Formal Modeling of an Online Exchange System for Complex Goods in VDM++

Mestrado Integrado em Engenharia Informática e Computação

Métodos Formais em Engenharia de Software

Hugo Cardoso - 201105625

Vasco Gomes - 201106906

Faculdade de Engenharia da Universidade do Porto

Rua Roberto Frias, sn, 4200-465 Porto, Portugal

December 16, 2014

Contents

1. Informal system description and list of requirements

1.1 Informal system description

In the growing online space where auctions and bulletin boards reach new lengths every year, exchange markets stand out due to their fast-paced trading and guaranteed symmetry between buyers and sellers.

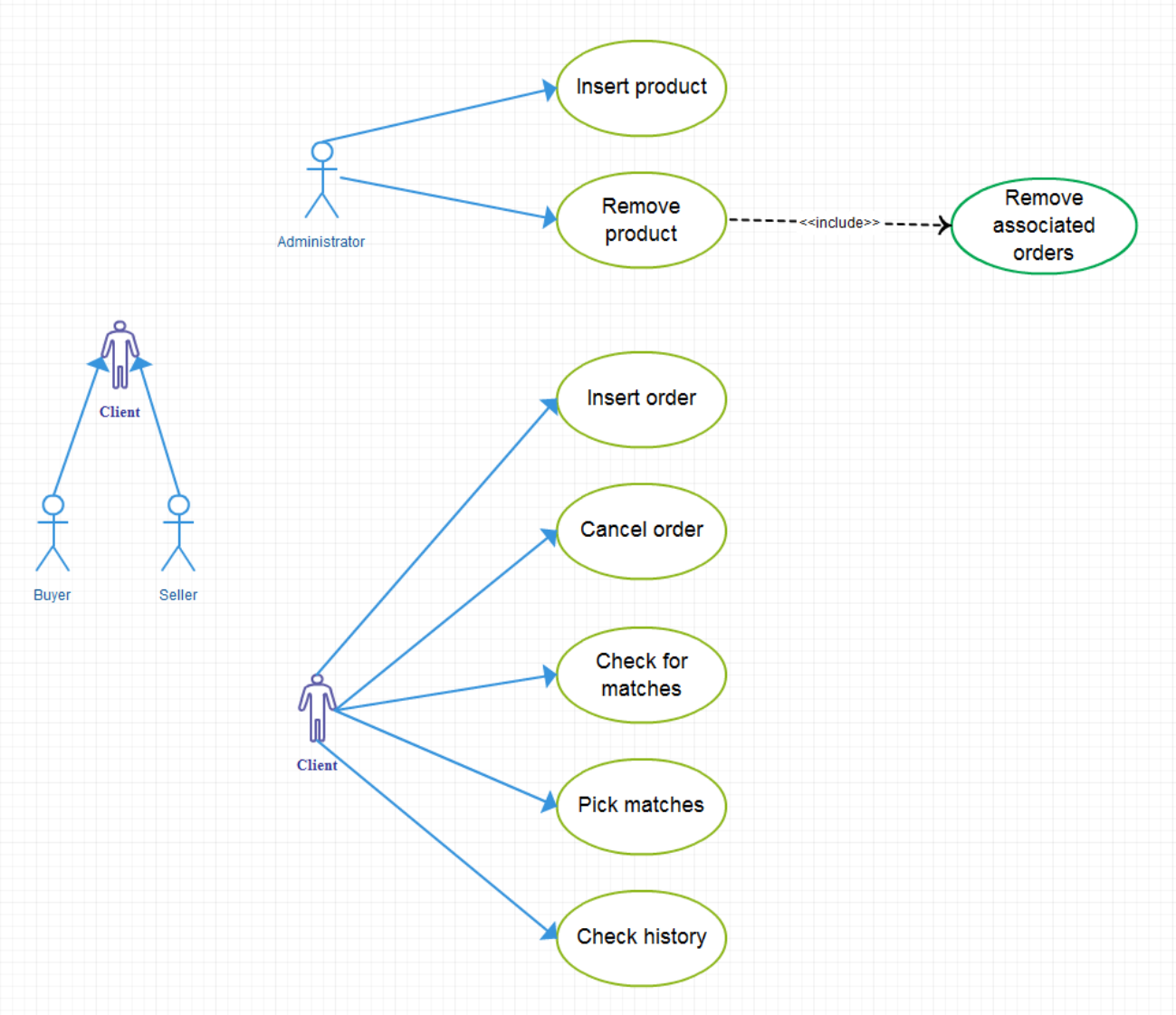
The system here being modeled represents an online exchange system where users should be able place orders by referring a product and describing their attributes, and then attempt to trade them. Each order is composed of a product, type (buying or selling) and list of attributes. The system should be able to create matches accordingly.

