Use Cases Combined with Block Diagrams by Texel and Williams

Use Case 16: Operator_Updates_Nominal_Values_In_DB

Overview:

This Use Case enables the Operator to change the nominal values of all three environmental conditions in the database. These updated values are the nominal values against which the values detected by the sensors are compared to reflect percent deviation.

Preconditions:

1. There are no alarms currently active.
2. SEM_Desktop_View is displayed.
3. The database is accessible.

Scenario:

Action	Software Reaction
1. Operator clicks on the Update All button on the SEM_Desktop_View.	1. Update_All_View pop-up appears.
2. Disable the alarm.	2. Alarm disabled.
3. Enter Air Pressure value.	3. Air Pressure field is updated.
4. Enter Oxygen value.	4. Oxygen field is updated.
5. Enter Temperature value.	5. Temperature field is updated.
6. Operator clicks on the OK button.	6. The Update_All_View pop-up is destroyed, the DB is updated, and the Operator is returned to the SEM_Desktop_View.
7. Operator clicks on the Cancel button.	7. The Update_All_View is destroyed and monitoring continues. The database is not updated.
8. Enable alarm.	8. Alarm is enabled.

Scenario Notes:

Items 3, 4, and 5 may be done in any order. Additionally the Operator does not have to update all three values. This Use Case *permits* the modification of all three values, *but* the Operator may choose to update one, two, or all three. Steps 6 and 7 are mutually exclusive. Step 8 happens regardless of whether 6 or 7 was selected.

Post Conditions:

1. The nominal Air Pressure value is updated in the DB (if OK button was selected).
2. The nominal Oxygen value is updated in the DB(if OK button was selected).
3. The nominal Temperature value is updated in the DB(if OK button was selected).
4. The Operator is returned to the Desktop.
5. The alarm is enabled.

Exceptions:

1. The DB cannot be accessed.

Use Cases Utilized:

None

Required GUI:

1. SEM_Desktop_View
2. Update_All_View pop-up

Timing Constraints:

None

Figure 2-5. Example Scenario: Use Case 16

Use Case 1.1 Use Case Horror: Example of a Poorly Written Use Case

Register for Course (Main Scenario, Poorly Written Version)

1. Display a blank schedule.
2. Display a list of all classes in the following way: The left window lists all the courses in the system in alphabetical order. The lower window displays the times the highlighted course is available. The third window shows all the courses currently in the schedule.
3. Do
4. Student clicks on a course.
5. Update the lower window to show the times the course is available.
6. Student clicks on a course time and then clicks on the "Add Course" button.
7. Check if the Student has the necessary prerequisites and that the course offering is open.
8. If the course is open and the Student has the necessary prerequisites, add the Student to the course. Display the updated schedule showing the new course. If no, put up a message, "You are missing the prerequisites. Choose another course."
9. Mark the course offering as "enrolled" in the schedule.
10. End do when the Student clicks on "Save Schedule."
11. Save the schedule and return to the main selection screen.

www.informit.com/articles/article.asp?
p=30084&seqNum=18rl=1

Use Case 1.3 ***Register for Courses: Use Case with Extensions***

Register for Courses (Use Case with Extensions)

Primary actor: Student

System under Discussion: Course Enrollment System

Level: User Goal

1. Student requests to construct a schedule.
2. The system prepares a blank schedule form.
3. The system pulls in a list of open and available courses from the Course Catalog System.
4. Student selects up to 4 primary course offerings and 2 alternate course offerings from the available offerings.
5. For each course, the system verifies that the Student has the necessary prerequisites and adds the Student to the course, marking the Student as "enrolled" in that course in the schedule.
6. When the Student indicates the schedule is complete, the system saves the schedule.

Extensions:

1a. Student already has a schedule:

System brings up the current version of the Student's schedule for editing instead of creating a new one.

