# CoCoME Specification in $\mathcal{CL}$

Stephen Fenech

December 4, 2008

## 1 Introduction

The Common Component Modelling Example (CoCoME) is based on a Trading System that handles the sales and inventory of a Store chain. The example is defined into 8 Use Cases that describe the main processes. The Use Cases span from selling items at a cash desk to the exchange of product between stores.

The CoCoME case study not only specifies behavioural properties but also functional requirements mostly in the form of timing constraints. $\mathcal{CL}$does not support the specification of timing constraints nativly; however, one could encode these constraints in the definition of the actions. We have only done this in cases where the timing constraint affected the behaviour of the system in order to avoid access clutter.

## 2 UC3 - Order Products

| Action | Description |
|---|---|
| MgrSOP | Manager initiates the start of the Order Products process |
| SysLstItems | The System lists all the products |
| SysLstLowItems | The system lists the products which are running out of stock |
| MgrEntersAmm | The store manager chooses the product items to order and enters the corresponding amount |
| MgrPressOrder | The store Manager presses the Order button |
| SysPlaceOrder | The System places the order to the appropriate supplier |
| SysDispOrderID | The system displayes the order identifier generated to the Store Manager |

Table 1: UC3 Action Definition

### 2.1 Specification

1. $[MgrSOP]O(SysLstItems \& SysLstLowItems)$
2. $[SysLstItems \& SysLstLowItems]P(MgrEntersAmm)$

3. $[MgrEntersAmm]P(MgrPressOrder)$
4. $[MgrPressOrder]O(SysPlaceOrder\&SysDispOrderID)$

### 2.2 Description

The specification of this use case is quite straightforward since the example does not consider any exceptional behaviour and thus only a sequence of actions that are have to or may occur. Once the manager starts the order products process ($MgrSOP$) the system is obliged to show the list of items and the list of items running low ($SysLstItems\&SysLstLowItems$). After this the manager has the permission to enter the amount of the items he would like to order ($MgrEntersAmm$) after which he is permitted to press the order button ($MgrPressOrder$) in which case the system is obliged to place the order and display the order id ($SysPlaceOrder\&SysDispOrderID$).

## 3 UC4 - Receive Order Products

| Action | Description |
|---|---|
| SppDeliver | Supplyer delivers the ordered stock which is identified by an order ID |
| SppCmpCrr | Supplier made a complete and correct delivery. This is checked by the Stock Manager |
| MgrRec | Manager receives the order by pressing the button Roll in received order |
| SysUpdateInv | The System updates the inventory |
| MgrSendsBack | The Stock Manager sens the products back to the supplier |

Table 2: UC4 Action Definition

### 3.1 Specification

1. $[SppDeliver]O_{O(MgrSendsBack)}(SppCmpCrr)$
2. $[SppCmpCrr]O(MgrPressOrder)$
3. $[MgrPressOrder]O(SysUpdateInv)$

### 3.2 Description

The case study describes that that Manager is required to check that the supplier has sent the correct and complete order. Instead of defining an action $MgrChecksOrder$ we defined the action $SppCmpCrr$ since the obligation is on the supplier to send the correct information. If however the supplier has violated this obligation, the manager is obliged to send the order back ($MgrSendsBack$), otherwise he is obliged to process the order ($MgrPressOrder$) and the system is obliged to update accordingly ($SysUpdateInv$).

## 4 UC5 - Show Stock Reports

| Action | Description |
|---|---|
| MgrStoreID | Manager enters the store identifier and presses the button Create Report |
| SysDispReport | System displaces a report including all the available stock items in the store. |

Table 3: UC5 Action Definition

### 4.1 Specification

1. $[MgrStoreID]O(SysDispReport)$

### 4.2 Description

Once the manager enters the store id ($MgrStoreID$) the system is obliged to display the report($SysDispReport$).

## 5 UC6 - Show Delivery Reports

| Action | Description |
|---|---|
| MgrEntID | Managerenters the enterprise identifier and presses the button Create Report |
| SysDispReportEnt | The System generates and displays an Enterprise report |

Table 4: UC6 Action Definition

### 5.1 Specification

1. $[MgrEntID]O(SysDispReportEnt)$

### 5.2 Description

This Use Case is similar to the previous.

| Action | Description |
|---|---|
| MgrReqOverview | Manager initialtes the chape price process by requesting the listing of all the available products in the store |
| SysLstItems | The System lists all the products |
| MgrSelectsItem | The Manager Selects an Item |
| MgrChgPrice | The Manager changes price |
| MgrPressCommit | The Manager commits by pressing enter |
| SysChangesPrice | The System changes the price according to the amount set by the manager |

Table 5: UC7 Action Definition

# 6 UC7 - Change Price

## 6.1 Specification

1. $[MgrReqOverview]O(SysLstItems)$
2. $[SysLstItems]P(MgrSelectsItem)$
3. $[MgrSelectsItem]P(MgrChgPrice)$
4. $[MgrChgPrice]P(MgrPressCommit)$
5. $[MgrPressCommit]O(SysChangesPrice)$

## 6.2 Description

This use case shows the process of how a manager may change a price of an item. The manager starts this process by requesting a list of available products($MgrReqOverview$). The system is obliged to list all the items ($SysLstItems$) and give permission to the manager to choose items ($MgrSelectsItem$). If the manager does select an item, the system should give permission to the manager to change the price ($MgrChgPrice$) afterwhich it should give permission for the manager to commit the price change ($MgrPressCommit$). If the manager commits the changes, the system is obliged to make these changes permanent ($SysChangesPrice$).

# 7 UC8 - Product Exchange Among Stores

## 7.1 Specification

1. $[ProdRunsOut]O(StrServLowStock)$
2. $[StrServLowStock]O_{O(StrServQueueReq)}(StrServReqEnt)$
3. $[StrServReqEnt]O(EntServInvReq)$

| Action | Description |
|---|---|
| ProdRunsOut | A product of a store runs out |
| StrServLowStock | The store server recognizes low stock of the product. |
| StrServReqEnt | The Store Server sends a request to the Enterprise Server |
| EntServInvReq | The enterprise server sends an Inventory request to nearby stores |
| StrXRepInv | The store replies with the inventory information |
| EntServUpdates | The enterprise server updates the database and does a database look-up for the product |
| EntServChooses | The enterprise server using an ”optimization criterion” chooses from which store to request the transfer |
| EntServRecStr | The enterprise server sends a message to the receiving store. |
| EntServTfrStr | The enterprise server sends a message to the transferring store |
| StrServQueueReq | Store server queues request to enterprise. |
| 15min 15 minutes have passed | |
| AllReqReceived | All requests have been received |

Table 6: UC8 Action Definition

4. $[EntServInvReq]O(StrXRepInv)$
5. $[StrXRepInv]O(EntServUpdates)$
6. $[15min + AllReqReceived]O(EntServChooses)$
7. $[EntServChooses]O(EntServRecStr\&EntServTfrStr)$