CoCoME REPORT

Table des matières

Introduction..2
CoCoME..2
Inventory :..2
Data :..2
Application :..3
Gui :..3
CashDeskLine :..3
Plug-in..4

Introduction

CoCoME is the abbreviation of Common Component Modeling Example.
This is a program representing a sell system. We won't detail how it works because it's not what we need. We will focus on the architecture of the program, we want to analyze components, interfaces and other elements we need, in order to have a base example to use and to see what will give us the result of the plug-in we are working on.

CoCoME

The major component is TradingSystem.
At the base of the program we have tow major components named Inventory and CashDeskLine.
Inventory and CashDeskLine are connected with two interfaces:
CashDeskConnectorIf is the provided interface of Inventory and required by CashDeskLine.
SaleRegisteredEvent is provided by CashDeskLine and required by Inventory.
Bank is provided by CashDeskLine and is the external interface of the complete program

Inventory :

This component contains four components:

- Gui, this is the global user interface which require the informations provided through the interfaces StoreIf and ReportingIf provided by the Application component.
- Application, it makes the link between Inventory and the other contained components. That's why it has the interfaces SaleRegisteredEvent and CashDeskConnectorIf. It needs informations from Data, so it has three interfaces required, those are named EnterpriseQueryIf, PersistenceIf, StoreQueryIf.
- Data has the required interface JDBC
- DataBase.

Data :

- Enterprise provide EnterpriseQueryIf
- Persistence provide PersistenceIf
- Store provide StoreQueryIf

These three components manage informations from their name (the component Enterprise manage information from the enterprise, ...)
Each of the components contains base classes. Like for the component **Enterprise** :

- TradingEnterprise
- ProductSupplier
- Product

Application :

- Reporting
  - Interface provided → ReportingIf (for Gui)
  - Interface Required → EnterpriseQueryIf, PersistenceIf, StoreQueryIf
- Store
  - Interface provided → CashDeskConnectorIf (for Inventory), StoreIf (for Gui)
  - Interface Required → PersistenceIf, StoreQueryIf, SaleRegisteredEvent
- ProductDispatcher
  - Interface provided → ProductDispatcherIf
  - Interface Required → EnterpriseQueryIf, PersistenceIf, StoreQueryIf

Gui :

- Reporting
- Store

For the rest of the program we will not focus on the part about the CashDesk, but we'll see major components.

CashDeskLine :

- CashDesk
- EventBus
- Coordinator

We will not detail all of the components present in CashDesk, they are principally components used for controllers like ScannerController, CardReaderController.

Plug-in

Now we will talk about the result we are having by passing the CoCoME in the plug-in, and we will check if results seems correct to our previous analyze of the architecture.

After we've used the plug-in, we obtain :

Around 59 components for all of the program, which is really more important than we should have. Moreover we have some components like Store which are not detected as components by the plug-in and this for every classes that composes his package. There are some other component identified but some should not be, like some classes used for tests of the database in order to fill it. They are also tag as ROOT component and we don't think it's could be true event if they were components. We've just take these examples, maybe they aren't relevant, but they are chosen to point on incoherences in the plug-in. So we must take the rules and determines why we have those results and correct them.

We did not find any cycle in the program, this should be good, maybe we could make an example with a cycle in order to test the rules why manage this point.

Most of the binding are not resolved.

Seems that most of the interfaces are identified, for the whole program, the plug-in has identified 120 interface, but due to the result, we probably have more than we have really, 120 seems to be pretty huge, but it certainly is coherent if we take in consideration the number of components identified. But has we only need to see the key “interface” in the program, this should be strange to identify other interfaces.

In the same way, communication are few in comparison with the number of interfaces but most of them are pertinents.

To finish with this, we will simply say that most of the plug-in identify what we need, but it needs to be refined in order to match everything we need and correctly.