1.2 List of requirements

Id	Priority	Description
R1	Mandatory	The system administrator should be able configure the products that are allowed to be bought/sold.
R2	Mandatory	The system administrator should be able to remove a product and all orders associated to it.
R3	Mandatory	The end user (buyer or seller) should be to place a new order on the system (either buying or selling).
R4	Mandatory	The end user (buyer or seller) should be able to cancel an order previously placed if it has not yet been matched.
R5	Mandatory	The end user (buyer or seller) should be able to check all orders that match a given order.
R6	Mandatory	The end user (buyer or seller) should be able to make a trade by matching two given orders.
R7	Mandatory	The end user (buyer or seller) should be able to check all transactions previously completed.

2. Visual UML model

2.1 Use case model



The major use case scenarios, later used as test scenarios, are described below:

Scenario	**Insert Product**
Description	Normal scenario for inserting new products into the system, to make them useable in future orders.
Pre-conditions	None
Post-conditions	1. Inserted product exists in the system.
Steps	(unspecified)
Exceptions	(unspecified)

Scenario	**Remove Product**
Description	Alternative scenario for removing a product from the system, while also removing all orders of said product.
Pre-conditions	1. Product to remove exists in the system.
Post-conditions	1. Removed product is not in the system. 2. All orders of the removed product are also removed from the system.
Steps	1. An admin inserts a product. 2. A user creates an order of said product. 3. An admin removes the inserted product. 4. Both the product and the order are gone from the system.
Exceptions	(unspecified)

Scenario	**Insert Order**
Description	Normal scenario for inserting a new order of a product into the system (either buying or selling).
Pre-conditions	1. Product of the order exists in the system. 2. The order starts unfulfilled.
Post-conditions	1. Order exists in the system.
Steps	1. An admin inserts a product. 2. A user creates an order of said product.
Exceptions	(unspecified)