1b. Current semester is closed and next semester is not yet open:

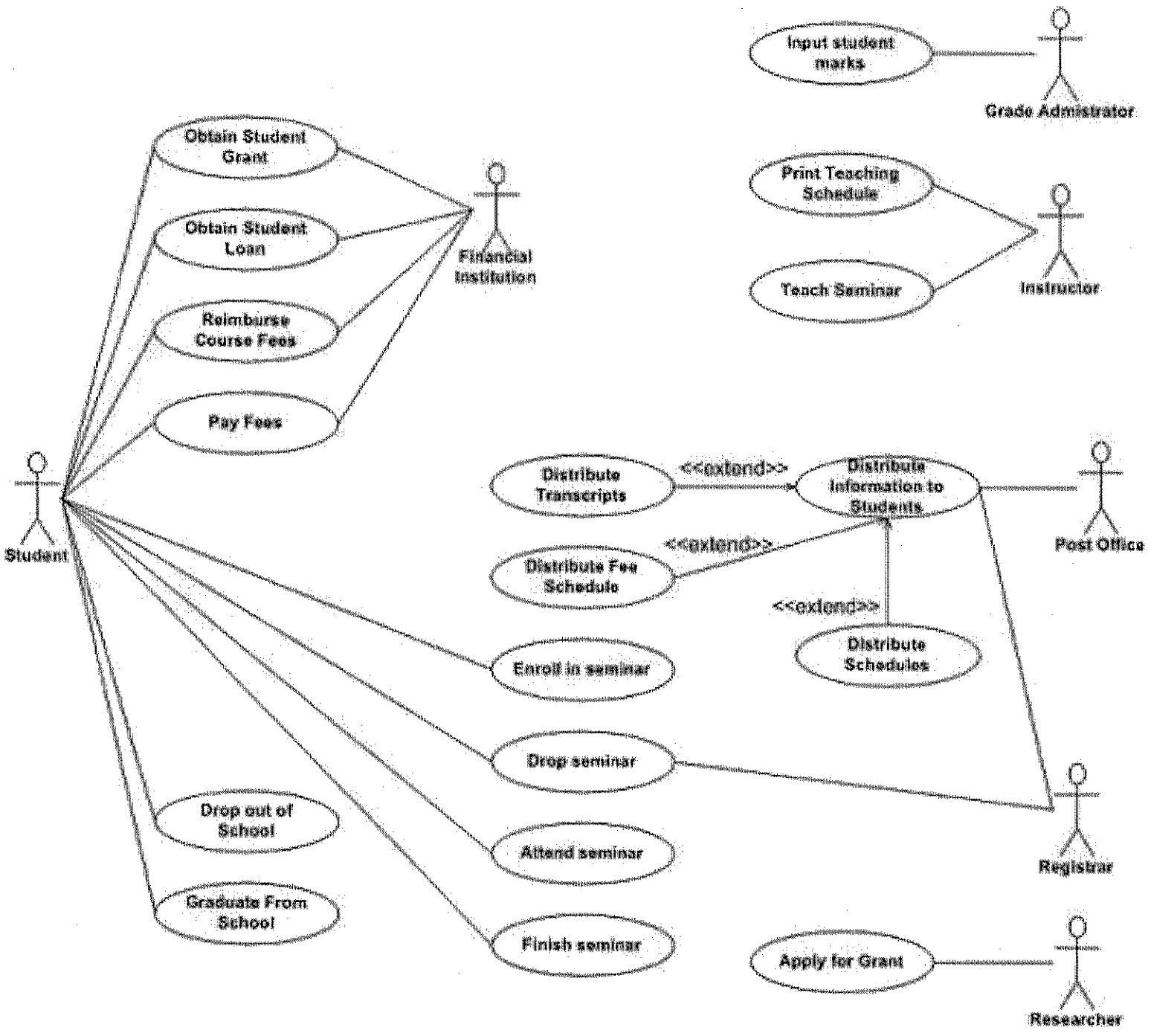
System lets Student look at existing schedules, but not create new ones.

