1 Introduction

1.1 Purpose

This subsection should

- a) Delineate the purpose of the SRS;
- b) Specify the intended audience for the SRS.

1.2 Scope

Name of software to be developed: CoCoME System

This subsection should

- b) Explain what the software product(s) will, and, if necessary, will not do;
- c) Describe the application of the software being specified, including relevant benefifits, objectives, and goals;
- d) Be consistent with similar statements in higher-level specififications (e.g., the system requirements specifification), if they exist.

1.3 Definitions, acronyms, and abbreviations

This subsection should provide the defifinitions of all terms, acronyms, and abbreviations required to properly interpret the SRS. This information may be provided by reference to one or more appendixes in the SRS or by reference to other documents.

1.4 References

This subsection should

- a) Provide a complete list of all documents referenced elsewhere in the SRS;
- b) Identify each document by title, report number (if applicable), date, and publishing organization;
- c) Specify the sources from which the references can be obtained.

This information may be provided by reference to an appendix or to another document.

1.5 Overview

This subsection should

- a) Describe what the rest of the SRS contains;
- b) Explain how the SRS is organized.

2 Overall description

2.1 Product perspective

This subsection of the SRS should put the product into perspective with other related products. If the product is independent and totally self-contained, it should be so stated here. If the SRS defines a product that is a component of a larger system, as frequently occurs, then this subsection should relate the requirements of that larger system to functionality of the software and should identify interfaces between that system and the software.

2.2 Product functions

2.3 User characteristics

The applicable objects of this system are Cashier, StoreManager, Administrator.
If they know the basic operation of computer, they can use the system to operate the required functions.
Maybe some users need some relevant training.

2.4 Constraints

This subsection of the SRS should provide a general description of any other items that will limit the developer's options. These include

- a) Regulatory policies;
- b) Hardware limitations (e.g., signal timing requirements);
- c) Interfaces to other applications;
- d) Parallel operation;
- e) Audit functions;
- f) Control functions;
- g) Higher-order language requirements;
- h) Signal handshake protocols (e.g., XON-XOFF, ACK-NACK);
- i) Reliability requirements;
- j) Criticality of the application;
- k) Safety and security considerations.

2.5 Assumptions and dependencies

This subsection of the SRS should list each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the software but are, rather, any changes to them that can affect the requirements in the SRS. For example, an assumption may be that a specific operating system will be available on the hardware designated for the software product. If, in fact, the operating system is not available, the SRS would then have to change accordingly.

2.6 Apportioning of requirements

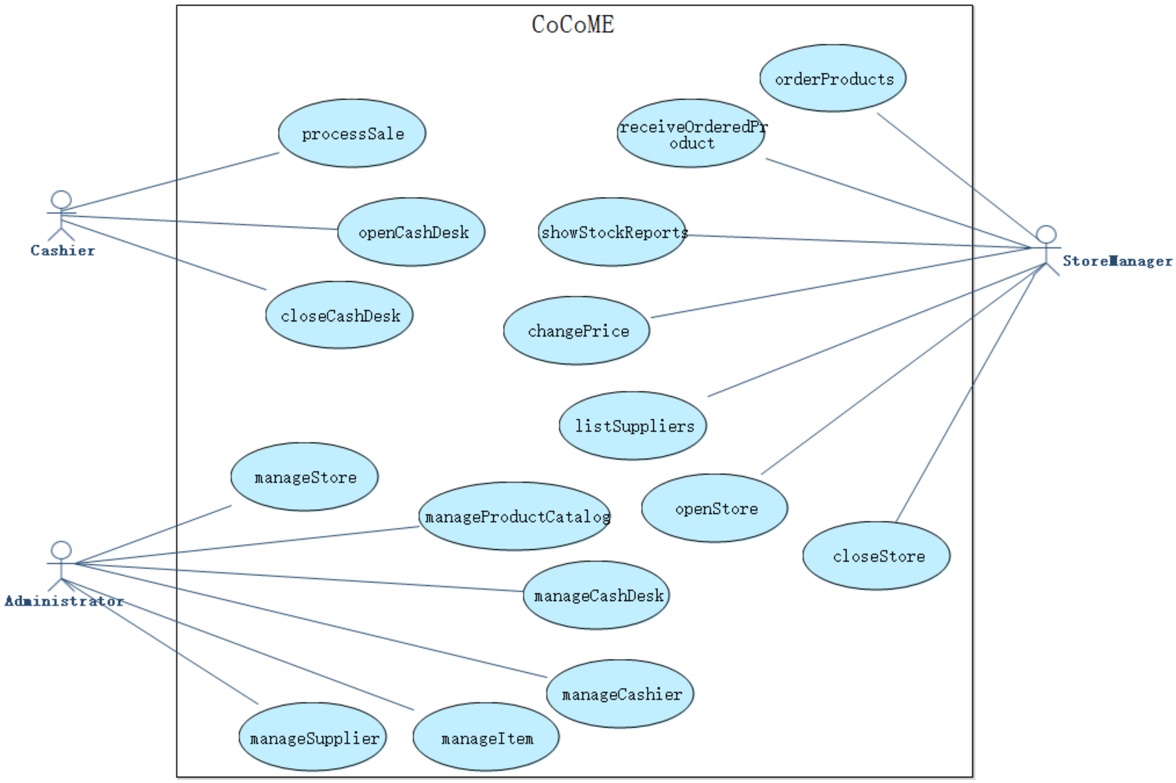
This subsection of the SRS should identify requirements that may be delayed until future versions of the system.

3 Specific requirements

3.1 Functional requirements

3.1.1 User Requirements

Use Case Diagram



A1 - Cashier

Actor Name:	Cashier
Actor ID:	A1
Description:	The cashier is responsible for opening or closing the cash desk and the checkout of items
Required Functions	**Related Use Case**
The cashier checks out items	processSale
The cashier opens cash desk	openCashDesk
The cashier closes cash desk	closeCashDesk

A2 - StoreManager

Actor Name:	StoreManager
Actor ID:	A2
Description:	The store manager is responsible for procurement and price setting of items,and opening or closing the store
Required Functions	**Related Use Case**
The store manager places an order for purchase	orderProducts
The store manager receives the order for purchase	receiveOrderedProduct
The store manager views the stock report	showStockReports
The store manager changes the price of item	changePrice
The store manager views all suppliers	listSuppliers
The store manager opens the store	openStore
The store manager closes the store	closeStore

A3 - Administrator

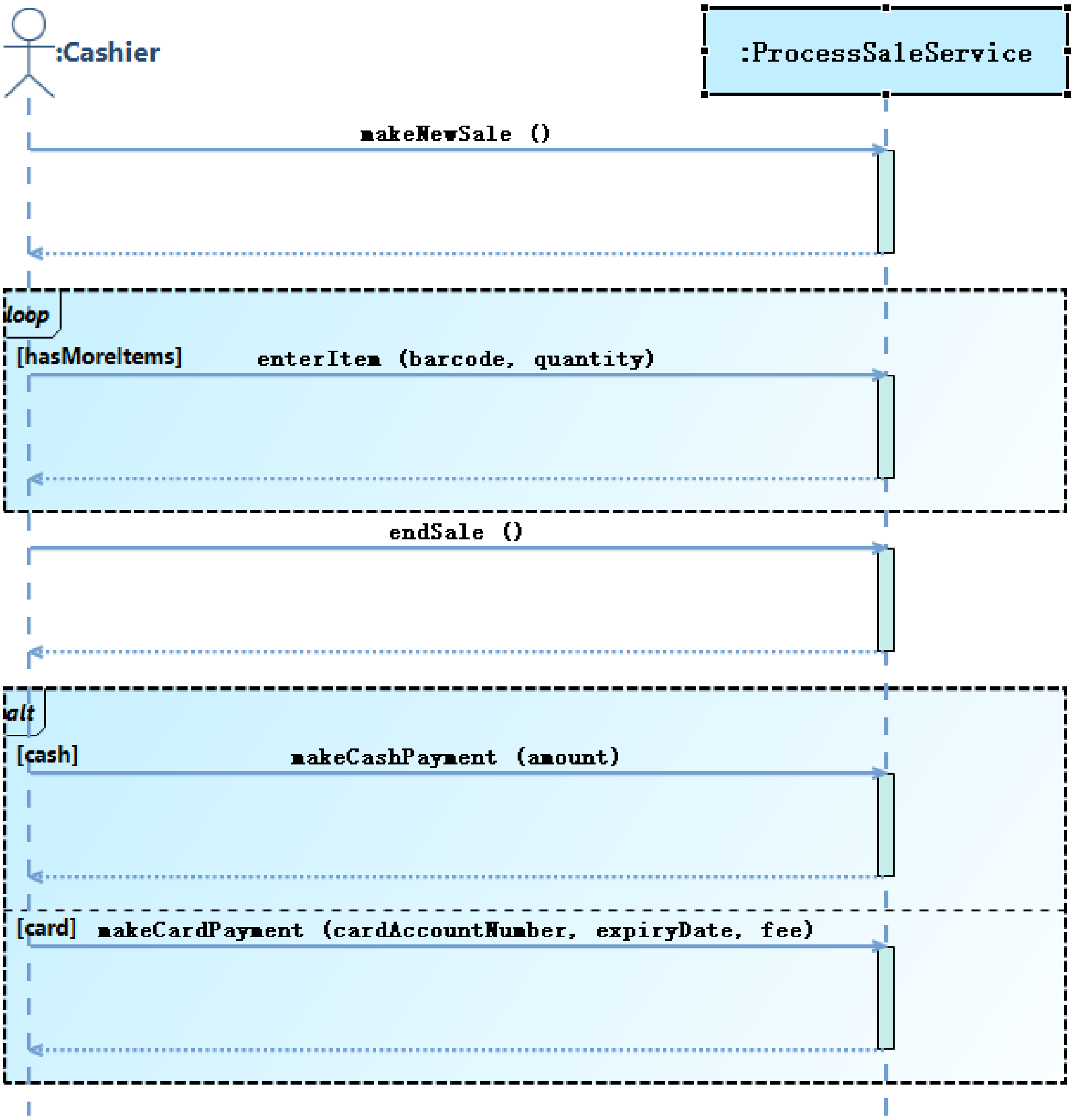
Actor Name:	Administrator
Actor ID:	A3
Description:	The system administrator is responsible for managing information, including store information, cash desk information, cashier information, item information, item catalogue information and supplier information
Required Functions	**Related Use Case**
The administrator manages store information, including entering, inquiring, modifying and deleting of store information	manageStore
The administrator manages catalogues of items, including entering, inquiring, modifying and deleting of catalogue information	manageProductCatalog
The administrator manages cash desk information, including entering, inquiring, modifying and deleting of cash desk information	manageCashDesk
The administrator manages cashier information, including entering, inquiring, modifying and deleting of cashier information	manageCashier
The administrator manages item information, including entering, inquiring, modifying and deleting of item information	manageItem
The administrator manages supplier information, including entering, inquiring, modifying and deleting of supplier information	manageSupplier

3.1.2 System Requirement

3.1.2.1 Use Case Description

UC1 - processSale

UseCase Name:	processSale
UseCase ID:	UC1
Brief Description:	The cashier checks out items
Involved Actor:	Cashier
Preconditions:	
Postconditions:	
Basic Path:	1. Cashier clicks to execute the operation makeNewSale 2. Cashier clicks to execute the operation enterItem, with entering barcode, quantity *If hasMoreItems, repeat the step(s) 2* 3. Cashier clicks to execute the operation endSale 4. Execute paymentMethodAlt Select cash: Cashier clicks to execute the operation makeCashPayment, with entering amount Select card: Cashier clicks to execute the operation makeCardPayment, with entering cardAccountNumber, expiryDate, fee
Alternative Path:	



UC2 - openCashDesk

UseCase Name:	openCashDesk
UseCase ID:	UC2
Brief Description:	The cashier opens cash desk
Involved Actor:	Cashier
Preconditions:	
Postconditions:	
Basic Path:	
Alternative Path:	

UC3 - closeCashDesk

UseCase Name:	closeCashDesk
UseCase ID:	UC3
Brief Description:	The cashier closes cash desk
Involved Actor:	Cashier
Preconditions:	
Postconditions:	
Basic Path:	
Alternative Path:	

UC4 - orderProducts

UseCase Name:	orderProducts
UseCase ID:	UC4
Brief Description:	The store manager places an order for purchase
Involved Actor:	StoreManager
Preconditions:	
Postconditions:	
Basic Path:	
Alternative Path:	

UC5 - receiveOrderedProduct

UseCase Name:	receiveOrderedProduct
UseCase ID:	UC5
Brief Description:	The store manager receives the order for purchase
Involved Actor:	StoreManager
Preconditions:	
Postconditions:	
Basic Path:	
Alternative Path:	