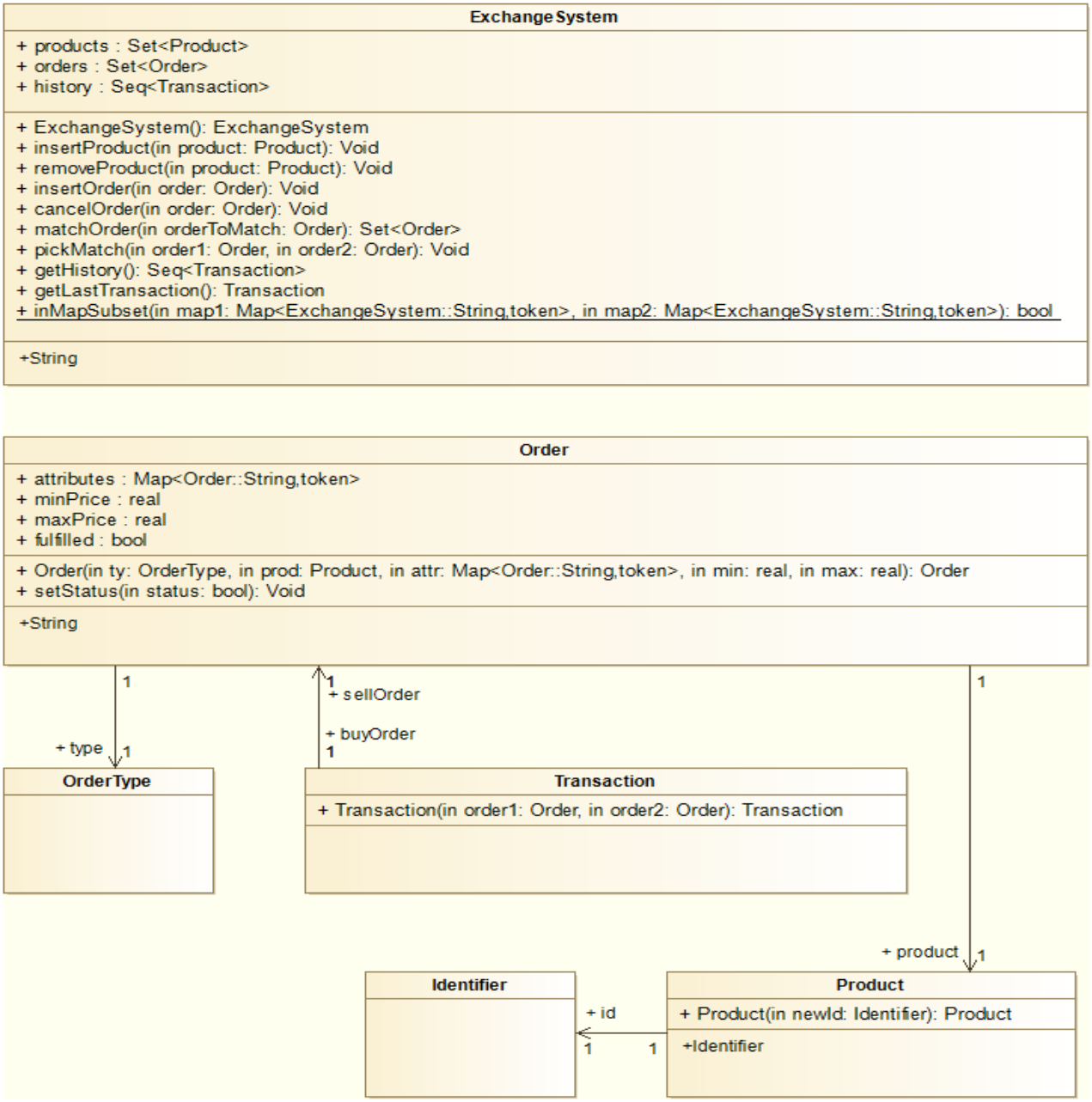
Scenario	**Cancel Order**
Description	Alternative scenario for canceling an order previously placed, which has not yet been matched.
Pre-conditions	1. Order to cancel exists in the system.
Post-conditions	1. Canceled order is not in the system.
Steps	1. An admin inserts a product. 2. A user creates an order of said product. 3. The user cancels the order. 4. The order is gone from the system.
Exceptions	(unspecified)