3 a. Course Catalog System does not respond:

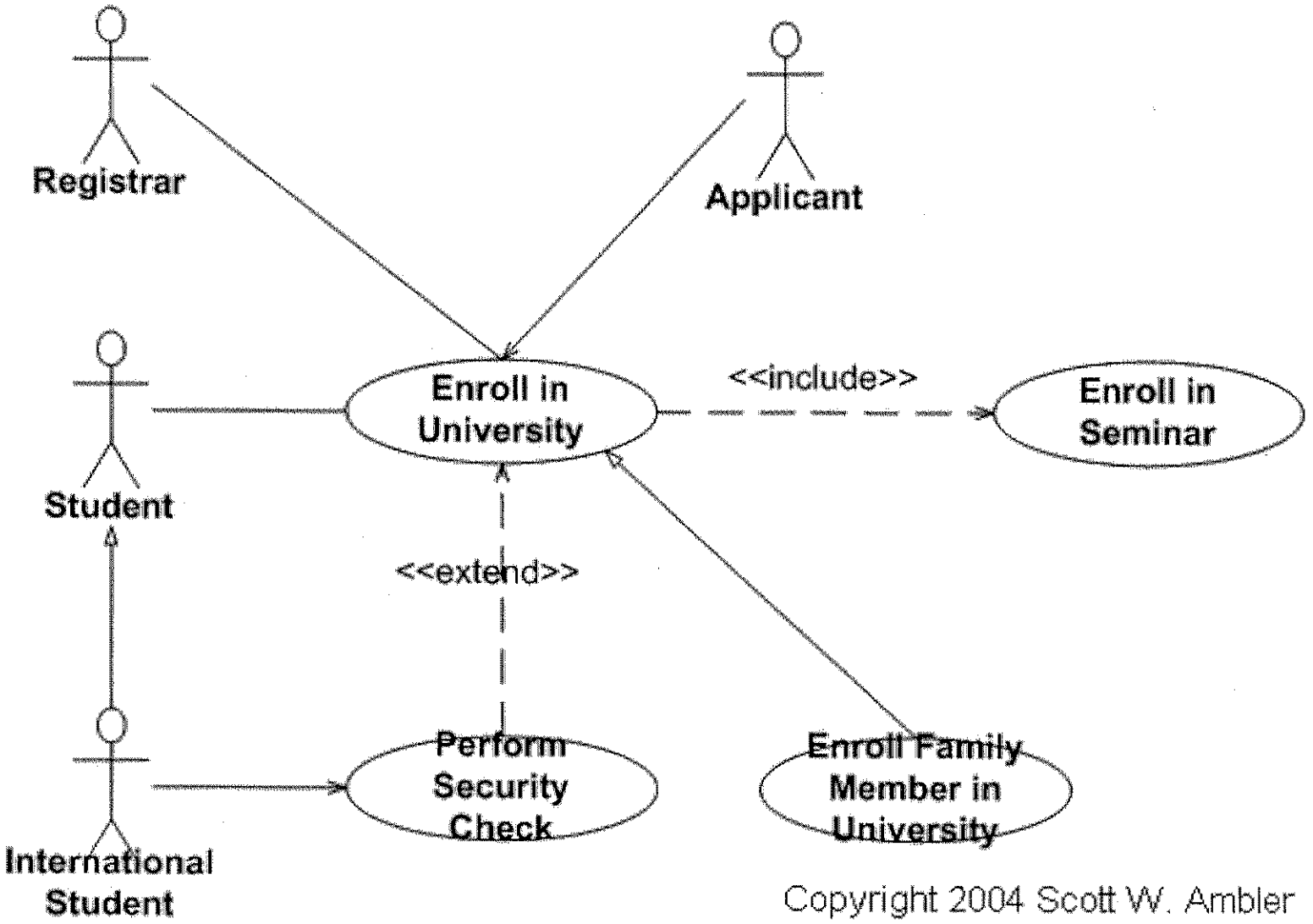
The system notifies the Student and terminates the use case.