UC6 - showStockReports

UseCase Name:	showStockReports
UseCase ID:	UC6
Brief Description:	The store manager views the stock report
Involved Actor:	StoreManager
Preconditions:	
Postconditions:	
Basic Path:	
Alternative Path:	

UC7 - changePrice

UseCase Name:	changePrice
UseCase ID:	UC7
Brief Description:	The store manager changes the price of item
Involved Actor:	StoreManager
Preconditions:	
Postconditions:	
Basic Path:	
Alternative Path:	

UC8 - listSuppliers

UseCase Name:	listSuppliers
UseCase ID:	UC8
Brief Description:	The store manager views all suppliers
Involved Actor:	StoreManager
Preconditions:	
Postconditions:	
Basic Path:	
Alternative Path:	

UC9 - openStore

UseCase Name:	openStore
UseCase ID:	UC9
Brief Description:	The store manager opens the store
Involved Actor:	StoreManager
Preconditions:	
Postconditions:	
Basic Path:	
Alternative Path:	

UC10 - closeStore

UseCase Name:	closeStore
UseCase ID:	UC10
Brief Description:	The store manager closes the store
Involved Actor:	StoreManager
Preconditions:	
Postconditions:	
Basic Path:	
Alternative Path:	

UC11 - manageStore

UseCase Name:	manageStore
UseCase ID:	UC11
Brief Description:	The administrator manages store information, including entering, inquiring, modifying and deleting of store information
Involved Actor:	Administrator
Preconditions:	
Postconditions:	
Basic Path:	
Alternative Path:	

UC12 - manageProductCatalog

UseCase Name:	manageProductCatalog
UseCase ID:	UC12
Brief Description:	The administrator manages catalogues of items, including entering, inquiring, modifying and deleting of catalogue information
Involved Actor:	Administrator
Preconditions:	
Postconditions:	
Basic Path:	
Alternative Path:	

UC13 - manageCashDesk

UseCase Name:	manageCashDesk
UseCase ID:	UC13
Brief Description:	The administrator manages cash desk information, including entering, inquiring, modifying and deleting of cash desk information
Involved Actor:	Administrator
Preconditions:	
Postconditions:	
Basic Path:	
Alternative Path:	

UC14 - manageCashier

UseCase Name:	manageCashier
UseCase ID:	UC14
Brief Description:	The administrator manages cashier information, including entering, inquiring, modifying and deleting of cashier information
Involved Actor:	Administrator
Preconditions:	
Postconditions:	
Basic Path:	
Alternative Path:	

UC15 - manageItem

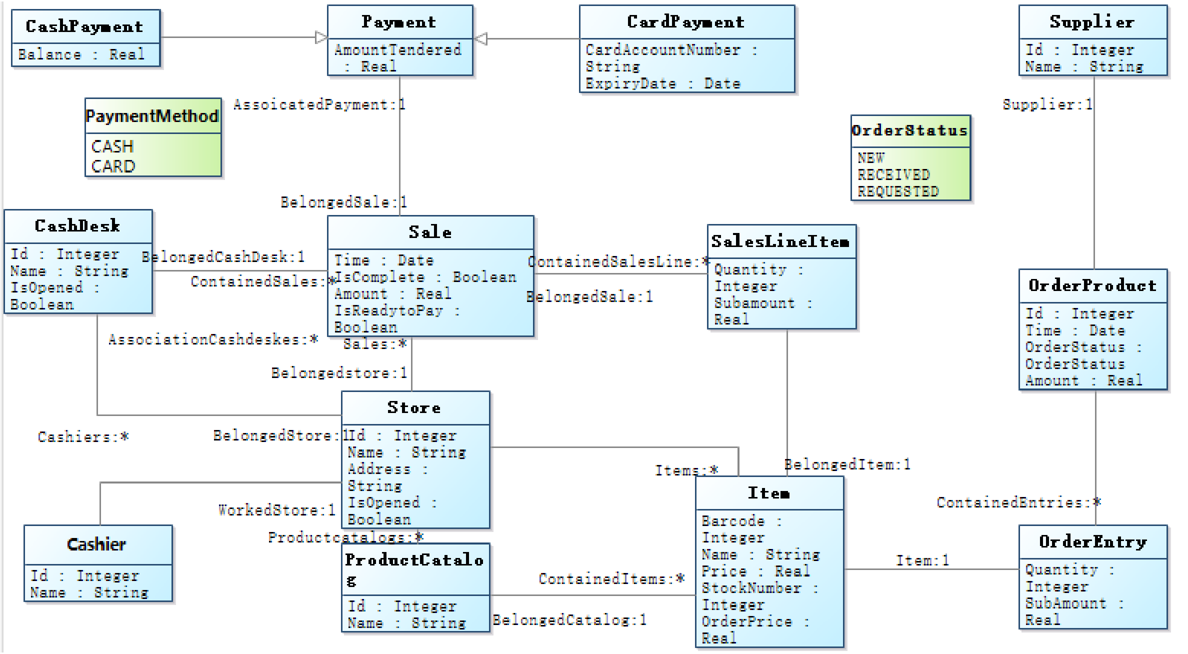
UseCase Name:	manageItem
UseCase ID:	UC15
Brief Description:	The administrator manages item information, including entering, inquiring, modifying and deleting of item information
Involved Actor:	Administrator
Preconditions:	
Postconditions:	
Basic Path:	
Alternative Path:	

UC16 - manageSupplier

UseCase Name:	manageSupplier
UseCase ID:	UC16
Brief Description:	The administrator manages supplier information, including entering, inquiring, modifying and deleting of supplier information
Involved Actor:	Administrator
Preconditions:	
Postconditions:	
Basic Path:	
Alternative Path:	

3.1.2.2 Entity Analysis

Conceptual Class Diagram



E1 - Store

Entity Name:	Store		
Entity ID:	E1		
Entity Description:	The place where items are sold		
Attribute Name	**Attribute Type**	**Attribute Description**	
Id	Integer	The Id of Store	
Name	String	The Name of Store	
Address	String	The Address of Store	
IsOpened	Boolean	The IsOpened of Store	
Relationship Name	**Related Entity**	**Relationship Type**	**Relationship Description**
AssociationCashdeskes	CashDesk	Association	One Store is linked with many CashDesk
Productcatalogs	ProductCatalog	Association	One Store is linked to many ProductCatalog
Items	Item	Association	One Store is linked to many Item
Cashiers	Cashier	Association	One Store is linked with many Cashier
Sales	Sale	Association	One Store is linked with many Sale

E2 - ProductCatalog

Entity Name:	ProductCatalog		
Entity ID:	E2		
Entity Description:	The catalogue of items		
Attribute Name	**Attribute Type**	**Attribute Description**	
Id	Integer	The Id of ProductCatalog	
Name	String	The Name of ProductCatalog	
Relationship Name	**Related Entity**	**Relationship Type**	**Relationship Description**
ContainedItems	Item	Association	One ProductCatalog is linked with many Item

E3 - CashDesk

Entity Name:	CashDesk		
Entity ID:	E3		
Entity Description:	The cash desk in store		
Attribute Name	**Attribute Type**	**Attribute Description**	
Id	Integer	The Id of CashDesk	
Name	String	The Name of CashDesk	
IsOpened	Boolean	The IsOpened of CashDesk	
Relationship Name	**Related Entity**	**Relationship Type**	**Relationship Description**
ContainedSales	Sale	Association	One CashDesk is linked with many Sale
BelongedStore	Store	Association	Many CashDesk are linked with one Store

E4 - Sale

Entity Name:	Sale		
Entity ID:	E4		
Entity Description:	The sales order for items		
Attribute Name	**Attribute Type**	**Attribute Description**	
Time	LocalDate	The Time of Sale	
IsComplete	Boolean	The IsComplete of Sale	
Amount	Real	The Amount of Sale	
IsReadytoPay	Boolean	The IsReadytoPay of Sale	
Relationship Name	**Related Entity**	**Relationship Type**	**Relationship Description**
Belongedstore	Store	Association	Many Sale are linked with one Store
BelongedCashDesk	CashDesk	Association	Many Sale are linked with one CashDesk
ContainedSalesLine	SalesLineItem	Association	One Sale is linked with many SalesLineItem
AssoicatedPayment	Payment	Association	One Sale is linked with one Payment

E5 - Cashier

Entity Name:	Cashier		
Entity ID:	E5		
Entity Description:	The cashier in store		
Attribute Name	**Attribute Type**	**Attribute Description**	
Id	Integer	The Id of Cashier	
Name	String	The Name of Cashier	
Relationship Name	**Related Entity**	**Relationship Type**	**Relationship Description**
WorkedStore	Store	Association	Many Cashier are linked with one Store

E6 - SalesLineItem

Entity Name:	SalesLineItem		
Entity ID:	E6		
Entity Description:	The sales order for a item		
Attribute Name	**Attribute Type**	**Attribute Description**	
Quantity	Integer	The Quantity of SalesLineItem	
Subamount	Real	The Subamount of SalesLineItem	
Relationship Name	**Related Entity**	**Relationship Type**	**Relationship Description**
BelongedSale	Sale	Association	Many SalesLineItem are linked with one Sale
BelongedItem	Item	Association	One SalesLineItem is linked to one Item

E7 - Item

Entity Name:	Item		
Entity ID:	E7		
Entity Description:	The item to be sold		
Attribute Name	**Attribute Type**	**Attribute Description**	
Barcode	Integer	The Barcode of Item	
Name	String	The Name of Item	
Price	Real	The Price of Item	
StockNumber	Integer	The StockNumber of Item	
OrderPrice	Real	The OrderPrice of Item	
Relationship Name	**Related Entity**	**Relationship Type**	**Relationship Description**
BelongedCatalog	ProductCatalog	Association	Many Item are linked with one ProductCatalog

E8 - Payment

Entity Name:	Payment		
Entity ID:	E8		
Entity Description:	The bill for the goods sold		
Attribute Name	**Attribute Type**	**Attribute Description**	
AmountTendered	Real	The AmountTendered of Payment	
Relationship Name	**Related Entity**	**Relationship Type**	**Relationship Description**
BelongedSale	Sale	Association	One Payment is linked with one Sale

E9 - CashPayment

Entity Name:	CashPayment	
Entity ID:	E9	
Entity Description:	Pay in cash	
Super Entity:	Payment	
Attribute Name	**Attribute Type**	**Attribute Description**
Balance	Real	The Balance of CashPayment

E10 - CardPayment

Entity Name:	CardPayment	
Entity ID:	E10	
Entity Description:	Pay in card	
Super Entity:	Payment	
Attribute Name	**Attribute Type**	**Attribute Description**
CardAccountNumber	String	The CardAccountNumber of CardPayment
ExpiryDate	LocalDate	The ExpiryDate of CardPayment

E11 - OrderEntry

Entity Name:	OrderEntry		
Entity ID:	E11		
Entity Description:	The purchase order for a item		
Attribute Name	**Attribute Type**	**Attribute Description**	
Quantity	Integer	The Quantity of OrderEntry	
SubAmount	Real	The SubAmount of OrderEntry	
Relationship Name	**Related Entity**	**Relationship Type**	**Relationship Description**
Item	Item	Association	One OrderEntry is linked to one Item

E12 - Supplier

Entity Name:	Supplier	
Entity ID:	E12	
Entity Description:	The supplier of items	
Attribute Name	**Attribute Type**	**Attribute Description**
Id	Integer	The Id of Supplier
Name	String	The Name of Supplier

E13 - OrderProduct

Entity Name:	OrderProduct		
Entity ID:	E13		
Entity Description:	The purchase order for items		
Attribute Name	**Attribute Type**	**Attribute Description**	
Id	Integer	The Id of OrderProduct	
Time	LocalDate	The Time of OrderProduct	
OrderStatus	[NEW|RECEIVED|REQUESTED]	The OrderStatus of OrderProduct	
Amount	Real	The Amount of OrderProduct	
Relationship Name	**Related Entity**	**Relationship Type**	**Relationship Description**
Supplier	Supplier	Association	One OrderProduct is linked to one Supplier
ContainedEntries	OrderEntry	Association	One OrderProduct is linked to many OrderEntry

3.1.2.3 System Interfaces

System Interfaces

SI1 - CoCoMESystem

System Interface Name:	CoCoMESystem
System Interface ID:	SI1
Description:	
Operation:	• openCashDesk • closeCashDesk • openStore • closeStore • changePrice • receiveOrderedProduct • listSuppliers • showStockReports
Temporary Variable	**Variable Description**
currentCashDesk	currentCashDesk is a object of CashDesk
currentStore	currentStore is a object of Store

SI2 - ThirdPartyServices

System Interface Name:	ThirdPartyServices
System Interface ID:	SI2
Description:	
Operation:	• thirdPartyCardPaymentService

SI3 - ProcessSaleService

System Interface Name:	ProcessSaleService
System Interface ID:	SI3
Description:	
Operation:	• makeNewSale • enterItem • endSale • makeCashPayment • makeCardPayment
Temporary Variable	**Variable Description**
currentSaleLine	currentSaleLine is a object of SalesLineItem
currentSale	currentSale is a object of Sale
currentPaymentMethod	currentPaymentMethod has several options: [CASH|CARD]