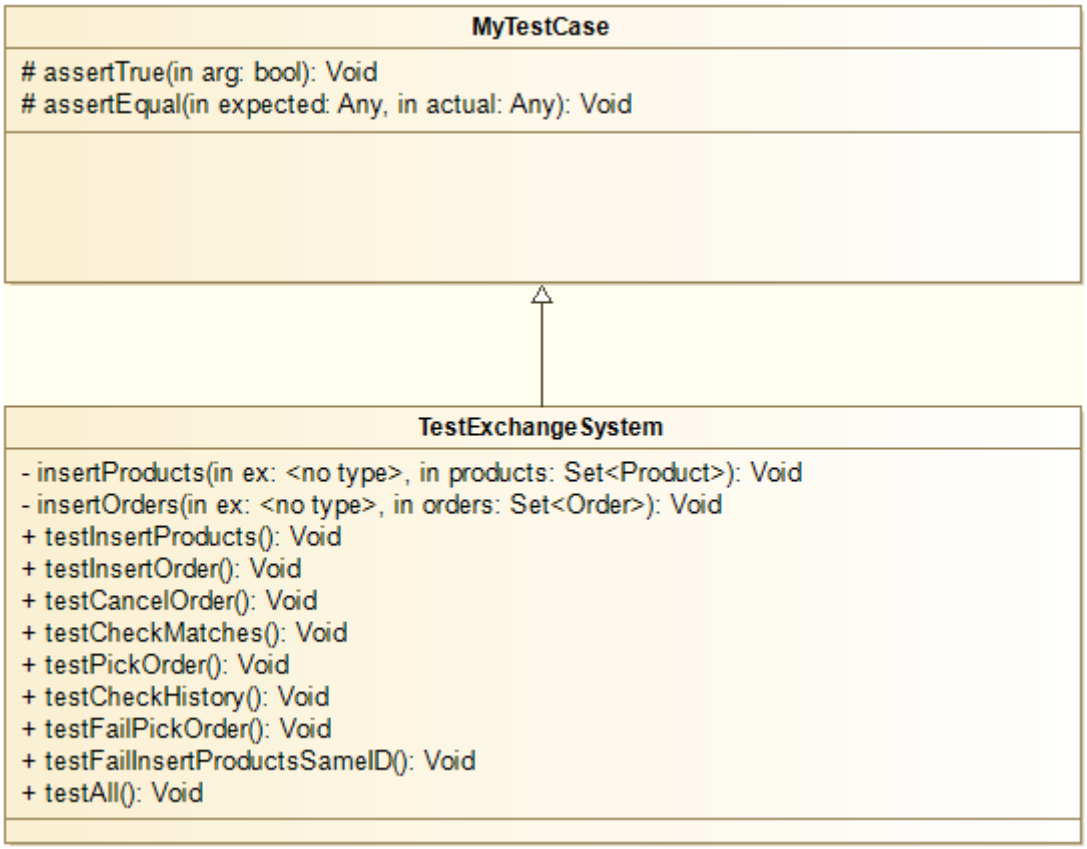
Scenario	**Check Matches**
Description	Normal querying scenario for checking all orders that match a given order.
Pre-conditions	1. Order to match exists in the system.
Post-conditions	None
Steps	1. An admin inserts a product. 2. A user creates an order for selling said product. 3. Another user creates an order for buying the same product. 4. Checking matches of first order returns second order. 5. Checking matches of second order returns first order.
Exceptions	(unspecified)

Scenario	**Make Trade**
Description	Normal scenario for executing a transaction, closing the deal between a buyer and a seller who match.
Pre-conditions	1. The two orders are different. 2. One order is of buying and the other is of selling. 3. Both orders are associated with the same product. 4. The first order's minimum price is less or equal than the second order's maximum price. 5. The second order's minimum price is less or equal than the first order's maximum price. 6. The sell order's attributes contain at least the buy order's attribute requirements (seller satisfies buyer requisites).
Post-conditions	Both orders are fulfilled. Both orders were removed from the system. A transaction composed by both orders was created on the system history.
Steps	1. An admin inserts a product. 2. A user creates an order for selling said product. 3. Another user creates an order for buying the same product. 4. A trade is made between both users' matching orders.
Exceptions	(unspecified)

Scenario	**Check Transaction History**
Description	Normal querying scenario for checking all transactions previously completed.
Pre-conditions	None
Post-conditions	None
Steps	(unspecified)
Exceptions	(unspecified)

2.2 Class model





Class	Description
`ExchangeSystem`	Core model; defines the state variables and operations available to the users.
`Product`	Defines a product that may be bought/sold in the exchange system.
`Order`	Defines an order that may be placed in the exchange system.
`Transaction`	Defines a transaction that previously took place in the exchange system. A transaction is composed of a pair of orders.
`MyTestCase`	Superclass for test classes; defines `assertEquals` and `assertTrue`.
`TestExchangeSystem`	Defines the test/usage scenarios and test cases for the exchange system.

3. Formal VDM++ model and model validation

The formal VDM++ model and respective validation, along with code coverage specifications, can be consulted in the attached PDF.

4. Conclusions

Reaching the conclusion of the project, we believe the model developed covers all the requirements specified. Although the project has reached its final stage, not all cases have been tested and, if time permitted, more tests should have been added to ensure full code (and scenario) coverage. Nevertheless, we believe the projected was completed successfully.

The entire project was developed in pair-programming and, as such, both members believe all the work was shared equally. In its entirety, the project took approximately 15 hours to develop.

5. References

1. A Formal Specification in B of a Medical Decision Support System, Christine Poerschke, David E. Lightfoot, and John L. Nealon, LNCS 2651, pp. 497–512, 2003
2. VDM-10 Language Manual, Peter Gorm Larsen et al, Overture Technical Report Series No. TR-001, March 2014
3. Overture tool web site, http://overturetool.org