4 a. Course full or Student has not fulfilled all prerequisites:

System disables selection of that course and notifies the Student.



www.agilemodeling.com/artifacts/useCaseDiagrams.htm

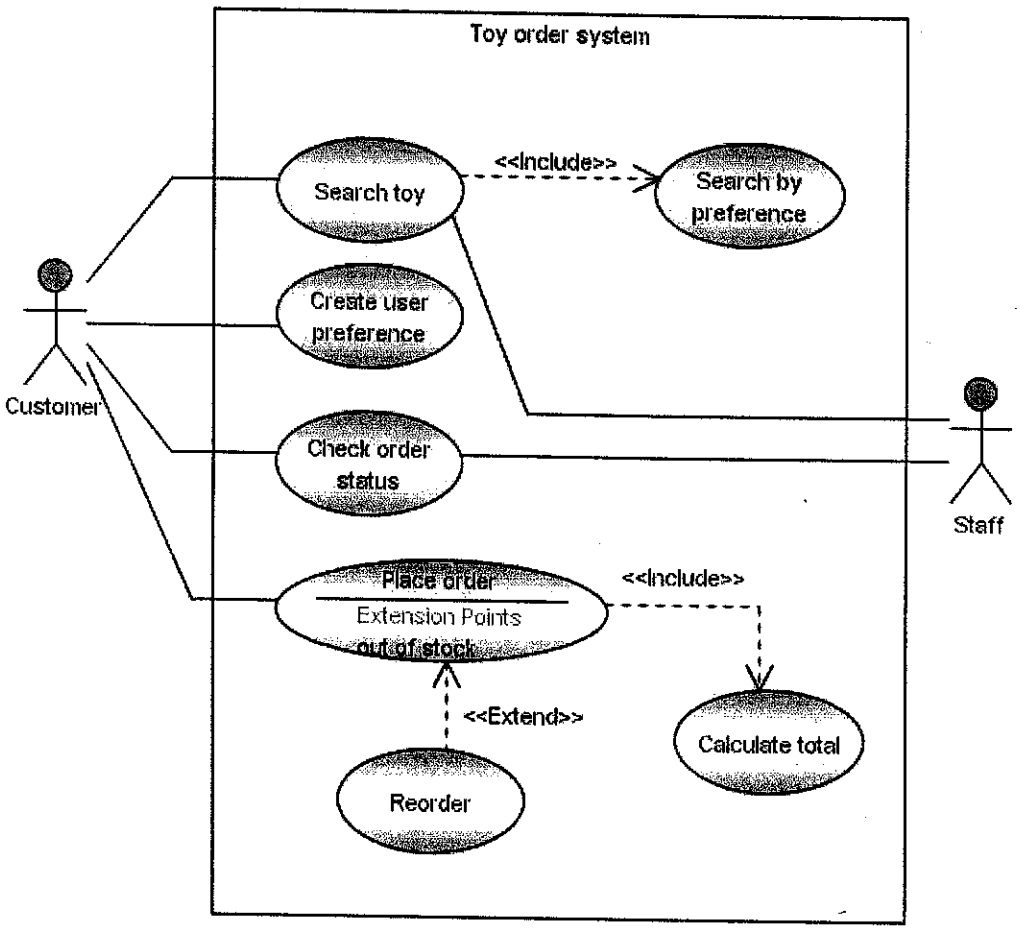


www.agilemodeling.com/artifacts/useCaseDiagrams.htm

Use Case Diagram

A Use Case Diagram is a diagram that help system analyst to discover the requirements of the target system from the user's perspective. The usage of use case diagram includes:

- Describes the behavior of a system from a user's standpoint
- Provides functional description of a system and its major processes.
- Provides graphic description of the users of a system and what kinds of interactions to expect within that system
- Displays the details of the processes that occur within the application area