SI4 - ManageStoreCRUDService

System Interface Name:	ManageStoreCRUDService
System Interface ID:	SI4
Description:	
Operation:	• createStore • queryStore • modifyStore • deleteStore

SI5 - ManageProductCatalogCRUDService

System Interface Name:	ManageProductCatalogCRUDService
System Interface ID:	SI5
Description:	
Operation:	• createProductCatalog • queryProductCatalog • modifyProductCatalog • deleteProductCatalog

SI6 - ManageCashDeskCRUDService

System Interface Name:	ManageCashDeskCRUDService
System Interface ID:	SI6
Description:	
Operation:	• createCashDesk • queryCashDesk • modifyCashDesk • deleteCashDesk

SI7 - ManageCashierCRUDService

System Interface Name:	ManageCashierCRUDService
System Interface ID:	SI7
Description:	
Operation:	• createCashier • queryCashier • modifyCashier • deleteCashier

SI8 - ManageItemCRUDService

System Interface Name:	ManageItemCRUDService
System Interface ID:	SI8
Description:	
Operation:	• createItem • queryItem • modifyItem • deleteItem

SI9 - ManageSupplierCRUDService

System Interface Name:	ManageSupplierCRUDService
System Interface ID:	SI9
Description:	
Operation:	• createSupplier • querySupplier • modifySupplier • deleteSupplier

SI10 - CoCoMEOrderProducts

System Interface Name:	CoCoMEOrderProducts
System Interface ID:	SI10
Description:	
Operation:	• makeNewOrder • listAllOutOfStoreProducts • orderItem • chooseSupplier • placeOrder
Temporary Variable	**Variable Description**
currentOrderProduct	currentOrderProduct is a object of OrderProduct

System Operation Description

OP1 - openStore

Operation Name:	openStore
Operation ID:	OP1
Description:	
Service:	CoCoMESystem
Input:	name: *storeID*, type: Integer
Output Type:	Boolean
Definition:	*sto* is the object *s* in the instance set of class Store. *s* represents an object of class Store, and *s* meets: The attribute *Id* of the object *s* is equal to *storeID*
Preconditions:	1. *sto* exists 2. The attribute *IsOpened* of the object *sto* is equal to **false**
Postconditions:	1. The object currentStore became *sto* 2. The attribute *IsOpened* of the object *sto* became **true** 3. The return value was **true**

Contract of openStore:

```
Contract CoCoMESystem::openStore(storeID : Integer) : Boolean {
        /*
         * Generated by RM2Doc - Definition
         * sto is the object s in the instance set of class Store. s represents an object of class Store, and s meets:
         *     The attribute Id of the object s is equal to storeID
         */
        definition:
            sto:Store = Store.allInstance()->any(s:Store | s.Id = storeID)
        /*
         * Generated by RM2Doc - Precondition
         * sto exists
         * The attribute IsOpened of the object sto is equal to false
         */
        precondition:
            sto.oclIsUndefined() = false and
            sto.IsOpened = false
        /*
         * Generated by RM2Doc - Postcondition
         * The object currentStore became sto
         * The attribute IsOpened of the object sto became true
         * The return value was true
         */
        postcondition:
            self.currentStore = sto and
            sto.IsOpened = true and
            result = true
}
```

OP2 - closeStore

Operation Name:	closeStore
Operation ID:	OP2
Description:	
Service:	CoCoMESystem
Input:	name: *storeID*, type: Integer
Output Type:	Boolean
Definition:	*sto* is the object *s* in the instance set of class Store. *s* represents an object of class Store, and *s* meets: The attribute *Id* of the object *s* is equal to *storeID*
Preconditions:	1. *sto* exists 2. The attribute *IsOpened* of the object *sto* is equal to **true**
Postconditions:	1. The attribute *IsOpened* of the object *sto* became **false** 2. The return value was **true**

Contract of closeStore:

```
Contract CoCoMESystem::closeStore(storeID : Integer) : Boolean {
        /*
         * Generated by RM2Doc - Definition
         * sto is the object s in the instance set of class Store. s represents an object of class Store, and s meets:
         *     The attribute Id of the object s is equal to storeID
         */
        definition:
            sto:Store = Store.allInstance()->any(s:Store | s.Id = storeID)
        /*
         * Generated by RM2Doc - Precondition
         * sto exists
         * The attribute IsOpened of the object sto is equal to true
         */
        precondition:
            sto.oclIsUndefined() = false and
            sto.IsOpened = true
        /*
         * Generated by RM2Doc - Postcondition
         * The attribute IsOpened of the object sto became false
         * The return value was true
         */
        postcondition:
            sto.IsOpened = false and
            result = true
}
```

OP3 - openCashDesk

Operation Name:	openCashDesk
Operation ID:	OP3
Description:	
Service:	CoCoMESystem
Input:	name: *cashDeskID*, type: Integer
Output Type:	Boolean
Definition:	*cd* is the object *s* in the instance set of class CashDesk. *s* represents an object of class CashDesk, and *s* meets: The attribute *Id* of the object *s* is equal to *cashDeskID*
Preconditions:	1. *cd* exists 2. The attribute *IsOpened* of the object *cd* is equal to **false** 3. *currentStore* exists 4. The attribute *IsOpened* of the object *currentStore* is equal to **true**
Postconditions:	1. The object currentCashDesk became *cd* 2. The attribute *IsOpened* of the object *cd* became **true** 3. The return value was **true**

Contract of openCashDesk:

```
Contract CoCoMESystem::openCashDesk(cashDeskID : Integer) : Boolean {
        /*
         * Generated by RM2Doc - Definition
         * cd is the object s in the instance set of class CashDesk. s
represents an object of class CashDesk, and s meets:
         *     The attribute Id of the object s is equal to cashDeskID
         */
        definition:
            cd:CashDesk = CashDesk.allInstance()->any(s:CashDesk | s.Id =
cashDeskID)
        /*
         * Generated by RM2Doc - Precondition
         * cd exists
         * The attribute IsOpened of the object cd is equal to false
         * currentStore exists
         * The attribute IsOpened of the object currentStore is equal to true
         */
        precondition:
            cd.oclIsUndefined() = false and
            cd.IsOpened = false and
            currentStore.oclIsUndefined() = false and
            currentStore.IsOpened = true
        /*
         * Generated by RM2Doc - Postcondition
         * The object currentCashDesk became cd
         * The attribute IsOpened of the object cd became true
         * The return value was true
         */
        postcondition:
            self.currentCashDesk = cd and
            cd.IsOpened = true and
            result = true
}
```

OP4 - closeCashDesk

Operation Name:	closeCashDesk
Operation ID:	OP4
Description:	
Service:	CoCoMESystem
Input:	name: *cashDeskID*, type: Integer
Output Type:	Boolean
Definition:	*cd* is the object *s* in the instance set of class CashDesk. *s* represents an object of class CashDesk, and *s* meets: The attribute *Id* of the object *s* is equal to *cashDeskID*
Preconditions:	1. *cd* exists 2. The attribute *IsOpened* of the object *cd* is equal to **true** 3. *currentStore* exists 4. The attribute *IsOpened* of the object *currentStore* is equal to **true**
Postconditions:	1. The object currentCashDesk became *cd* 2. The attribute *IsOpened* of the object *cd* became **false** 3. The return value was **true**

Contract of closeCashDesk:

```
Contract CoCoMESystem::closeCashDesk(cashDeskID : Integer) : Boolean {
        /*
         * Generated by RM2Doc - Definition
         * cd is the object s in the instance set of class CashDesk. s
represents an object of class CashDesk, and s meets:
         *     The attribute Id of the object s is equal to cashDeskID
         */
        definition:
            cd:CashDesk = CashDesk.allInstance()->any(s:CashDesk | s.Id = cashDeskID)
        /*
         * Generated by RM2Doc - Precondition
         * cd exists
         * The attribute IsOpened of the object cd is equal to true
         * currentStore exists
         * The attribute IsOpened of the object currentStore is equal to true
         */
        precondition:
            cd.oclIsUndefined() = false and
            cd.IsOpened = true and
            currentStore.oclIsUndefined() = false and
            currentStore.IsOpened = true
        /*
         * Generated by RM2Doc - Postcondition
         * The object currentCashDesk became cd
         * The attribute IsOpened of the object cd became false
         * The return value was true
         */
        postcondition:
            self.currentCashDesk = cd and
            cd.IsOpened = false and
            result = true
}
```

OP5 - makeNewSale

Operation Name:	makeNewSale
Operation ID:	OP5
Description:	
Service:	ProcessSaleService
Input:	None
Output Type:	Boolean
Preconditions:	1. *currentCashDesk* exists 2. The attribute *IsOpened* of the object *currentCashDesk* is equal to **true** 3. (*currentSale* doesn't exist, or (*currentSale* exists, and the attribute *IsComplete* of the object *currentSale* is equal to **true**))
Postconditions:	1. *s* represented the object of class Sale 2. The object *s* was created 3. The object *s* was linked to the object *currentCashDesk* by *BelongedCashDesk* 4. The object *currentCashDesk* was linked to the object *s* by *ContainedSales* 5. The attribute *IsComplete* of the object *s* became **false** 6. The attribute *IsReadytoPay* of the object *s* became **false** 7. The object *s* was put into the instance set of class Sale 8. The object currentSale became *s* 9. The return value was **true**

Contract of makeNewSale:

```
Contract ProcessSaleService::makeNewSale() : Boolean {
        /*
         * Generated by RM2Doc - Precondition
         * currentCashDesk exists
         * The attribute IsOpened of the object currentCashDesk is equal to true
         * (currentSale doesn't exist, or (currentSale exists, and the attribute
IsComplete of the object currentSale is equal to true))
         */
        precondition:
            currentCashDesk.oclIsUndefined() = false and
            currentCashDesk.IsOpened = true and
            (currentSale.oclIsUndefined() = true or
                (currentSale.oclIsUndefined() = false and
                    currentSale.IsComplete = true
                )
            )
        /*
         * Generated by RM2Doc - Postcondition
         * s represented the object of class Sale
         * The object s was created
         * The object s was linked to the object currentCashDesk by
BelongedCashDesk
         * The object currentCashDesk was linked to the object s by
ContainedSales
         * The attribute IsComplete of the object s became false
         * The attribute IsReadytoPay of the object s became false
         * The object s was put into the instance set of class Sale
         * The object currentSale became s
         * The return value was true
         */
        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
            Sale.allInstance()->includes(s) and
            self.currentSale = s and
            result = true
}
```

OP6 - enterItem

Operation Name:	enterItem
Operation ID:	OP6
Description:	
Service:	ProcessSaleService
Input:	1. name: *barcode*, type: Integer 2. name: *quantity*, type: Integer
Output Type:	Boolean
Definition:	*item* is the object *i* in the instance set of class Item. *i* represents an object of class Item, and *i* meets: The attribute *Barcode* of the object *i* is equal to *barcode*
Preconditions:	1. *currentSale* exists 2. The attribute *IsComplete* of the object *currentSale* is equal to **false** 3. *item* exists 4. The attribute *StockNumber* of the object *item* is greater than **0**
Postconditions:	1. *sli* represented the object of class SalesLineItem 2. The object *sli* was created 3. The object currentSaleLine became *sli* 4. The object *sli* was linked to the object *currentSale* by *BelongedSale* 5. The object *currentSale* was linked to the object *sli* by *ContainedSalesLine* 6. The attribute *Quantity* of the object *sli* became *quantity* 7. The object *sli* was linked to the object *item* by *BelongedItem* 8. The attribute *StockNumber* of the object *item* became the previous value of the attribute *StockNumber* of the object *item* minus *quantity* 9. The attribute *Subamount* of the object *sli* became the attribute *Price* of the object *item* times *quantity* 10. The object *sli* was put into the instance set of class SalesLineItem 11. The return value was **true**

