Project Report

Computer Science DMAI-0914 – Group 5

University College of Northern Denmark

Adrian Frunza Alex Muriuki Njogu Kristupas Sakalius

Monika Vyšniauskaitė Stefan Patatu

Workshop Persistence Project

Technology and Business Computer Science Bachelor Degree DMAI-0914

Project participants

Adrian Frunza	Signature: _____
Alex Muriuki Njogu	Signature:_____
Kristupas Sakalius	Signature:_____
Monika Vyšniauskaitė	Signature:_____
Stefan Patatu	Signature: _____

Supervisors

Istvan Knoll
Ann Francke Riisberg

Submission date

20.03.2015

Introduction

In this workshop we are to show the knowledge we have acquired in the second semester so far in connection with creation and maintenance of databases and to get experience with accessing databases from Java programs. We were asked to create a part of a computer system for "Western Style Ltd.", fix the errors/mistakes in their domain model and make the necessary adjustments accordingly. We will try to cover the majority of things we have made below.

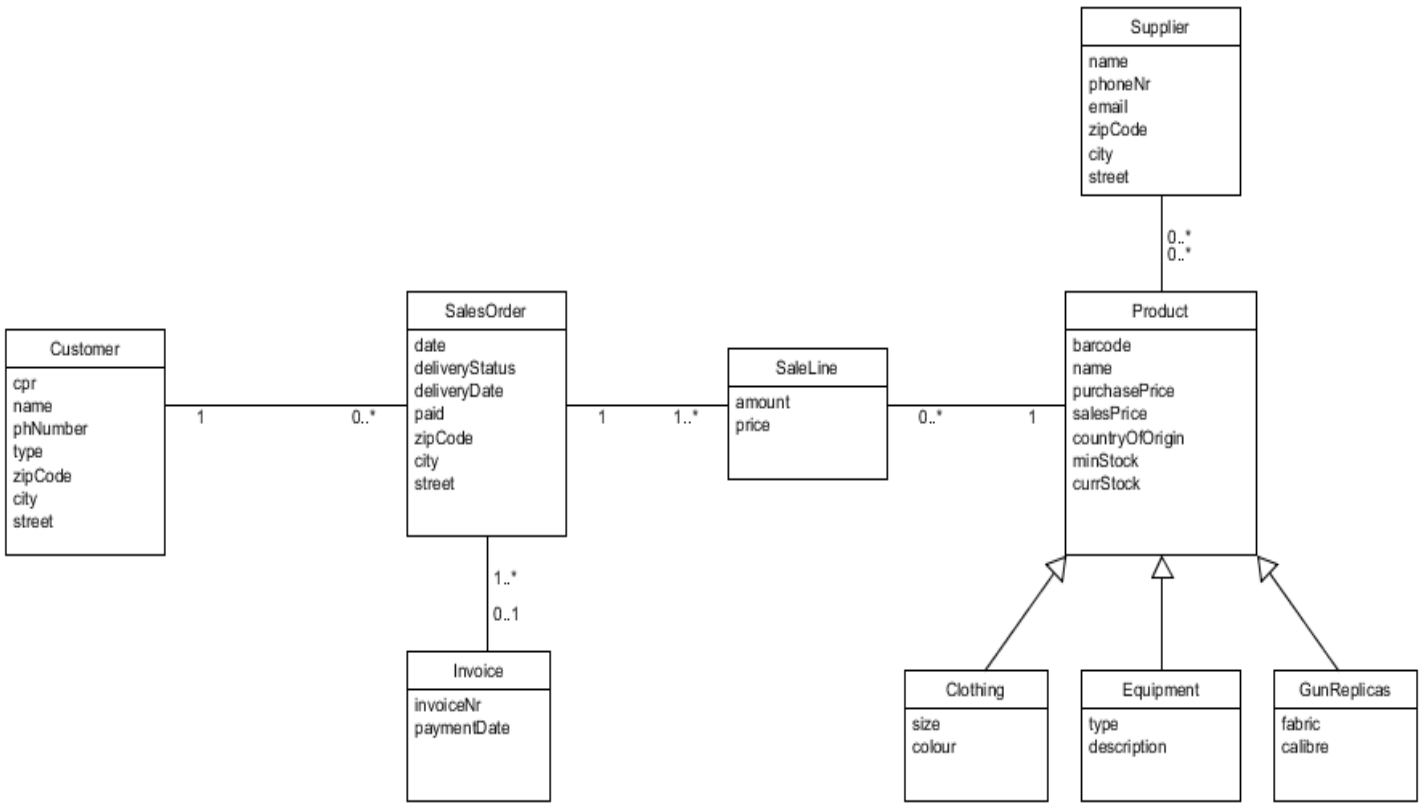
System development

Domain model

This is our domain model. All the classes, their respective attributes, connections, multiplicities can be seen there.
The first thing we decided to edit in the domain model was to add a saleLine between the product and the SalesOrder which would contain the amount of items purchased of that type and the price of a single unit.

