and  
|    copy.IsReserved = false
```

The contract of *makeReservation*

postcondition:

```
let res:Reserve in
res.oclIsNew() and
copy.IsReserved = true and
res.IsReserveClosed = false and
res.ReserveDate = Today and
res.ReservedUser = user and
res.ReservedCopy = copy and
user.ReservedBook->includes(res) and
copy.ReservationRecord->includes(res) and
Reserve.allInstances()->includes(res) and
result = true
```

Prototype Functionality

Prototype Library

System Function		System Status	
<ul style="list-style-type: none"> ► User ► Student ► Faculty ▼ Librarian 		<p>Operation Parameters</p> <p>uid: <input type="text"/></p> <p>barcode: <input type="text"/></p> <p>Operation Return</p>	
		<p>Definition</p> <pre> user:User = User.allInstances()->any(u:User u.UserID = uid) stu:Student = Student.allInstances()->any(s:Student s.UserID = uid) fac:Faculty = Faculty.allInstances()->any(f:Faculty f.UserID = uid) copy:BookCopy = BookCopy.allInstances()->any(bc:BookCopy bc.Barcode = barcode) res:Reserve = Reserve.allInstances()->any(r:Reserve r.ReservedCopy = copy and r.ReservedUser = user and r.IsReserveClose </pre> <p>Precondition</p> <pre> user.oclsIsUndefined() = false and copy.oclsIsUndefined() = false and user.BorrowStatus = BorrowStatus::NORMAL and user.SuspensionDays = 0 and if user.oclsTypeOf(Student) then if </pre> <p>Postcondition</p> <pre> let loan:Loan in loan.oclsIsNew() and loan.LoanedUser = user and loan.LoanedCopy = copy and loan.IsReturned = false and loan.LoanDate = Today and user.LoanedNumber = user.LoanedNumber@pre + 1 and user.LoanedBook->includes(loan) and </pre> <p>Invariants</p> <ul style="list-style-type: none"> Loan_OverDueFeeGreatThanEqualZero Loan_RenewedTimesLessThanEqualSix Loan_LoanOverDueFeeGreatThanEqualZero Loan_RenewDataAfterLoanDate Loan_DueDateAfterLoanDate Loan_ReturnDateAfterORSameLoanDate Loan_DueDateAfterORSameRenewDate Loan_ReturnDateSameORAfterRenewDate BookCopy_BarCodeUnique User_UniqueUserID User_OverDueFeeGreatThanEqualZero User_LoanedNumberGreatThanEqualZero User_SuspensionDaysGreatThanEqualZero 	
		<p>System Log</p>	
<ul style="list-style-type: none"> ► Administrator ► Timer ► ThirdPartSystem 		<p>Execute Reset</p>	

Prototype Status

Prototype Library					
System Function		System Status			
Class statistics		Object Statistics		All Invariants	
Class Name		# of Objects		User_UniqueUserID	
User		0		User_OverDueFeeGreatThanEqualZero	
Student		0		User_LoanedNumberGreatThanEqualZero	
Faculty		0		User_SuspensionDaysGreatThanEqualZero	
Book		0		Student_StudentLoanLessThanEqualTwelve	
Subject		0		Student_StudentLoanedBookAssociationInvariants	
BookCopy		0		Faculty_FacultyLoanLessThanEqualTwentyFour	
Loan		0		Faculty_FacultyLoanedBookAssociationInvariants	
Reserve		0		Book_BookCallNoUnique	
RecommendBook		0		Book_BookISBNUnique	
Administrator		0		Book_BookCopyNumGreatThanEqualZero	
Librarian		0		BookCopy_BarCodeUnique	
				Loan_OverDueFeeGreatThanEqualZero	
				Loan_RenewedTimesLessThanEqualSix	
				Loan_LoanOverDueFeeGreatThanEqualZero	
				Loan_RenewDataAfterLoanDate	
				Loan_DueDateAfterLoanDate	
				Loan_ReturnDateAfterORSameLoanDate	
				Loan_DueDateAfterORSameRenewDate	
				Loan_ReturnDateSameORAAfterRenewDate	
				RecommendBook_BookCallNoUnique	
				RecommendBook_BookISBNUnique	
				RecommendBook_BookCopyNumGreatThanEqualZero	
				Administrator_AdministratorIDUnique	
				Librarian_LibrarianIDUnique	
Association statistics					
Source Class	Association Name	Target Class	Multiple	Association Number	
No content in table					
Load Status Save Status Refresh Status Check All Invariants					

RMCode

- RMCode is a good UML requirement modelling tool
- RMCode provides an evolutionary approach to validate the requirement
- Good for training your mind as like project manager

Contribute to RMCode

- Download RMCode from
 - <http://lab.mydreamy.net>
- Try the provided case studies from GitHub
 - <https://github.com/yylonly/RMCodeCaseStudies.git>
- Create new case studies, then make a pull request
- Open a new issue on GitHub, if bugs were found.

Contribute to RMCode

If you are interested in the development of :

- Full-stack JavaScript Web Application
- Java EE Application
- Android/IOS Application

Thank You