Contract of enterItem:

```
Contract ProcessSaleService::enterItem(barcode : Integer, quantity : Integer) :
Boolean {
        /*
```

```
         * Generated by RM2Doc - Definition
         * item is the object i in the instance set of class Item. i represents an object of class Item, and i meets:
         *     The attribute Barcode of the object i is equal to barcode
         */
        definition:
            item:Item = Item.allInstance()->any(i:Item | i.Barcode = barcode)
        /*
         * Generated by RM2Doc - Precondition
         * currentSale exists
         * The attribute IsComplete of the object currentSale is equal to false
         * item exists
         * The attribute StockNumber of the object item is greater than 0
         */
        precondition:
            currentSale.oclIsUndefined() = false and
            currentSale.IsComplete = false and
            item.oclIsUndefined() = false and
            item.StockNumber > 0
        /*
         * Generated by RM2Doc - Postcondition
         * sli represented the object of class SalesLineItem
         * The object sli was created
         * The object currentSaleLine became sli
         * The object sli was linked to the object currentSale by BelongedSale
         * The object currentSale was linked to the object sli by ContainedSalesLine
         * The attribute Quantity of the object sli became quantity
         * The object sli was linked to the object item by BelongedItem
         * The attribute StockNumber of the object item became the previous value of the attribute StockNumber of the object item minus quantity
         * The attribute Subamount of the object sli became the attribute Price of the object item times quantity
         * The object sli was put into the instance set of class SalesLineItem
         * The return value was true
         */
        postcondition:
            let sli:SalesLineItem in
            sli.oclIsNew() and
            self.currentSaleLine = sli and
            sli.BelongedSale = currentSale and
            currentSale.ContainedSalesLine->includes(sli) and
            sli.Quantity = quantity and
            sli.BelongedItem = item and
            item.StockNumber = item.StockNumber@pre - quantity and
            sli.Subamount = item.Price * quantity and
            SalesLineItem.allInstance()->includes(sli) and
            result = true
}
```

OP7 - endSale

Operation Name:	endSale
Operation ID:	OP7
Description:	
Service:	ProcessSaleService
Input:	None
Output Type:	Real
Definition:	1. *sls* is the Set of class SalesLineItem, including which *currentSale* is linked to 2. *sub* is the Set of Real, including the *Subamount* of each object in the set *sls*
Preconditions:	1. *currentSale* exists 2. The attribute *IsComplete* of the object *currentSale* is equal to **false** 3. The attribute *IsReadytoPay* of the object *currentSale* is equal to **false**
Postconditions:	1. The attribute *Amount* of the object *currentSale* became the sum of *sub* 2. The attribute *IsReadytoPay* of the object *currentSale* became **true** 3. The return value was the attribute *Amount* of the object *currentSale*

Contract of endSale:

```
Contract ProcessSaleService::endSale() : Real {
        /*
         * Generated by RM2Doc - Definition
         * sls is the Set of class SalesLineItem, including  which currentSale is linked to
         * sub is the Set of Real, including the Subamount of each object in the set sls
         */
        definition:
            sls:Set(SalesLineItem) = currentSale.ContainedSalesLine,
            sub:Set(Real) = sls->collect(s:SalesLineItem | s.Subamount)
        /*
         * Generated by RM2Doc - Precondition
         * currentSale exists
         * The attribute IsComplete of the object currentSale is equal to false
         * The attribute IsReadytoPay of the object currentSale is equal to false
         */
        precondition:
            currentSale.oclIsUndefined() = false and
            currentSale.IsComplete = false and
            currentSale.IsReadytoPay = false
        /*
```

```
         * Generated by RM2Doc - Postcondition
         * The attribute Amount of the object currentSale became the sum of sub
         * The attribute IsReadytoPay of the object currentSale became true
         * The return value was the attribute Amount of the object currentSale
         */
        postcondition:
            currentSale.Amount = sub.sum() and
            currentSale.IsReadytoPay = true and
            result = currentSale.Amount
}
```

OP8 - makeCashPayment

Operation Name:	makeCashPayment
Operation ID:	OP8
Description:	
Service:	ProcessSaleService
Input:	name: *amount*, type: Real
Output Type:	Boolean
Preconditions:	1. *currentSale* exists 2. The attribute *IsComplete* of the object *currentSale* is equal to **false** 3. The attribute *IsReadytoPay* of the object *currentSale* is equal to **true** 4. The *amount* is greater than or equal to the attribute *Amount* of the object *currentSale*
Postconditions:	1. *cp* represented the object of class CashPayment 2. The object *cp* was created 3. The attribute *AmountTendered* of the object *cp* became *amount* 4. The object *cp* was linked to the object *currentSale* by *BelongedSale* 5. The object *currentSale* was linked to the object *cp* by *AssoicatedPayment* 6. The object *currentSale* was linked to the object *currentStore* by *Belongedstore* 7. The object *currentStore* was linked to the object *currentSale* by *Sales* 8. The attribute *IsComplete* of the object *currentSale* became **true** 9. The attribute *Time* of the object *currentSale* was equal to *Now* 10. The attribute *Balance* of the object *cp* became *amount* minus the attribute *Amount* of the object *currentSale* 11. The object *cp* was put into the instance set of class CashPayment 12. The return value was **true**

Contract of makeCashPayment:

```
Contract ProcessSaleService::makeCashPayment(amount : Real) : Boolean {
        /*
         * Generated by RM2Doc - Precondition
         * currentSale exists
         * The attribute IsComplete of the object currentSale is equal to false
         * The attribute IsReadytoPay of the object currentSale is equal to true
         * The amount is greater than or equal to the attribute Amount of the
object currentSale
         */
        precondition:
            currentSale.oclIsUndefined() = false and
            currentSale.IsComplete = false and
            currentSale.IsReadytoPay = true and
            amount >= currentSale.Amount
        /*
         * Generated by RM2Doc - Postcondition
         * cp represented the object of class CashPayment
         * The object cp was created
         * The attribute AmountTendered of the object cp became amount
         * The object cp was linked to the object currentSale by BelongedSale
         * The object currentSale was linked to the object cp by
AssoicatedPayment
         * The object currentSale was linked to the object currentStore by
Belongedstore
         * The object currentStore was linked to the object currentSale by Sales
         * The attribute IsComplete of the object currentSale became true
         * The attribute Time of the object currentSale was equal to Now
         * The attribute Balance of the object cp became amount minus the
attribute Amount of the object currentSale
         * The object cp was put into the instance set of class CashPayment
         * The return value was true
         */
        postcondition:
            let cp:CashPayment in
            cp.oclIsNew() 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.isEqual(Now) and
            cp.Balance = amount - currentSale.Amount and
            CashPayment.allInstance()->includes(cp) and
            result = true
}
```

OP9 - makeCardPayment

Operation Name:	makeCardPayment
Operation ID:	OP9
Description:	
Service:	ProcessSaleService
Input:	1. name: *cardAccountNumber*, type: String 2. name: *expiryDate*, type: LocalDate 3. name: *fee*, type: Real
Output Type:	Boolean
Preconditions:	1. *currentSale* exists 2. The attribute *IsComplete* of the object *currentSale* is equal to **false** 3. The attribute *IsReadytoPay* of the object *currentSale* is equal to **true** 4. The system operation thirdPartyCardPaymentService is executed
Postconditions:	1. *cdp* represented the object of class CardPayment 2. The object *cdp* was created 3. The attribute *AmountTendered* of the object *cdp* became *fee* 4. The object *cdp* was linked to the object *currentSale* by *BelongedSale* 5. The object *currentSale* was linked to the object *cdp* by *AssoicatedPayment* 6. The attribute *CardAccountNumber* of the object *cdp* became *cardAccountNumber* 7. The attribute *ExpiryDate* of the object *cdp* became *expiryDate* 8. The object *cdp* was put into the instance set of class CardPayment 9. The object *currentSale* was linked to the object *currentStore* by *Belongedstore* 10. The object *currentStore* was linked to the object *currentSale* by *Sales* 11. The attribute *IsComplete* of the object *currentSale* became **true** 12. The attribute *Time* of the object *currentSale* was equal to *Now* 13. The return value was **true**

Contract of makeCardPayment:

```
Contract ProcessSaleService::makeCardPayment(cardAccountNumber : String,
expiryDate : Date, fee: Real) : Boolean {
        /*
```

```
     * Generated by RM2Doc - Precondition
     * currentSale exists
     * The attribute IsComplete of the object currentSale is equal to false
     * The attribute IsReadytoPay of the object currentSale is equal to true
     * The system operation thirdPartyCardPaymentService is executed
     */
    precondition:
        currentSale.oclIsUndefined() = false and
        currentSale.IsComplete = false and
        currentSale.IsReadytoPay = true and
        thirdPartyCardPaymentService(cardAccountNumber, expiryDate, fee)
    /*
     * Generated by RM2Doc - Postcondition
     * cdp represented the object of class CardPayment
     * The object cdp was created
     * The attribute AmountTendered of the object cdp became fee
     * The object cdp was linked to the object currentSale by BelongedSale
     * The object currentSale was linked to the object cdp by
AssoicatedPayment
     * The attribute CardAccountNumber of the object cdp became
cardAccountNumber
     * The attribute ExpiryDate of the object cdp became expiryDate
     * The object cdp was put into the instance set of class CardPayment
     * The object currentSale was linked to the object currentStore by
Belongedstore
     * The object currentStore was linked to the object currentSale by Sales
     * The attribute IsComplete of the object currentSale became true
     * The attribute Time of the object currentSale was equal to Now
     * The return value was true
     */
    postcondition:
        let cdp:CardPayment in
        cdp.oclIsNew() and
        cdp.AmountTendered = fee and
        cdp.BelongedSale = currentSale and
        currentSale.AssoicatedPayment = cdp and
        cdp.CardAccountNumber = cardAccountNumber and
        cdp.ExpiryDate = expiryDate and
        CardPayment.allInstance()->includes(cdp) and
        currentSale.Belongedstore = currentStore and
        currentStore.Sales->includes(currentSale) and
        currentSale.IsComplete = true and
        currentSale.Time.isEqual(Now) and
        result = true
}
```

OP10 - thirdPartyCardPaymentService

Operation Name:	thirdPartyCardPaymentService
Operation ID:	OP10
Description:	
Service:	ThirdPartyServices
Input:	1. name: *cardAccountNumber*, type: String 2. name: *expiryDate*, type: LocalDate 3. name: *fee*, type: Real
Output Type:	Boolean
Preconditions:	None
Postconditions:	The return value was **true**

Contract of thirdPartyCardPaymentService:

```
Contract  ThirdPartyServices::thirdPartyCardPaymentService(cardAccountNumber :
String, expiryDate : Date, fee : Real) : Boolean {
        /*
         * Generated by RM2Doc - Precondition
         * None
         */
        precondition:
            true
        /*
         * Generated by RM2Doc - Postcondition
         * The return value was true
         */
        postcondition:
            result = true
}
```

OP11 - makeNewOrder

Operation Name:	makeNewOrder
Operation ID:	OP11
Description:	
Service:	CoCoMEOrderProducts
Input:	name: *orderid*, type: Integer
Output Type:	Boolean
Preconditions:	None
Postconditions:	1. *op* represented the object of class OrderProduct 2. The object *op* was created 3. The attribute *OrderStatus* of the object *op* became **NEW** 4. The attribute *Id* of the object *op* became *orderid* 5. The attribute *Time* of the object *op* was equal to *Now* 6. The object *op* was put into the instance set of class OrderProduct 7. The object currentOrderProduct became *op* 8. The return value was **true**

Contract of makeNewOrder:

```
Contract CoCoMEOrderProducts::makeNewOrder(orderid : Integer) : Boolean {
        /*
         * Generated by RM2Doc - Precondition
         * None
         */
        precondition:
            true
        /*
         * Generated by RM2Doc - Postcondition
         * op represented the object of class OrderProduct
         * The object op was created
         * The attribute OrderStatus of the object op became NEW
         * The attribute Id of the object op became orderid
         * The attribute Time of the object op was equal to Now
         * The object op was put into the instance set of class OrderProduct
         * The object currentOrderProduct became op
         * The return value was true
         */
        postcondition:
            let op:OrderProduct in
            op.oclIsNew() and
            op.OrderStatus = OrderStatus::NEW and
            op.Id = orderid and
            op.Time.isEqual(Now) and
            OrderProduct.allInstance()->includes(op) and
            self.currentOrderProduct = op and
            result = true
}
```

OP12 - listAllOutOfStoreProducts

Operation Name:	listAllOutOfStoreProducts
Operation ID:	OP12
Description:	
Service:	CoCoMEOrderProducts
Input:	None
Output Type:	Set of Item
Preconditions:	None
Postconditions:	The return value was the set of class Item, including all *item* in the instance set of class Item. *item* represented an object of class Item, and *item* meet: The attribute *StockNumber* of the object *item* was equal to **0**

Contract of listAllOutOfStoreProducts:

```
Contract CoCoMEOrderProducts::listAllOutOfStoreProducts() : Set(Item) {
        /*
         * Generated by RM2Doc - Precondition
         * None
         */
        precondition:
            true
        /*
         * Generated by RM2Doc - Postcondition
         * The return value was the set of class Item, including all item in the
instance set of class Item. item represented an object of class Item, and item
meet:
         *     The attribute StockNumber of the object item was equal to 0
         */
        postcondition:
            result = Item.allInstance()->select(item:Item | item.StockNumber =
0)
}
```