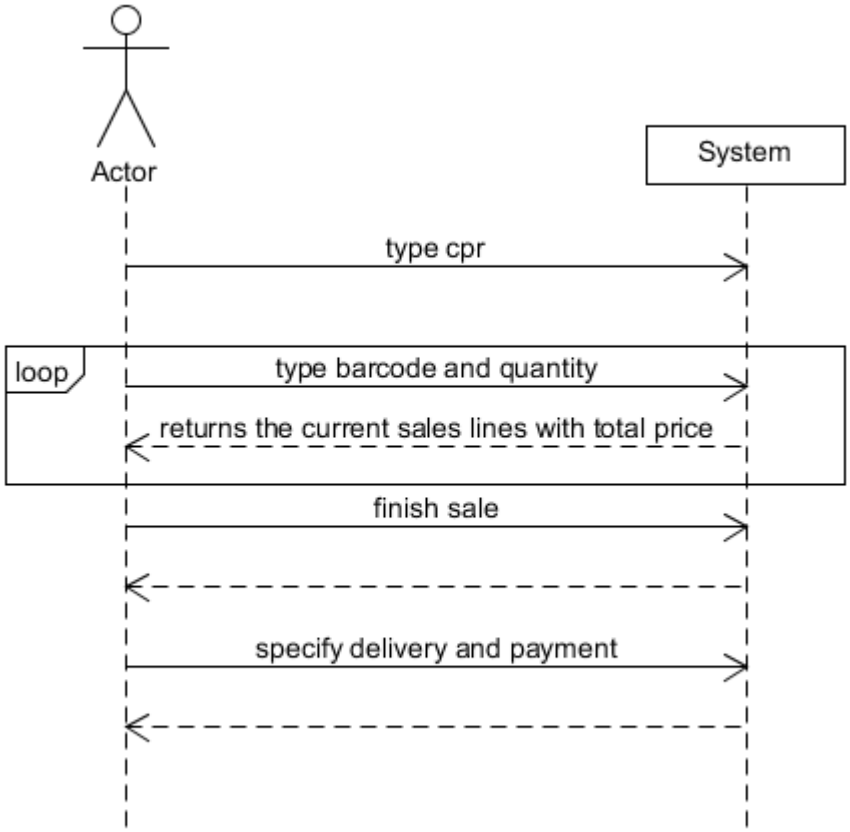
Then while thinking about the database we were going to connect our system to we decided that there should be another class called address. It would contain zip code, city and the street. This class then would be connected to customers, orders for deliveries and suppliers.

The SalesOrder at first was supposed to always have an invoice. However, after thinking more about it we found out we first create an order and only then an invoice. This means the multiplicity in the domain model had to be changed from 1 to 0..1 .