A Sample Use Case Diagram

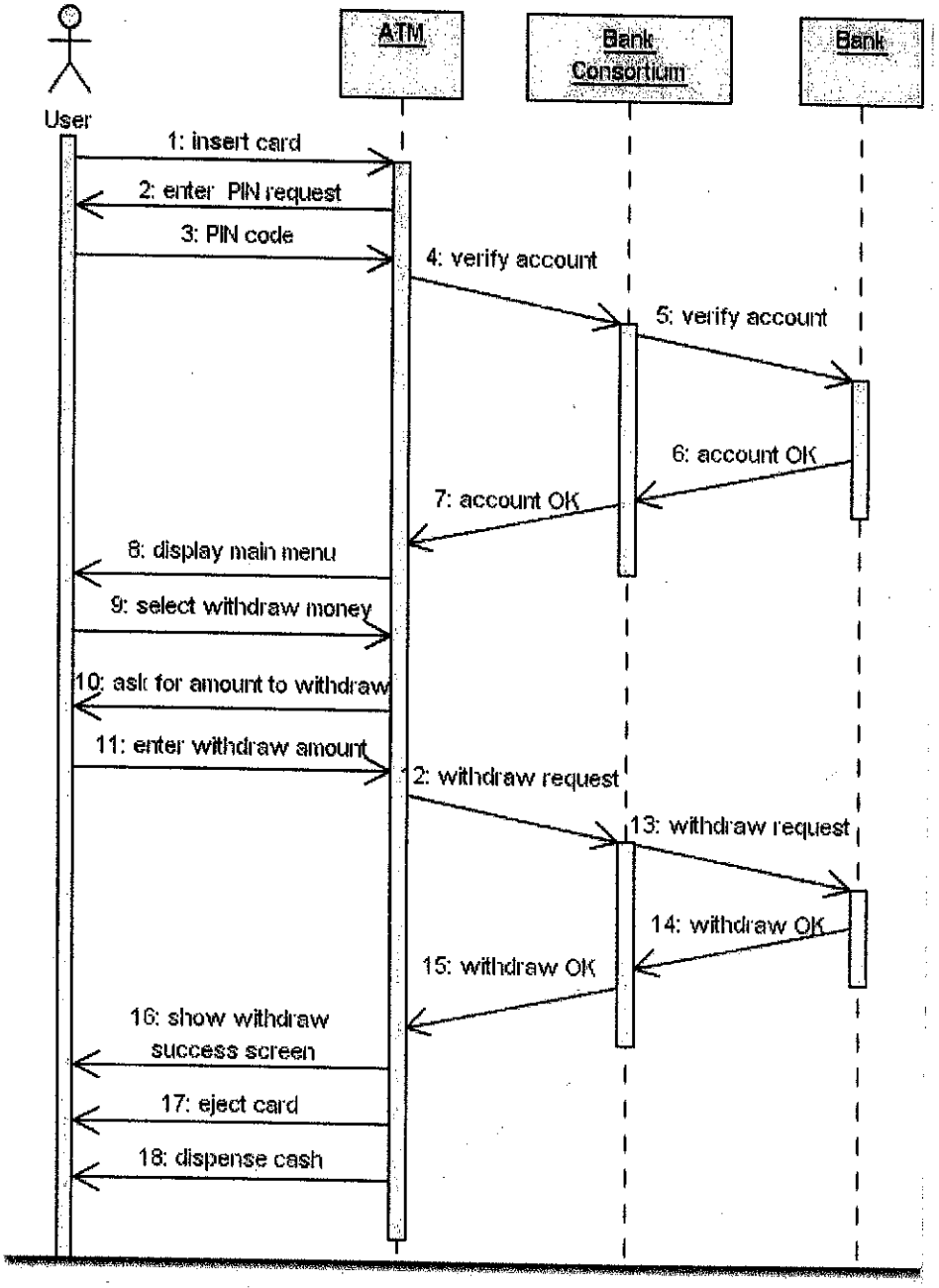
VisualParadigm

Gallery

http://www.visual-paradigm.com/VPGallery/diagrams/UseCase.html

Sequence Diagram

A Sequence diagram is a model that describes how groups of objects collaborate in some behavior over time and capturing the behavior of a single use case. It shows the objects and the messages that are passed between these objects in the use case.