OP13 - orderItem

Operation Name:	orderItem
Operation ID:	OP13
Description:	
Service:	CoCoMEOrderProducts
Input:	1. name: *barcode*, type: Integer 2. name: *quantity*, type: Integer
Output Type:	Boolean
Definition:	*item* is the object *i* in the instance set of class Item. *i* represents an object of class Item, and *i* meets: The attribute *Barcode* of the object *i* is equal to *barcode*
Preconditions:	*item* exists
Postconditions:	1. *order* represented the object of class OrderEntry 2. The object *order* was created 3. The attribute *Quantity* of the object *order* became *quantity* 4. The attribute *subAmount* of the object *order* became the attribute *OrderPrice* of the object *item* times *quantity* 5. The object *order* was linked to the object *item* by *Item* 6. The object *order* was put into the instance set of class OrderEntry 7. The object *currentOrderProduct* was linked to the object *order* by *ContainedEntries* 8. The return value was **true**

Contract of orderItem:

```
Contract CoCoMEOrderProducts::orderItem(barcode: Integer, quantity : Integer) :
Boolean {
        /*
         * Generated by RM2Doc - Definition
         * item is the object i in the instance set of class Item. i represents
an object of class Item, and i meets:
         *     The attribute Barcode of the object i is equal to barcode
         */
        definition:
            item:Item = Item.allInstance()->any(i:Item | i.Barcode = barcode)
        /*
         * Generated by RM2Doc - Precondition
         * item exists
         */
        precondition:
            item.oclIsUndefined() = false
        /*
         * Generated by RM2Doc - Postcondition
         * order represented the object of class OrderEntry
         * The object order was created
         * The attribute Quantity of the object order became quantity
         * The attribute subAmount of the object order became the attribute
OrderPrice of the object item times quantity
         * The object order was linked to the object item by Item
         * The object order was put into the instance set of class OrderEntry
         * The object currentOrderProduct was linked to the object order by
ContainedEntries
         * The return value was true
         */
        postcondition:
            let order:OrderEntry in
            order.oclIsNew() and
            order.Quantity = quantity and
            order.subAmount = item.OrderPrice * quantity and
            order.Item = item and
            OrderEntry.allInstance()->includes(order) and
            currentOrderProduct.ContainedEntries->includes(order) and
            result = true
}
```

OP14 - chooseSupplier

Operation Name:	chooseSupplier
Operation ID:	OP14
Description:	
Service:	CoCoMEOrderProducts
Input:	name: *supplierID*, type: Integer
Output Type:	Boolean
Definition:	*sup* is the object *s* in the instance set of class Supplier. *s* represents an object of class Supplier, and *s* meets: The attribute *Id* of the object *s* is equal to *supplierID*
Preconditions:	1. *sup* exists 2. *currentOrderProduct* exists
Postconditions:	1. The object *currentOrderProduct* was linked to the object *sup* by *Supplier* 2. The return value was **true**

Contract of chooseSupplier:

```
Contract CoCoMEOrderProducts::chooseSupplier(supplierID : Integer) : Boolean {
        /*
         * Generated by RM2Doc - Definition
         * sup is the object s in the instance set of class Supplier. s represents an object of class Supplier, and s meets:
         *     The attribute Id of the object s is equal to supplierID
         */
        definition:
            sup:Supplier = Supplier.allInstance()->any(s:Supplier | s.Id = supplierID)
        /*
         * Generated by RM2Doc - Precondition
         * sup exists
         * currentOrderProduct exists
         */
        precondition:
            sup.oclIsUndefined() = false and
            currentOrderProduct.oclIsUndefined() = false
        /*
         * Generated by RM2Doc - Postcondition
         * The object currentOrderProduct was linked to the object sup by Supplier
         * The return value was true
         */
        postcondition:
            currentOrderProduct.Supplier = sup and
            result = true
}
```

OP15 - placeOrder

Operation Name:	placeOrder
Operation ID:	OP15
Description:	
Service:	CoCoMEOrderProducts
Input:	None
Output Type:	Boolean
Preconditions:	*currentOrderProduct* exists
Postconditions:	1. The attribute *OrderStatus* of the object *currentOrderProduct* became **REQUESTED** 2. For each object of class OrderEntry in all objects which *currentOrderProduct* was linked to by *ContainedEntries*, *o* represented it(the object) and the following operations were performed: The attribute *Amount* of the object *currentOrderProduct* was equal to the previous value of the attribute *Amount* of the object *currentOrderProduct* plus the attribute *SubAmount* of the object *o* 3. The return value was **true**

Contract of placeOrder:

```
Contract CoCoMEOrderProducts::placeOrder() : Boolean {
        /*
         * Generated by RM2Doc - Precondition
         * currentOrderProduct exists
         */
        precondition:
            currentOrderProduct.oclIsUndefined() = false
        /*
         * Generated by RM2Doc - Postcondition
         * The attribute OrderStatus of the object currentOrderProduct became
REQUESTED
         * For each object of class OrderEntry in all objects which
currentOrderProduct was linked to by ContainedEntries, o represented it(the
object) and the following operations were performed:
         *     The attribute Amount of the object currentOrderProduct was equal
to the previous value of the attribute Amount of the object currentOrderProduct
plus the attribute SubAmount of the object o
         * The return value was true
         */
        postcondition:
            currentOrderProduct.OrderStatus = OrderStatus::REQUESTED and
            currentOrderProduct.ContainedEntries->forAll(o:OrderEntry |
                currentOrderProduct.Amount = currentOrderProduct.Amount@pre +
o.SubAmount)
            and
            result = true
}
```

OP16 - changePrice

Operation Name:	changePrice
Operation ID:	OP16
Description:	
Service:	CoCoMESystem
Input:	1. name: *barcode*, type: Integer 2. name: *newPrice*, type: Real
Output Type:	Boolean
Definition:	*item* is the object *i* in the instance set of class Item. *i* represents an object of class Item, and *i* meets: The attribute *Barcode* of the object *i* is equal to *barcode*
Preconditions:	*item* exists
Postconditions:	1. The attribute *Price* of the object *item* became *newPrice* 2. The return value was **true**

Contract of changePrice:

```
Contract CoCoMESystem::changePrice(barcode : Integer, newPrice : Real) : Boolean
{
        /*
         * Generated by RM2Doc - Definition
         * item is the object i in the instance set of class Item. i represents an object of class Item, and i meets:
         *     The attribute Barcode of the object i is equal to barcode
         */
        definition:
            item:Item = Item.allInstance()->any(i:Item | i.Barcode = barcode)
        /*
         * Generated by RM2Doc - Precondition
         * item exists
         */
        precondition:
            item.oclIsUndefined() = false
        /*
         * Generated by RM2Doc - Postcondition
         * The attribute Price of the object item became newPrice
         * The return value was true
         */
        postcondition:
            item.Price = newPrice and
            result = true
}
```

OP17 - receiveOrderedProduct

Operation Name:	receiveOrderedProduct
Operation ID:	OP17
Description:	
Service:	CoCoMESystem
Input:	name: *orderID*, type: Integer
Output Type:	Boolean
Definition:	*op* is the object *i* in the instance set of class OrderProduct. *i* represents an object of class OrderProduct, and *i* meets: The attribute *Id* of the object *i* is equal to *orderID*
Preconditions:	*op* exists
Postconditions:	1. The attribute *OrderStatus* of the object *op* became **RECEIVED** 2. For each object of class OrderEntry in all objects which *op* was linked to by *ContainedEntries*, *oe* represented it(the object) and the following operations were performed: The attribute *StockNumber* of the object *oe* was equal to the previous value of the attribute *StockNumber* of the object *oe* plus the attribute *Quantity* of the object *oe* 3. The return value was **true**

Contract of receiveOrderedProduct:

```
Contract CoCoMESystem::receiveOrderedProduct(orderID : Integer) : Boolean {
        /*
         * Generated by RM2Doc - Definition
         * op is the object i in the instance set of class OrderProduct. i
represents an object of class OrderProduct, and i meets:
         *     The attribute Id of the object i is equal to orderID
         */
        definition:
            op:OrderProduct = OrderProduct.allInstance()->any(i:OrderProduct |
i.Id = orderID)
        /*
         * Generated by RM2Doc - Precondition
         * op exists
         */
        precondition:
            op.oclIsUndefined() = false
        /*
         * Generated by RM2Doc - Postcondition
         * The attribute OrderStatus of the object op became RECEIVED
         * For each object of class OrderEntry in all objects which op was
linked to by ContainedEntries, oe represented it(the object) and the following
operations were performed:
         *     The attribute StockNumber of the object oe was equal to the
previous value of the attribute StockNumber of the object oe plus the attribute
Quantity of the object oe
         * The return value was true
         */
        postcondition:
            op.OrderStatus = OrderStatus::RECEIVED and
            op.ContainedEntries->forAll(oe:OrderEntry |
                oe.Item.StockNumber = oe.Item.StockNumber@pre + oe.Quantity)
            and
            result = true
}
```

OP18 - listSuppliers

Operation Name:	listSuppliers
Operation ID:	OP18
Description:	
Service:	CoCoMESystem
Input:	None
Output Type:	Set of Supplier
Preconditions:	None
Postconditions:	The return value was the instance set of class Supplier

Contract of listSuppliers:

```
Contract CoCoMESystem::listSuppliers() : Set(Supplier) {
        /*
         * Generated by RM2Doc - Precondition
         * None
         */
        precondition:
            true
        /*
         * Generated by RM2Doc - Postcondition
         * The return value was the instance set of class Supplier
         */
        postcondition:
            result = Supplier.allInstance()
}
```

OP19 - showStockReports

Operation Name:	showStockReports
Operation ID:	OP19
Description:	
Service:	CoCoMESystem
Input:	None
Output Type:	Set of Item
Preconditions:	None
Postconditions:	The return value was the instance set of class Item

Contract of showStockReports:

```
Contract CoCoMESystem::showStockReports() : Set(Item) {
        /*
         * Generated by RM2Doc - Precondition
         * None
         */
        precondition:
            true
        /*
         * Generated by RM2Doc - Postcondition
         * The return value was the instance set of class Item
         */
        postcondition:
            result = Item.allInstance()
}
```

OP20 - createStore

Operation Name:	createStore
Operation ID:	OP20
Description:	
Service:	ManageStoreCRUDService
Input:	1. name: *id*, type: Integer 2. name: *name*, type: String 3. name: *address*, type: String 4. name: *isopened*, type: Boolean
Output Type:	Boolean
Definition:	*store* is the object *sto* in the instance set of class Store. *sto* represents an object of class Store, and *sto* meets: The attribute *Id* of the object *sto* is equal to *id*
Preconditions:	*store* doesn't exist
Postconditions:	1. *sto* represented the object of class Store 2. The object *sto* was created 3. The attribute *Id* of the object *sto* became *id* 4. The attribute *Name* of the object *sto* became *name* 5. The attribute *Address* of the object *sto* became *address* 6. The attribute *IsOpened* of the object *sto* became *isopened* 7. The object *sto* was put into the instance set of class Store 8. The return value was **true**

Contract of createStore:

```
Contract  ManageStoreCRUDService::createStore(id : Integer, name : String,
address : String, isopened : Boolean) : Boolean {
        /*
         * Generated by RM2Doc - Definition
         * store is the object sto in the instance set of class Store. sto
represents an object of class Store, and sto meets:
         *     The attribute Id of the object sto is equal to id
         */
        definition:
            store:Store = Store.allInstance()->any(sto:Store | sto.Id = id)
        /*
         * Generated by RM2Doc - Precondition
         * store doesn't exist
         */
        precondition:
            store.oclIsUndefined() = true
        /*
         * Generated by RM2Doc - Postcondition
         * sto represented the object of class Store
         * The object sto was created
         * The attribute Id of the object sto became id
         * The attribute Name of the object sto became name
         * The attribute Address of the object sto became address
         * The attribute IsOpened of the object sto became isopened
         * The object sto was put into the instance set of class Store
         * The return value was true
         */
        postcondition:
            let sto:Store in
            sto.oclIsNew() and
            sto.Id = id and
            sto.Name = name and
            sto.Address = address and
            sto.IsOpened = isopened and
            Store.allInstance()->includes(sto) and
            result = true
}
```

OP21 - queryStore

Operation Name:	queryStore
Operation ID:	OP21
Description:	
Service:	ManageStoreCRUDService
Input:	name: *id*, type: Integer
Output Type:	Store
Definition:	*store* is the object *sto* in the instance set of class Store. *sto* represents an object of class Store, and *sto* meets: The attribute *Id* of the object *sto* is equal to *id*
Preconditions:	*store* exists
Postconditions:	The return value was *store*

Contract of queryStore:

```
Contract  ManageStoreCRUDService::queryStore(id : Integer) : Store {
        /*
         * Generated by RM2Doc - Definition
         * store is the object sto in the instance set of class Store. sto represents an object of class Store, and sto meets:
         *     The attribute Id of the object sto is equal to id
         */
        definition:
            store:Store = Store.allInstance()->any(sto:Store | sto.Id = id)
        /*
         * Generated by RM2Doc - Precondition
         * store exists
         */
        precondition:
            store.oclIsUndefined() = false
        /*
         * Generated by RM2Doc - Postcondition
         * The return value was store
         */
        postcondition:
            result = store
}
```