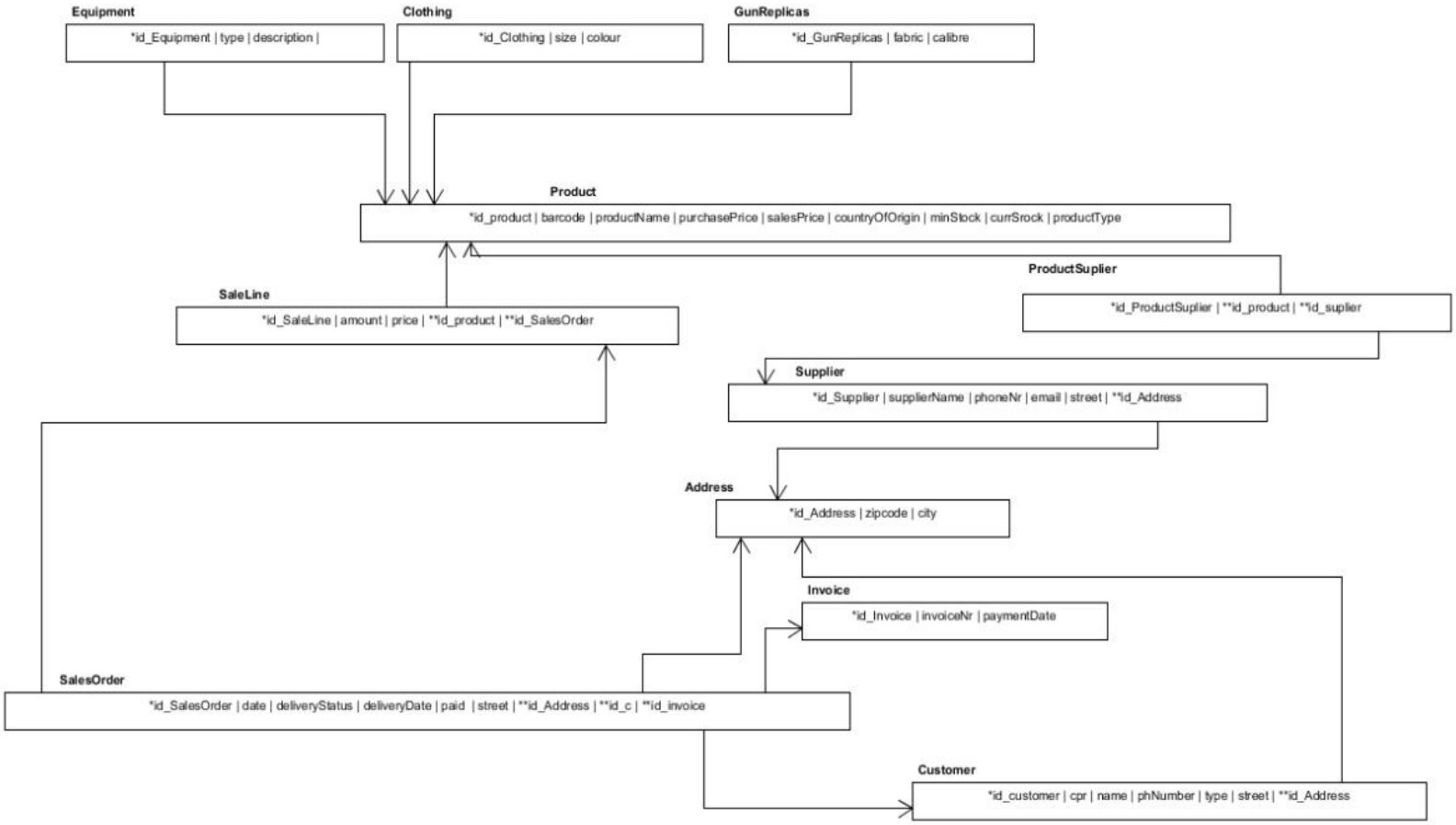
Sequence diagram

This is our sequence diagram. You can see how actor (cashier) interacts with the system and how it responds.



Relational model

The relational model shows us exactly how we are going to connect our tables in our database. We have started from the domain model by discussing how we should design the database.
One of the things discussed was about the way we should connect the customer to the order. We have decided to connect our “customer” table directly to the “order” table. The other proposal was to connect the customer to the “invoice” table and every time we have an order we assign it to an invoice. Our decision is based on the most efficient way of accessing the orders of a customer.
Also, we have normalized our database by creating the “address” table.
The relation “many to many” between product and supplier was solved by creating an additional table where we connect these two tables.



Fully dressed use case

Use case name	createSale	
Actors	Cashier	
Pre-conditions	Customer & Product(s) has to exist in the system	
Post-conditions	Sale is registered in the system	
Frequency	As many times as needed	
Main Success Scenario	1. Cashier types in customer's cpr to start the sale.	2. The system asks for a barcode and quantity.
	3. Cashier types in the product's barcode, quantity and the same attributes of all the products the customer wants to buy.	4. The system shows the price of each product purchased and updates total amount.
	5. Cashier clicks that all items were added	6. The system asks for the payment and delivery type
	7. Cashier chooses the up-front payment option and receives cash from customer.	8 The system shows that the sale has ended and prints out receipt.
Alternate Flows	1a. The customer doesn't exist in the system 3a. The product with that barcode doesn't exist 7a. Customer wants to pay at the end of the month.	

Fully Dressed Use Case

During design of the fully dressed use case, two trivial issues were encountered. The first one was how to uniquely identify the customer and the second one was determining which was the main success scenario between up-front payment and monthly payment.
To uniquely identify a customer, it was decided that the cashier keys in the customer's CPR and the main success scenario is when a customer pays upfront as it is much common in most business cases.
The fully dressed use case implemented is in two-column or conversational format. This is because of its clear visual representation between the actor and the system as opposed to the one-column style which is more compact. [Applying UML and Patterns – Craig Larman]
Main use case is createSale for creation of sales order because this is where the business makes their revenue from. The primary actor here is the cashier as he or she calls upon the services of the system in order to create a sale. Customer and products have to exist in the system forehand.

Main Success Scenario

The trigger event that starts the scenario in this case is the customer walking up to the cashier with items to purchase. The cashier keys in the customer's CPR to start the sale. The system then asks for the barcode and the quantity of item(s) to be purchased so as to countercheck with the current stock. After the cashier keys in the barcode and quantity, the system shows price of product and updates the total amount up to that point. Cashier asserts that all the purchased items were added so that the system asks for mode of payment and delivery type. Cashier chooses up-front mode of payment and receives cash for the purchased items from customer. System signals that the sale has ended and prints out a receipt.

Alternate Flows

Alternate flows are branches from the main success scenario and they are mostly a mix of "happy" and "unhappy" scenarios.
In this case, we have three alternate flows.

1. The customer does not exist in the system. A customer has to exist in the system for the sale to occur. Customer is created to handle this.

2. The product with that barcode does not exist. A product also has to exist in the system for the sale to occur. Product is created to handle this.

3. The customer wants to pay for the purchased items at the end of the month and not up-front. The cashier chooses the monthly payment option in order to bill the customer an invoice for the goods purchased at the end of the month.