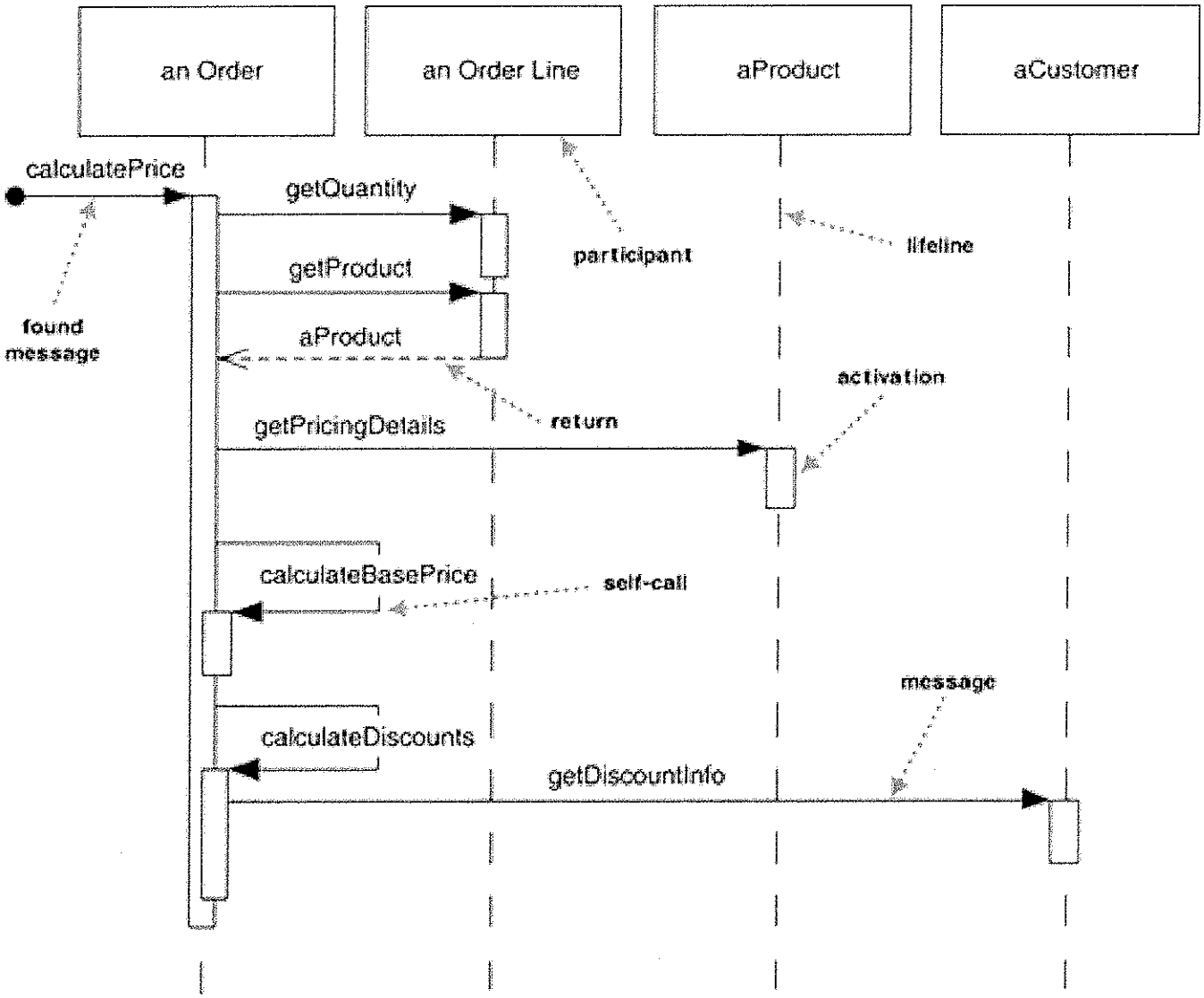


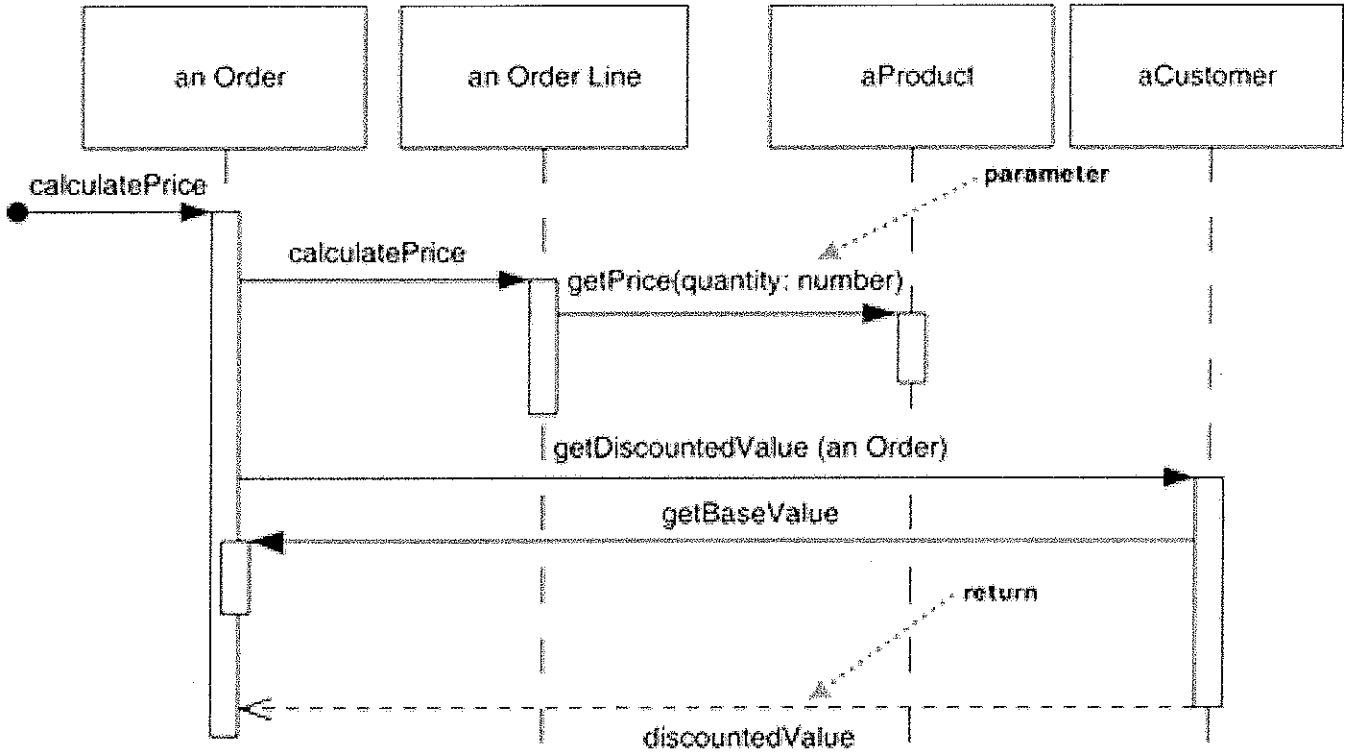
A Sample Sequence Diagram

VisualParadigm

Gallery

http://www.visual-paradigm.com/VPGallery/diagrams/Sequence.html


an Order
an Order Line
aProduct
aCustomer
calculatePrice
getQuantity
getProduct
aProduct
getPricingDetails
calculateBasePrice
calculateDiscounts
getDiscountInfo
found message
participant
lifeline
return
activation
self-call
message


an Order
an Order Line
aProduct
aCustomer
calculatePrice
parameter
calculatePrice
getPrice(quantity: number)
getDiscountedValue (an Order)
getBaseValue
return
discountedValue