OP22 - modifyStore

Operation Name:	modifyStore
Operation ID:	OP22
Description:	
Service:	ManageStoreCRUDService
Input:	1. name: *id*, type: Integer 2. name: *name*, type: String 3. name: *address*, type: String 4. name: *isopened*, type: Boolean
Output Type:	Boolean
Definition:	*store* is the object *sto* in the instance set of class Store. *sto* represents an object of class Store, and *sto* meets: The attribute *Id* of the object *sto* is equal to *id*
Preconditions:	*store* exists
Postconditions:	1. The attribute *Id* of the object *store* became *id* 2. The attribute *Name* of the object *store* became *name* 3. The attribute *Address* of the object *store* became *address* 4. The attribute *IsOpened* of the object *store* became *isopened* 5. The return value was **true**

Contract of modifyStore:

```
Contract  ManageStoreCRUDService::modifyStore(id : Integer, name : String,
address : String, isopened : Boolean) : Boolean {
        /*
         * Generated by RM2Doc - Definition
         * store is the object sto in the instance set of class Store. sto
represents an object of class Store, and sto meets:
         *     The attribute Id of the object sto is equal to id
         */
        definition:
            store:Store = Store.allInstance()->any(sto:Store | sto.Id = id)
        /*
         * Generated by RM2Doc - Precondition
         * store exists
         */
        precondition:
            store.oclIsUndefined() = false
        /*
         * Generated by RM2Doc - Postcondition
         * The attribute Id of the object store became id
         * The attribute Name of the object store became name
         * The attribute Address of the object store became address
         * The attribute IsOpened of the object store became isopened
         * The return value was true
         */
        postcondition:
            store.Id = id and
            store.Name = name and
            store.Address = address and
            store.IsOpened = isopened and
            result = true
}
```

OP23 - deleteStore

Operation Name:	deleteStore
Operation ID:	OP23
Description:	
Service:	ManageStoreCRUDService
Input:	name: *id*, type: Integer
Output Type:	Boolean
Definition:	*store* is the object *sto* in the instance set of class Store. *sto* represents an object of class Store, and *sto* meets: The attribute *Id* of the object *sto* is equal to *id*
Preconditions:	1. *store* exists 2. The object *store* is in the instance set of class Store
Postconditions:	1. The object *store* was deleted from the instance set of class Store 2. The return value was **true**

Contract of deleteStore:

```
Contract  ManageStoreCRUDService::deleteStore(id : Integer) : Boolean {
        /*
         * Generated by RM2Doc - Definition
         * store is the object sto in the instance set of class Store. sto
represents an object of class Store, and sto meets:
         *     The attribute Id of the object sto is equal to id
         */
        definition:
            store:Store = Store.allInstance()->any(sto:Store | sto.Id = id)
        /*
```

```
         * Generated by RM2Doc - Precondition
         * store exists
         * The object store is in the instance set of class Store
         */
        precondition:
            store.oclIsUndefined() = false and
            Store.allInstance()->includes(store)
        /*
         * Generated by RM2Doc - Postcondition
         * The object store was deleted from the instance set of class Store
         * The return value was true
         */
        postcondition:
            Store.allInstance()->excludes(store) and
            result = true
}
```

OP24 - createProductCatalog

Operation Name:	createProductCatalog
Operation ID:	OP24
Description:	
Service:	ManageProductCatalogCRUDService
Input:	1. name: *id*, type: Integer 2. name: *name*, type: String
Output Type:	Boolean
Definition:	*productcatalog* is the object *pro* in the instance set of class ProductCatalog. *pro* represents an object of class ProductCatalog, and *pro* meets: The attribute *Id* of the object *pro* is equal to *id*
Preconditions:	*productcatalog* doesn't exist
Postconditions:	1. *pro* represented the object of class ProductCatalog 2. The object *pro* was created 3. The attribute *Id* of the object *pro* became *id* 4. The attribute *Name* of the object *pro* became *name* 5. The object *pro* was put into the instance set of class ProductCatalog 6. The return value was **true**

Contract of createProductCatalog:

```
Contract  ManageProductCatalogCRUDService::createProductCatalog(id : Integer, name : String) : Boolean {
        /*
         * Generated by RM2Doc - Definition
         * productcatalog is the object pro in the instance set of class ProductCatalog. pro represents an object of class ProductCatalog, and pro meets:
         *     The attribute Id of the object pro is equal to id
         */
        definition:
            productcatalog:ProductCatalog = ProductCatalog.allInstance()->any(pro:ProductCatalog | pro.Id = id)
        /*
         * Generated by RM2Doc - Precondition
         * productcatalog doesn't exist
         */
        precondition:
            productcatalog.oclIsUndefined() = true
        /*
         * Generated by RM2Doc - Postcondition
         * pro represented the object of class ProductCatalog
         * The object pro was created
         * The attribute Id of the object pro became id
         * The attribute Name of the object pro became name
         * The object pro was put into the instance set of class ProductCatalog
         * The return value was true
         */
        postcondition:
            let pro:ProductCatalog in
            pro.oclIsNew() and
            pro.Id = id and
            pro.Name = name and
            ProductCatalog.allInstance()->includes(pro) and
            result = true
}
```

OP25 - queryProductCatalog

Operation Name:	queryProductCatalog
Operation ID:	OP25
Description:	
Service:	ManageProductCatalogCRUDService
Input:	name: *id*, type: Integer
Output Type:	ProductCatalog
Definition:	*productcatalog* is the object *pro* in the instance set of class ProductCatalog. *pro* represents an object of class ProductCatalog, and *pro* meets: The attribute *Id* of the object *pro* is equal to *id*
Preconditions:	*productcatalog* exists
Postconditions:	The return value was *productcatalog*

Contract of queryProductCatalog:

```
Contract  ManageProductCatalogCRUDService::queryProductCatalog(id : Integer) : ProductCatalog {
        /*
         * Generated by RM2Doc - Definition
         * productcatalog is the object pro in the instance set of class ProductCatalog. pro represents an object of class ProductCatalog, and pro meets:
         *     The attribute Id of the object pro is equal to id
         */
        definition:
            productcatalog:ProductCatalog = ProductCatalog.allInstance()->any(pro:ProductCatalog | pro.Id = id)
        /*
         * Generated by RM2Doc - Precondition
         * productcatalog exists
         */
        precondition:
            productcatalog.oclIsUndefined() = false
        /*
         * Generated by RM2Doc - Postcondition
         * The return value was productcatalog
         */
        postcondition:
            result = productcatalog
}
```

OP26 - modifyProductCatalog

Operation Name:	modifyProductCatalog
Operation ID:	OP26
Description:	
Service:	ManageProductCatalogCRUDService
Input:	1. name: *id*, type: Integer 2. name: *name*, type: String
Output Type:	Boolean
Definition:	*productcatalog* is the object *pro* in the instance set of class ProductCatalog. *pro* represents an object of class ProductCatalog, and *pro* meets: The attribute *Id* of the object *pro* is equal to *id*
Preconditions:	*productcatalog* exists
Postconditions:	1. The attribute *Id* of the object *productcatalog* became *id* 2. The attribute *Name* of the object *productcatalog* became *name* 3. The return value was **true**

Contract of modifyProductCatalog:

```
Contract  ManageProductCatalogCRUDService::modifyProductCatalog(id : Integer,
name : String) : Boolean {
        /*
         * Generated by RM2Doc - Definition
         * productcatalog is the object pro in the instance set of class
ProductCatalog. pro represents an object of class ProductCatalog, and pro meets:
         *     The attribute Id of the object pro is equal to id
         */
        definition:
            productcatalog:ProductCatalog = ProductCatalog.allInstance()-
>any(pro:ProductCatalog | pro.Id = id)
        /*
         * Generated by RM2Doc - Precondition
         * productcatalog exists
         */
        precondition:
            productcatalog.oclIsUndefined() = false
        /*
         * Generated by RM2Doc - Postcondition
         * The attribute Id of the object productcatalog became id
         * The attribute Name of the object productcatalog became name
         * The return value was true
         */
        postcondition:
            productcatalog.Id = id and
            productcatalog.Name = name and
            result = true
}
```

OP27 - deleteProductCatalog

Operation Name:	deleteProductCatalog
Operation ID:	OP27
Description:	
Service:	ManageProductCatalogCRUDService
Input:	name: *id*, type: Integer
Output Type:	Boolean
Definition:	*productcatalog* is the object *pro* in the instance set of class ProductCatalog. *pro* represents an object of class ProductCatalog, and *pro* meets: The attribute *Id* of the object *pro* is equal to *id*
Preconditions:	1. *productcatalog* exists 2. The object *productcatalog* is in the instance set of class ProductCatalog
Postconditions:	1. The object *productcatalog* was deleted from the instance set of class ProductCatalog 2. The return value was **true**

Contract of deleteProductCatalog:

```
Contract  ManageProductCatalogCRUDService::deleteProductCatalog(id : Integer) :
Boolean {
        /*
         * Generated by RM2Doc - Definition
         * productcatalog is the object pro in the instance set of class
ProductCatalog. pro represents an object of class ProductCatalog, and pro meets:
         *     The attribute Id of the object pro is equal to id
         */
        definition:
            productcatalog:ProductCatalog = ProductCatalog.allInstance()-
>any(pro:ProductCatalog | pro.Id = id)
        /*
         * Generated by RM2Doc - Precondition
         * productcatalog exists
         * The object productcatalog is in the instance set of class
ProductCatalog
         */
        precondition:
            productcatalog.oclIsUndefined() = false and
            ProductCatalog.allInstance()->includes(productcatalog)
        /*
         * Generated by RM2Doc - Postcondition
         * The object productcatalog was deleted from the instance set of class
ProductCatalog
         * The return value was true
         */
        postcondition:
            ProductCatalog.allInstance()->excludes(productcatalog) and
            result = true
}
```

OP28 - createCashDesk

Operation Name:	createCashDesk
Operation ID:	OP28
Description:	
Service:	ManageCashDeskCRUDService
Input:	1. name: *id*, type: Integer 2. name: *name*, type: String 3. name: *isopened*, type: Boolean
Output Type:	Boolean
Definition:	*cashdesk* is the object *cas* in the instance set of class CashDesk. *cas* represents an object of class CashDesk, and *cas* meets: The attribute *Id* of the object *cas* is equal to *id*
Preconditions:	*cashdesk* doesn't exist
Postconditions:	1. *cas* represented the object of class CashDesk 2. The object *cas* was created 3. The attribute *Id* of the object *cas* became *id* 4. The attribute *Name* of the object *cas* became *name* 5. The attribute *IsOpened* of the object *cas* became *isopened* 6. The object *cas* was put into the instance set of class CashDesk 7. The return value was **true**

Contract of createCashDesk:

```
Contract  ManageCashDeskCRUDService::createCashDesk(id : Integer, name : String, isopened : Boolean) : Boolean {
        /*
         * Generated by RM2Doc - Definition
         * cashdesk is the object cas in the instance set of class CashDesk. cas represents an object of class CashDesk, and cas meets:
         *     The attribute Id of the object cas is equal to id
         */
        definition:
            cashdesk:CashDesk = CashDesk.allInstance()->any(cas:CashDesk | cas.Id = id)
        /*
         * Generated by RM2Doc - Precondition
         * cashdesk doesn't exist
         */
        precondition:
            cashdesk.oclIsUndefined() = true
        /*
         * Generated by RM2Doc - Postcondition
         * cas represented the object of class CashDesk
         * The object cas was created
         * The attribute Id of the object cas became id
         * The attribute Name of the object cas became name
         * The attribute IsOpened of the object cas became isopened
         * The object cas was put into the instance set of class CashDesk
         * The return value was true
         */
        postcondition:
            let cas:CashDesk in
            cas.oclIsNew() and
            cas.Id = id and
            cas.Name = name and
            cas.IsOpened = isopened and
            CashDesk.allInstance()->includes(cas) and
            result = true
}
```

OP29 - queryCashDesk

Operation Name:	queryCashDesk
Operation ID:	OP29
Description:	
Service:	ManageCashDeskCRUDService
Input:	name: *id*, type: Integer
Output Type:	CashDesk
Definition:	*cashdesk* is the object *cas* in the instance set of class CashDesk. *cas* represents an object of class CashDesk, and *cas* meets: The attribute *Id* of the object *cas* is equal to *id*
Preconditions:	*cashdesk* exists
Postconditions:	The return value was *cashdesk*

Contract of queryCashDesk:

```
Contract  ManageCashDeskCRUDService::queryCashDesk(id : Integer) : CashDesk {
        /*
         * Generated by RM2Doc - Definition
         * cashdesk is the object cas in the instance set of class CashDesk. cas represents an object of class CashDesk, and cas meets:
         *     The attribute Id of the object cas is equal to id
         */
        definition:
            cashdesk:CashDesk = CashDesk.allInstance()->any(cas:CashDesk | cas.Id = id)
        /*
         * Generated by RM2Doc - Precondition
         * cashdesk exists
         */
        precondition:
            cashdesk.oclIsUndefined() = false
        /*
         * Generated by RM2Doc - Postcondition
         * The return value was cashdesk
         */
        postcondition:
            result = cashdesk
}
```

OP30 - modifyCashDesk

Operation Name:	modifyCashDesk
Operation ID:	OP30
Description:	
Service:	ManageCashDeskCRUDService
Input:	1. name: *id*, type: Integer 2. name: *name*, type: String 3. name: *isopened*, type: Boolean
Output Type:	Boolean
Definition:	*cashdesk* is the object *cas* in the instance set of class CashDesk. *cas* represents an object of class CashDesk, and *cas* meets: The attribute *Id* of the object *cas* is equal to *id*
Preconditions:	*cashdesk* exists
Postconditions:	1. The attribute *Id* of the object *cashdesk* became *id* 2. The attribute *Name* of the object *cashdesk* became *name* 3. The attribute *IsOpened* of the object *cashdesk* became *isopened* 4. The return value was **true**

Contract of modifyCashDesk:

```
Contract  ManageCashDeskCRUDService::modifyCashDesk(id : Integer, name : String,
isopened : Boolean) : Boolean {
        /*
         * Generated by RM2Doc - Definition
         * cashdesk is the object cas in the instance set of class CashDesk. cas
represents an object of class CashDesk, and cas meets:
         *     The attribute Id of the object cas is equal to id
         */
        definition:
            cashdesk:CashDesk = CashDesk.allInstance()->any(cas:CashDesk |
cas.Id = id)
        /*
         * Generated by RM2Doc - Precondition
         * cashdesk exists
         */
        precondition:
            cashdesk.oclIsUndefined() = false
        /*
         * Generated by RM2Doc - Postcondition
         * The attribute Id of the object cashdesk became id
         * The attribute Name of the object cashdesk became name
         * The attribute IsOpened of the object cashdesk became isopened
         * The return value was true
         */
        postcondition:
            cashdesk.Id = id and
            cashdesk.Name = name and
            cashdesk.IsOpened = isopened and
            result = true
}
```

OP31 - deleteCashDesk

Operation Name:	deleteCashDesk
Operation ID:	OP31
Description:	
Service:	ManageCashDeskCRUDService
Input:	name: *id*, type: Integer
Output Type:	Boolean
Definition:	*cashdesk* is the object *cas* in the instance set of class CashDesk. *cas* represents an object of class CashDesk, and *cas* meets: The attribute *Id* of the object *cas* is equal to *id*
Preconditions:	1. *cashdesk* exists 2. The object *cashdesk* is in the instance set of class CashDesk
Postconditions:	1. The object *cashdesk* was deleted from the instance set of class CashDesk 2. The return value was **true**

Contract of deleteCashDesk:

```
Contract  ManageCashDeskCRUDService::deleteCashDesk(id : Integer) : Boolean {
        /*
         * Generated by RM2Doc - Definition
         * cashdesk is the object cas in the instance set of class CashDesk. cas
represents an object of class CashDesk, and cas meets:
         *     The attribute Id of the object cas is equal to id
         */
        definition:
            cashdesk:CashDesk = CashDesk.allInstance()->any(cas:CashDesk |
cas.Id = id)
        /*
         * Generated by RM2Doc - Precondition
         * cashdesk exists
         * The object cashdesk is in the instance set of class CashDesk
         */
        precondition:
            cashdesk.oclIsUndefined() = false and
            CashDesk.allInstance()->includes(cashdesk)
        /*
         * Generated by RM2Doc - Postcondition
         * The object cashdesk was deleted from the instance set of class
CashDesk
         * The return value was true
         */
        postcondition:
            CashDesk.allInstance()->excludes(cashdesk) and
            result = true
}
```

OP32 - createCashier

Operation Name:	createCashier
Operation ID:	OP32
Description:	
Service:	ManageCashierCRUDService
Input:	1. name: *id*, type: Integer 2. name: *name*, type: String
Output Type:	Boolean
Definition:	*cashier* is the object *cas* in the instance set of class Cashier. *cas* represents an object of class Cashier, and *cas* meets: The attribute *Id* of the object *cas* is equal to *id*
Preconditions:	*cashier* doesn't exist
Postconditions:	1. *cas* represented the object of class Cashier 2. The object *cas* was created 3. The attribute *Id* of the object *cas* became *id* 4. The attribute *Name* of the object *cas* became *name* 5. The object *cas* was put into the instance set of class Cashier 6. The return value was **true**

Contract of createCashier:

```
Contract  ManageCashierCRUDService::createCashier(id : Integer, name : String) :
Boolean {
        /*
         * Generated by RM2Doc - Definition
         * cashier is the object cas in the instance set of class Cashier. cas
represents an object of class Cashier, and cas meets:
         *     The attribute Id of the object cas is equal to id
         */
        definition:
            cashier:Cashier = Cashier.allInstance()->any(cas:Cashier | cas.Id =
id)
        /*
         * Generated by RM2Doc - Precondition
         * cashier doesn't exist
         */
        precondition:
            cashier.oclIsUndefined() = true
        /*
         * Generated by RM2Doc - Postcondition
         * cas represented the object of class Cashier
         * The object cas was created
         * The attribute Id of the object cas became id
         * The attribute Name of the object cas became name
         * The object cas was put into the instance set of class Cashier
         * The return value was true
         */
        postcondition:
            let cas:Cashier in
            cas.oclIsNew() and
            cas.Id = id and
            cas.Name = name and
            Cashier.allInstance()->includes(cas) and
            result = true
}
```

OP33 - queryCashier

Operation Name:	queryCashier
Operation ID:	OP33
Description:	
Service:	ManageCashierCRUDService
Input:	name: *id*, type: Integer
Output Type:	Cashier
Definition:	*cashier* is the object *cas* in the instance set of class Cashier. *cas* represents an object of class Cashier, and *cas* meets: The attribute *Id* of the object *cas* is equal to *id*
Preconditions:	*cashier* exists
Postconditions:	The return value was *cashier*

Contract of queryCashier:

```
Contract  ManageCashierCRUDService::queryCashier(id : Integer) : Cashier {
        /*
         * Generated by RM2Doc - Definition
         * cashier is the object cas in the instance set of class Cashier. cas represents an object of class Cashier, and cas meets:
         *     The attribute Id of the object cas is equal to id
         */
        definition:
            cashier:Cashier = Cashier.allInstance()->any(cas:Cashier | cas.Id = id)
        /*
         * Generated by RM2Doc - Precondition
         * cashier exists
         */
        precondition:
            cashier.oclIsUndefined() = false
        /*
         * Generated by RM2Doc - Postcondition
         * The return value was cashier
         */
        postcondition:
            result = cashier
}
```

OP34 - modifyCashier

Operation Name:	modifyCashier
Operation ID:	OP34
Description:	
Service:	ManageCashierCRUDService
Input:	1. name: *id*, type: Integer 2. name: *name*, type: String
Output Type:	Boolean
Definition:	*cashier* is the object *cas* in the instance set of class Cashier. *cas* represents an object of class Cashier, and *cas* meets: The attribute *Id* of the object *cas* is equal to *id*
Preconditions:	*cashier* exists
Postconditions:	1. The attribute *Id* of the object *cashier* became *id* 2. The attribute *Name* of the object *cashier* became *name* 3. The return value was **true**

Contract of modifyCashier:

```
Contract  ManageCashierCRUDService::modifyCashier(id : Integer, name : String) :
Boolean {
        /*
         * Generated by RM2Doc - Definition
         * cashier is the object cas in the instance set of class Cashier. cas
represents an object of class Cashier, and cas meets:
         *     The attribute Id of the object cas is equal to id
         */
        definition:
            cashier:Cashier = Cashier.allInstance()->any(cas:Cashier | cas.Id =
id)
        /*
         * Generated by RM2Doc - Precondition
         * cashier exists
         */
        precondition:
            cashier.oclIsUndefined() = false
        /*
         * Generated by RM2Doc - Postcondition
         * The attribute Id of the object cashier became id
         * The attribute Name of the object cashier became name
         * The return value was true
         */
        postcondition:
            cashier.Id = id and
            cashier.Name = name and
            result = true
}
```

OP35 - deleteCashier

Operation Name:	deleteCashier
Operation ID:	OP35
Description:	
Service:	ManageCashierCRUDService
Input:	name: *id*, type: Integer
Output Type:	Boolean
Definition:	*cashier* is the object *cas* in the instance set of class Cashier. *cas* represents an object of class Cashier, and *cas* meets: The attribute *Id* of the object *cas* is equal to *id*
Preconditions:	1. *cashier* exists 2. The object *cashier* is in the instance set of class Cashier
Postconditions:	1. The object *cashier* was deleted from the instance set of class Cashier 2. The return value was **true**

Contract of deleteCashier:

```
Contract  ManageCashierCRUDService::deleteCashier(id : Integer) : Boolean {
        /*
         * Generated by RM2Doc - Definition
         * cashier is the object cas in the instance set of class Cashier. cas
represents an object of class Cashier, and cas meets:
         *     The attribute Id of the object cas is equal to id
         */
        definition:
            cashier:Cashier = Cashier.allInstance()->any(cas:Cashier | cas.Id =
id)
        /*
         * Generated by RM2Doc - Precondition
         * cashier exists
         * The object cashier is in the instance set of class Cashier
         */
        precondition:
            cashier.oclIsUndefined() = false and
            Cashier.allInstance()->includes(cashier)
        /*
```

```
         * Generated by RM2Doc - Postcondition
         * The object cashier was deleted from the instance set of class Cashier
         * The return value was true
         */
        postcondition:
            Cashier.allInstance()->excludes(cashier) and
            result = true
}
```

OP36 - createItem

Operation Name:	createItem
Operation ID:	OP36
Description:	
Service:	ManageItemCRUDService
Input:	1. name: *barcode*, type: Integer 2. name: *name*, type: String 3. name: *price*, type: Real 4. name: *stocknumber*, type: Integer 5. name: *orderprice*, type: Real
Output Type:	Boolean
Definition:	*item* is the object *ite* in the instance set of class Item. *ite* represents an object of class Item, and *ite* meets: The attribute *Barcode* of the object *ite* is equal to *barcode*
Preconditions:	*item* doesn't exist
Postconditions:	1. *ite* represented the object of class Item 2. The object *ite* was created 3. The attribute *Barcode* of the object *ite* became *barcode* 4. The attribute *Name* of the object *ite* became *name* 5. The attribute *Price* of the object *ite* became *price* 6. The attribute *StockNumber* of the object *ite* became *stocknumber* 7. The attribute *OrderPrice* of the object *ite* became *orderprice* 8. The object *ite* was put into the instance set of class Item 9. The return value was **true**

Contract of createItem:

```
Contract  ManageItemCRUDService::createItem(barcode : Integer, name : String,
price : Real, stocknumber : Integer, orderprice : Real) : Boolean {
        /*
         * Generated by RM2Doc - Definition
         * item is the object ite in the instance set of class Item. ite
represents an object of class Item, and ite meets:
         *     The attribute Barcode of the object ite is equal to barcode
         */
        definition:
            item:Item = Item.allInstance()->any(ite:Item | ite.Barcode =
barcode)
        /*
         * Generated by RM2Doc - Precondition
         * item doesn't exist
         */
        precondition:
            item.oclIsUndefined() = true
        /*
         * Generated by RM2Doc - Postcondition
         * ite represented the object of class Item
         * The object ite was created
         * The attribute Barcode of the object ite became barcode
         * The attribute Name of the object ite became name
         * The attribute Price of the object ite became price
         * The attribute StockNumber of the object ite became stocknumber
         * The attribute OrderPrice of the object ite became orderprice
         * The object ite was put into the instance set of class Item
         * The return value was true
         */
        postcondition:
            let ite:Item in
            ite.oclIsNew() and
            ite.Barcode = barcode and
            ite.Name = name and
            ite.Price = price and
            ite.StockNumber = stocknumber and
            ite.OrderPrice = orderprice and
            Item.allInstance()->includes(ite) and
            result = true
}
```

OP37 - queryItem

Operation Name:	queryItem
Operation ID:	OP37
Description:	
Service:	ManageItemCRUDService
Input:	name: *barcode*, type: Integer
Output Type:	Item
Definition:	*item* is the object *ite* in the instance set of class Item. *ite* represents an object of class Item, and *ite* meets: The attribute *Barcode* of the object *ite* is equal to *barcode*
Preconditions:	*item* exists
Postconditions:	The return value was *item*

Contract of queryItem:

```
Contract  ManageItemCRUDService::queryItem(barcode : Integer) : Item {
        /*
         * Generated by RM2Doc - Definition
         * item is the object ite in the instance set of class Item. ite
represents an object of class Item, and ite meets:
         *     The attribute Barcode of the object ite is equal to barcode
         */
        definition:
            item:Item = Item.allInstance()->any(ite:Item | ite.Barcode =
barcode)
        /*
         * Generated by RM2Doc - Precondition
         * item exists
         */
        precondition:
            item.oclIsUndefined() = false
        /*
         * Generated by RM2Doc - Postcondition
         * The return value was item
         */
        postcondition:
            result = item
}
```

OP38 - modifyItem

Operation Name:	modifyItem
Operation ID:	OP38
Description:	
Service:	ManageItemCRUDService
Input:	1. name: *barcode*, type: Integer 2. name: *name*, type: String 3. name: *price*, type: Real 4. name: *stocknumber*, type: Integer 5. name: *orderprice*, type: Real
Output Type:	Boolean
Definition:	*item* is the object *ite* in the instance set of class Item. *ite* represents an object of class Item, and *ite* meets: The attribute *Barcode* of the object *ite* is equal to *barcode*
Preconditions:	*item* exists
Postconditions:	1. The attribute *Barcode* of the object *item* became *barcode* 2. The attribute *Name* of the object *item* became *name* 3. The attribute *Price* of the object *item* became *price* 4. The attribute *StockNumber* of the object *item* became *stocknumber* 5. The attribute *OrderPrice* of the object *item* became *orderprice* 6. The return value was **true**

Contract of modifyItem:

```
Contract  ManageItemCRUDService::modifyItem(barcode : Integer, name : String,
price : Real, stocknumber : Integer, orderprice : Real) : Boolean {
        /*
         * Generated by RM2Doc - Definition
         * item is the object ite in the instance set of class Item. ite
represents an object of class Item, and ite meets:
         *     The attribute Barcode of the object ite is equal to barcode
         */
        definition:
            item:Item = Item.allInstance()->any(ite:Item | ite.Barcode =
barcode)
        /*
         * Generated by RM2Doc - Precondition
         * item exists
         */
        precondition:
            item.oclIsUndefined() = false
        /*
         * Generated by RM2Doc - Postcondition
         * The attribute Barcode of the object item became barcode
         * The attribute Name of the object item became name
         * The attribute Price of the object item became price
         * The attribute StockNumber of the object item became stocknumber
         * The attribute OrderPrice of the object item became orderprice
         * The return value was true
         */
        postcondition:
            item.Barcode = barcode and
            item.Name = name and
            item.Price = price and
            item.StockNumber = stocknumber and
            item.OrderPrice = orderprice and
            result = true
}
```

OP39 - deleteItem

Operation Name:	deleteItem
Operation ID:	OP39
Description:	
Service:	ManageItemCRUDService
Input:	name: *barcode*, type: Integer
Output Type:	Boolean
Definition:	*item* is the object *ite* in the instance set of class Item. *ite* represents an object of class Item, and *ite* meets: The attribute *Barcode* of the object *ite* is equal to *barcode*
Preconditions:	1. *item* exists 2. The object *item* is in the instance set of class Item
Postconditions:	1. The object *item* was deleted from the instance set of class Item 2. The return value was **true**

Contract of deleteItem:

```
Contract  ManageItemCRUDService::deleteItem(barcode : Integer) : Boolean {
        /*
         * Generated by RM2Doc - Definition
         * item is the object ite in the instance set of class Item. ite
represents an object of class Item, and ite meets:
         *     The attribute Barcode of the object ite is equal to barcode
         */
        definition:
            item:Item = Item.allInstance()->any(ite:Item | ite.Barcode =
barcode)
        /*
         * Generated by RM2Doc - Precondition
         * item exists
         * The object item is in the instance set of class Item
         */
        precondition:
            item.oclIsUndefined() = false and
            Item.allInstance()->includes(item)
        /*
         * Generated by RM2Doc - Postcondition
         * The object item was deleted from the instance set of class Item
         * The return value was true
         */
        postcondition:
            Item.allInstance()->excludes(item) and
            result = true
}
```

OP40 - createSupplier

Operation Name:	createSupplier
Operation ID:	OP40
Description:	
Service:	ManageSupplierCRUDService
Input:	1. name: *id*, type: Integer 2. name: *name*, type: String
Output Type:	Boolean
Definition:	*supplier* is the object *sup* in the instance set of class Supplier. *sup* represents an object of class Supplier, and *sup* meets: The attribute *Id* of the object *sup* is equal to *id*
Preconditions:	*supplier* doesn't exist
Postconditions:	1. *sup* represented the object of class Supplier 2. The object *sup* was created 3. The attribute *Id* of the object *sup* became *id* 4. The attribute *Name* of the object *sup* became *name* 5. The object *sup* was put into the instance set of class Supplier 6. The return value was **true**

Contract of createSupplier:

```
Contract  ManageSupplierCRUDService::createSupplier(id : Integer, name : String)
: Boolean {
        /*
         * Generated by RM2Doc - Definition
         * supplier is the object sup in the instance set of class Supplier. sup
represents an object of class Supplier, and sup meets:
         *     The attribute Id of the object sup is equal to id
         */
        definition:
            supplier:Supplier = Supplier.allInstance()->any(sup:Supplier |
sup.Id = id)
        /*
         * Generated by RM2Doc - Precondition
         * supplier doesn't exist
         */
        precondition:
            supplier.oclIsUndefined() = true
        /*
         * Generated by RM2Doc - Postcondition
         * sup represented the object of class Supplier
         * The object sup was created
         * The attribute Id of the object sup became id
         * The attribute Name of the object sup became name
         * The object sup was put into the instance set of class Supplier
         * The return value was true
         */
        postcondition:
            let sup:Supplier in
            sup.oclIsNew() and
            sup.Id = id and
            sup.Name = name and
            Supplier.allInstance()->includes(sup) and
            result = true
}
```

OP41 - querySupplier

Operation Name:	querySupplier
Operation ID:	OP41
Description:	
Service:	ManageSupplierCRUDService
Input:	name: *id*, type: Integer
Output Type:	Supplier
Definition:	*supplier* is the object *sup* in the instance set of class Supplier. *sup* represents an object of class Supplier, and *sup* meets: The attribute *Id* of the object *sup* is equal to *id*
Preconditions:	*supplier* exists
Postconditions:	The return value was *supplier*

Contract of querySupplier:

```
Contract  ManageSupplierCRUDService::querySupplier(id : Integer) : Supplier {
        /*
         * Generated by RM2Doc - Definition
         * supplier is the object sup in the instance set of class Supplier. sup represents an object of class Supplier, and sup meets:
         *     The attribute Id of the object sup is equal to id
         */
        definition:
            supplier:Supplier = Supplier.allInstance()->any(sup:Supplier | sup.Id = id)
        /*
         * Generated by RM2Doc - Precondition
         * supplier exists
         */
        precondition:
            supplier.oclIsUndefined() = false
        /*
         * Generated by RM2Doc - Postcondition
         * The return value was supplier
         */
        postcondition:
            result = supplier
}
```

OP42 - modifySupplier

Operation Name:	modifySupplier
Operation ID:	OP42
Description:	
Service:	ManageSupplierCRUDService
Input:	1. name: *id*, type: Integer 2. name: *name*, type: String
Output Type:	Boolean
Definition:	*supplier* is the object *sup* in the instance set of class Supplier. *sup* represents an object of class Supplier, and *sup* meets: The attribute *Id* of the object *sup* is equal to *id*
Preconditions:	*supplier* exists
Postconditions:	1. The attribute *Id* of the object *supplier* became *id* 2. The attribute *Name* of the object *supplier* became *name* 3. The return value was **true**

Contract of modifySupplier:

```
Contract  ManageSupplierCRUDService::modifySupplier(id : Integer, name : String)
: Boolean {
        /*
         * Generated by RM2Doc - Definition
         * supplier is the object sup in the instance set of class Supplier. sup
represents an object of class Supplier, and sup meets:
         *     The attribute Id of the object sup is equal to id
         */
        definition:
            supplier:Supplier = Supplier.allInstance()->any(sup:Supplier |
sup.Id = id)
```

```
        /*
         * Generated by RM2Doc - Precondition
         * supplier exists
         */
        precondition:
            supplier.oclIsUndefined() = false
        /*
         * Generated by RM2Doc - Postcondition
         * The attribute Id of the object supplier became id
         * The attribute Name of the object supplier became name
         * The return value was true
         */
        postcondition:
            supplier.Id = id and
            supplier.Name = name and
            result = true
}
```

OP43 - deleteSupplier

Operation Name:	deleteSupplier
Operation ID:	OP43
Description:	
Service:	ManageSupplierCRUDService
Input:	name: *id*, type: Integer
Output Type:	Boolean
Definition:	*supplier* is the object *sup* in the instance set of class Supplier. *sup* represents an object of class Supplier, and *sup* meets: The attribute *Id* of the object *sup* is equal to *id*
Preconditions:	1. *supplier* exists 2. The object *supplier* is in the instance set of class Supplier
Postconditions:	1. The object *supplier* was deleted from the instance set of class Supplier 2. The return value was **true**

Contract of deleteSupplier:

```
Contract  ManageSupplierCRUDService::deleteSupplier(id : Integer) : Boolean {
        /*
         * Generated by RM2Doc - Definition
         * supplier is the object sup in the instance set of class Supplier. sup
represents an object of class Supplier, and sup meets:
         *     The attribute Id of the object sup is equal to id
         */
        definition:
            supplier:Supplier = Supplier.allInstance()->any(sup:Supplier |
sup.Id = id)
        /*
         * Generated by RM2Doc - Precondition
         * supplier exists
         * The object supplier is in the instance set of class Supplier
         */
        precondition:
            supplier.oclIsUndefined() = false and
            Supplier.allInstance()->includes(supplier)
        /*
         * Generated by RM2Doc - Postcondition
         * The object supplier was deleted from the instance set of class
Supplier
         * The return value was true
         */
        postcondition:
            Supplier.allInstance()->excludes(supplier) and
            result = true
}
```

3.2 External interface requirements

User interfaces
Hardware interfaces
Software interfaces
Communications interfaces

3.3 Performance requirements

3.3.1 Static numerical requirements

This subsection should specify both the static and the dynamic numerical requirements placed on the software or on human interaction with the software as a whole. Static numerical requirements may include the following:

- a) The number of terminals to be supported;
- b) The number of simultaneous users to be supported;
- c) Amount and type of information to be handled.

3.3.2 Dynamic numerical requirements

Dynamic numerical requirements may include, for example, the numbers of transactions and tasks and the amount of data to be processed within certain time periods for both normal and peak workload conditions.

All of these requirements should be stated in measurable terms.

For example,

- *95% of the transactions shall be processed in less than 1 s.*

rather than,

- *An operator shall not have to wait for the transaction to complete.*

NOTE:Numerical limits applied to one specifific function are normally specified as part of the processing subparagraph description of that function.

3.4 Design constraints

3.4.1 Standards compliance

This subsection should specify the requirements derived from existing standards or regulations. They may include the following:

- a) Report format;
- b) Data naming;
- c) Accounting procedures;
- d) Audit tracing.

For example, this could specify the requirement for software to trace processing activity. Such traces are needed for some applications to meet minimum regulatory or financial standards. An audit trace requirement may, for example, state that all changes to a payroll database must be recorded in a trace file with before and after values.

3.4.2 Hardware limitations

3.5 Software system attributes

3.5.1 Reliability

This should specify the factors required to establish the required reliability of the software system at time of delivery.

3.5.2 Availability

This should specify the factors required to guarantee a defined availability level for the entire system such as checkpoint, recovery, and restart.

3.5.3 Security

This should specify the factors that protect the software from accidental or malicious access, use, modification, destruction, or disclosure. Specific requirements in this area could include the need to

- a) Utilize certain cryptographical techniques;
- b) Keep specific log or history data sets;
- c) Assign certain functions to different modules;
- d) Restrict communications between some areas of the program;
- e) Check data integrity for critical variables.

3.5.4 Maintainability

This should specify attributes of software that relate to the ease of maintenance of the software itself. There may be some requirement for certain modularity, interfaces, complexity, etc. Requirements should not be placed here just because they are thought to be good design practices.

3.5.5 Portability

This should specify attributes of software that relate to the ease of porting the software to other host machines and/or operating systems. This may include the following:

- a) Percentage of components with host-dependent code;
- b) Percentage of code that is host dependent;
- c) Use of a proven portable language;
- d) Use of a particular compiler or language subset;
- e) Use of a particular operating system.

3.6 Other requirements

3.6.1 Logical database requirements

This should specify the logical requirements for any information that is to be placed into a database. This may include the following:

- a) Types of information used by various functions;
- b) Frequency of use;
- c) Accessing capabilities;
- d) Data entities and their relationships;
- e) Integrity constraints;
- f) Data retention requirements.